2010-07-13

Data Sharing on Untrusted Storage with Attribute-Based Encryption

Shucheng Yu
Worcester Polytechnic Institute

Data Sharing on Untrusted Storage with Attribute-Based Encryption

by

Shucheng Yu

A Dissertation
Submitted to the Faculty
of the
WORCESTER POLYTECHNIC INSTITUTE
In partial fulfillment of the requirements for the
Degree of Doctor of Philosophy
in
Electrical and Computer Engineering

July 2010

Approved:

Professor Wenjing Lou ECE Department Dissertation Advisor	Professor Kaveh Pahlavan ECE Department Dissertation Committee
Professor Berk Sunar ECE Department Dissertation Committee	Professor Jie Wang CS Department, UMASS Lowell Dissertation Committee

Abstract

Storing data on untrusted storage makes secure data sharing a challenge issue. On one hand, data access policies should be enforced on these storage servers; on the other hand, confidentiality of sensitive data should be well protected against them. Cryptographic methods are usually applied to address this issue – only encrypted data are stored on storage servers while retaining secret key(s) to the data owner herself; user access is granted by issuing the corresponding data decryption keys. The main challenges for cryptographic methods include simultaneously achieving system scalability and fine-grained data access control, efficient key/user management, user accountability and etc. To address these challenge issues, this dissertation studies and enhances a novel public-key cryptography – attribute-based encryption (ABE), and applies it for fine-grained data access control on untrusted storage.

The first part of this dissertation discusses the necessity of applying ABE to secure data sharing on untrusted storage and addresses several security issues for ABE. More specifically, we propose three enhancement schemes for ABE: In the first enhancement scheme, we focus on how to revoke users in ABE with the help of untrusted servers. In this work, we enable the data owner to delegate most computation-intensive tasks pertained to user revocation to untrusted servers without disclosing data content to them. In the second enhancement scheme, we address key abuse attacks in ABE, in which authorized but malicious users abuse their access privileges by sharing their decryption keys with unauthorized users. Our proposed scheme makes it possible for the data owner to efficiently disclose the original key

owner's identity merely by checking the input and output of a suspicious user's decryption device. Our third enhancement schemes study the issue of privacy preservation in ABE. Specifically, our proposed schemes hide the data owner's access policy not only to the untrusted servers but also to all the users.

The second part presents our ABE-based secure data sharing solutions for two specific applications – Cloud Computing and Wireless Sensor Networks (WSNs). In Cloud Computing cloud servers are usually operated by third-party providers, which are almost certain to be outside the trust domain of cloud users. To secure data storage and sharing for cloud users, our proposed scheme lets the data owner (also a cloud user) generate her own ABE keys for data encryption and take the full control on key distribution/revocation. The main challenge in this work is how to make the computation load affordable to the data owner and data consumers (both are cloud users). We address this challenge by uniquely combining various computation delegation techniques with ABE and allow both the data owner and data consumers to securely mitigate most computation-intensive tasks to cloud servers, which are envisaged to have unlimited resources.

In WSNs, wireless sensor nodes are often unattendedly deployed in the field and vulnerable to strong attacks such as memory breach. For securing storage and sharing of data on distributed storage sensor nodes while retaining data confidentiality, sensor nodes encrypt their collected data using ABE public keys and store encrypted data on storage nodes. Authorized users are given corresponding decryption keys to read data. The main challenge in this case is that sensor nodes are extremely resource-constrained and can just afford limited computation/communication load. Taking this into account we divide the lifetime of sensor nodes into phases and distribute the computation tasks into each phase. We also revised the original ABE scheme to make the overhead pertained to user revocation minimal for sensor nodes.

Feasibility of the scheme is demonstrated by experiments on real sensor platforms.

To my beloved wife, Jiaai, and my parents

Acknowledgements

A lot of people helped me complete this dissertation, either directly or more subliminal. First of all, I would like to express my sincere thanks to my advisor Dr. Wenjing Lou, who guided me into my current research area, discussed with me about my ideas, proofread my papers, and gave me valuable advises on both my research and my life. She set an example not only as a hardworking and passionate researcher, but also as a responsible and decent person. The second person I would like to thank is Dr. Kui Ren, who was working very closely with me throughout my Ph.D study. He was always there to discuss with me on the technique details and help me think through the difficult problems.

I am very grateful to my dissertation committee members, Dr. Kaveh Pahlavan, Dr. Berk Sunar and Dr. Jie Wang for their valuable time and suggestions, which significantly improved this dissertation. I wish to thank Dr. Fred J. Looft, our department head, who has been very supportive on my research and graduation issues. I also want to thank current and former fellow graduate students in the Wireless Networking and Security Laboratory, Kai Zeng, Zhengyu Yang, Ming Li, Ning Cao and Hanfei Zhao for their friendship and supports. It was with them that made my Ph.D. a rich and enjoyable experience. I am also grateful to my collaborators, Cong Wang and Jin Li for their valuable help.

I am greatly indebted to my parents Guohong Yu and Jiamin Zuo. They are always very understanding and supportive on my choices. They love me more than themselves and have sacrificed so much to support me. I am very indebted to my grandfather, who passed away during my Ph.D study.

Most importantly, I am specially indebted to my wife, Jiaai Zhang, for her support and sacrifice. To stay with me abroad, Jiaai gave up her chances in China, left her family and friends, and accustomed herself to American life although it

was so hard for her. I could not have reached this far without her support and encouragement.

Finally, I would like to acknowledge National Science Foundation for funding my research.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Recent advances in IT have greatly facilitated remote data storage and sharing. New applications such as online social networks and online documents provide very convenient ways for people to store and share various data including personal profile, electronic documents and etc on remote online data servers. Cloud Computing, regarded as the future IT architecture, even promises to provide unlimited and elastic storage resource (and other computing resources) as a service to cloud users in a very cost-effective way [20–23]. Although still at its early stage, Cloud Computing has already drawn great attention, and its benefits have attracted an increasing number of users to outsource their local data centers to remote cloud servers.

Data security is a critical issue for remote data storage. On one hand, disclosure of sensitive information, such as health records, stored on remote data servers has to be strictly protected before users have liberty to use the data services. Fine-grained data access control mechanisms often need to be in place to assure appropriate disclosure of sensitive data among multiple users. On the other hand, in remote data storage users do not physically possess their data. Remote data service providers are almost certain to be outside the users' trust domain, and are not allowed to

learn users' sensitive information stored on their servers. It turns out that users can not rely on remote data servers to enforce access control policies like traditional access control [1,100–102] in which reference monitors should be fully trusted. User-enforced data access control is thus highly desired for remote data storage. More generally, such an issue also exists in any untrusted storage, e.g., distributed data storage in Wireless Sensor Networks (WSNs) [68–70], for which storage devices that are either owned by untrustworthy provider(s) or highly vulnerable to memory breach attacks,

This dissertation addresses the issue of securing data sharing on untrusted storage by exploring cryptographic methods to help users enforce data access policies – only encrypted data are stored on storage servers while retaining secret key(s) to the data owner herself; user access is granted by issuing the corresponding data decryption keys. In particular, we study a novel public-key cryptography – Attribute-Based Encryption (ABE), and enhance it toward providing a full-fledged cryptographic basis for a secure data sharing scheme on untrusted storage. Based on ABE, we also present our solutions for securing data sharing in Cloud Computing and wireless sensor networks respectively.

1.1 Motivation

In untrusted storage data servers are not allowed to learn the content of sensitive data, nor can they be relied on to enforce data access policies. To keep data confidential to data servers the data owner encrypts data before upload. User access is granted by possessing the data decryption key(s). When this kind of cryptographic-based access control scheme provide security protection on data, there are also several major challenges pertained to the scheme design. We can summarize the

challenges as follows.

Fine-grained Access Control vs. Scalability Disclosure of sensitive data usually requires fine-grained access control in the sense that different users may have access privileges to different types/sets of data. Traditionally, access policies are enforced by data servers with mechanisms such as ACL-based access control [100], capability-based access control [101], and role-based access control [102]. For untrusted storage, one might think of enforcing the same access policies like ACL with cryptographic methods. However, ACL-based and capability-based access control, when enforced with cryptographic methods, has the scalability issue. Traditional ACL-based access control demands every data object to record the list of authorized users. When ACLs are enforced with cryptographic methods, the complexity for each data object in terms of its ciphertext size and/or the corresponding data encryption operation is linear to the number of users in the system, and thus makes the system less scalable. Capability-based access control, if enforced with cryptographic methods, has the similar scalability issue. In role-based access control [102], access is granted by the user's role(s) and the data objects do not need to keep the authorized user list. Enforcing these access policies with cryptographic methods has to address various attacks such as user collusion, in which users with different roles (i.e., the corresponding decryption keys) attempt to obtain extra access privileges by piecing together their keys (i.e., roles). There are several recent work [11, 14, 29, 30] in the areas of "shared cryptographic file systems" and "access control of outsourced data" addressing the similar issue of data access control with conventional symmetric-key cryptography or public-key cryptography. When these schemes are suitable for conventional file systems, most of them are less suitable for fine-grained data access control in large-scale data centers which may have a large number users and data files.

Attribute-based encryption (ABE) [2, 12, 19], a recently invented one-to-many public-key cryptography, has the potential to enforce the fine-grained access policies for large-scale systems. In ABE data are associated with attributes. Access policies, defined on attributes, are enforced within the encryption procedure. Different from traditional broadcast encryption [34, 46, 50], ABE offers the ability to encrypt data without exact knowledge of the receiver set. In this sense the concept of ABE is closely related to Role-Based/Attribute-Based Access Control and suitable for large-scale applications. Existing constructions of ABE [2, 12, 19] focus on providing the basic functionalities such as data encryption/decryption and collusion resistance. Before ABE can be applied in practical systems there are still several challenge security issues to be addressed as described in following sections.

User Dynamics In practical application scenarios, users may join or leave the system. An effective and efficient user management mechanism should be in place to deal with user access privilege grant and revocation. In particular, user key (and hence access privilege) revocation is always a challenge issue in cryptography.

In attribute-based encryption (ABE), user secret key revocation is also a challenge issue. Existing solutions [2, 18] suggest associating expiration time attributes to user secret keys. When this type of solutions are able to revoke user secret keys at the designated time, they are not able to revoke users in the timely fashion. [4] proposed an efficient revocation scheme for IBE, which is also applicable to ABE. However, it is more suitable for realtime communication than data/file storage since it does not address how to prevent already-existed data files from being accessed by the revoked user(s).

User Accountability For cryptographic-based data access control, user access is enabled by possessing the corresponding data decryption key(s). This opens the door for an authorized but malicious user to "share" her secret key with unautho-

rized users. More seriously, in copyright-sensitive applications pirates may take this advantage to make profits by selling their secret keys (and hence access privileges) to others. These kind of attacks are extremely harmful for copyright-sensitive applications since it very easy for key abusers to duplicate and distribute data decryption keys to others by ways such as email. As the cost of doing this is extremely low, it is more destructive than directly distributing the data itself. Complete prevention this kind of attacks is usually believed to be very hard. Conventional practice against these attacks is to provide a way for the data owner to trace any suspicious pirate device and collect evidences of key abuse by disclosing the illegal key distributor's identity. Then the data owner can sue the illegal key distributors by presenting these evidences to law authorities. In doing so, it requires that the user decryption key is somehow correlated to her identity. In conventional broadcast encryption, the issue of key abuse is addressed by using a technique called traitor tracing which has been well studied by previous works [36–39].

In ABE [2, 12, 19], a user secret key is defined over attributes and does not have the one-to-one correspondence with any particular user. To defend against key abuse attacks, we can play the same trick in ABE as traitor tracing at a high level view. However, underlying techniques adopted by existing traitor tracing systems can not be directly applied to ABE because receivers are represented individually in conventional broadcast encryption while not in ABE. A novel solution is needed for defending against key abuse attacks for ABE. In doing so, the main challenge is how to efficiently conduct tracing activities without being detected by the suspected users (pirate devices).

Privacy Preservation As the data storage servers can not be trusted, it is desirable to disclose as less user privacy information as possible to servers besides data confidentiality. In particular, the data owner would like to keep her access

policy information confidential to servers and users may have concerns on disclosing their access privilege information to servers. In existing constructions of ABE [2,12, 19], either the access policy or attributes should be attached in plaintext to the data cipheretext to facilitate user decryption. For privacy preservation it is necessary to provide a new construction for ABE to hide the access policy or attributes.

Aside from the above general challenge issues, there are also many other application-specific challenges. Efficiency is one of them and different applications would have different requirements on it. Current construction of ABE introduces expensive operations such as bilinear pairings on encryptor and/or decryptor, which are not necessarily affordable to the parties. It is desirable either to look for efficient constructions for ABE, or to combine ABE with various computation delegation techniques to offload the computation-intensive operations to more powerful devices.

1.2 Contributions

This dissertation makes the several major contributions as follows.

1.2.1 Security enhancements to ABE

ABE is a newly invented PKC primitive that is promising in providing cryptographic-based fine-grained data access control for untrusted storage. Before ABE can be securely applied in practical systems, there are several important security issues to be addressed as listed in section 1.1. This dissertation addresses these issues and proposes several critical security enhancements to ABE.

- **User Revocation** User revocation is a challenge issue in ABE as attributes are shared among unlimited number of users. Revocation of one user may involve key update for other non-revoked users and/or re-encryption of data

files on the data servers. To facilitate user revocation on untrusted storage, this dissertation proposes a novel scheme in which the data owner is able to revoke any user in the timely fashion. The proposed scheme makes it possible for the data owner to securely offload most computation-intensive tasks pertained to user revocation to data servers which are envisaged to be powerful. It achieves this goal by uniquely combining the proxy re-encryption technique [3] with ABE. Security of the proposed scheme is formulated and proved under standard cryptography models.

- **Key Abuse Resistance** In order to defend against key abuse attacks in ABE and hence provide user accountability, this dissertation enhances an existing construction of ABE [1] and proposes a tracing mechanism that helps the data owner identify the key abuser(s). In practical systems, it would be difficult for the data owner to obtain a copy of the pirate's decryption key and check its validity. This is because the data owner may not be able to get physical access to the pirate's key storage device or the pirate may have randomized the key storage memory. To address this issue, we propose a black-box tracing mechanism, i.e., tracing the pirate device only by observing its outputs on some inputs. Such a solution also enables the data owner to remotely trace suspicious users by tricking them into decrypting tracing ciphertexts and thus makes the tracing process very convenient. Formal security proof and performance analysis are both provided for this scheme. This work provides the same security extension to ABE as that traitor tracing techniques do to conventional broadcast encryption [34, 46, 50]. To the best of our knowledge, this work is among the first that addresses the issue of user accountability in ABE.

[1] More precisely, we enhance one branch of ABE – Key-Policy Attribute-Based Encryption (KP-ABE). But the same technique is also applicable to Ciphertext-Policy Attribute-Based Encryption (CP-ABE), another branch of ABE.

- **Encryption with Hidden Policy** In current CP-ABE constructions [2, 9], the access policy should be attached in plaintext to the data ciphertext in order to facilitate user decryption. This plaintext discloses the data owner's access policy and/or the users' access privilege information, and may cause privacy concerns. In order to provide better privacy protection, we propose two novel CP-ABE constructions under different security models. These solutions hide the access policy information not only from the data servers but also from users. To the best of our knowledge, this work is among the first that addresses CP-ABE with hidden policy.

1.2.2 Secure Data Sharing Schemes for Cloud Computing and WSNs

- **Secure Data Sharing in Cloud Computing** Cloud Computing is a promising next-generation IT architecture which provides elastic and unlimited resources, including storage, as services to cloud users. In Cloud Computing cloud users and cloud service providers are almost certain to be from different trust domains. A secure user-enforced data access control mechanism must be provided before cloud users have the liberty to outsource sensitive data to the cloud for storage. In this dissertation, we propose a cryptographic-based data access control mechanism with ABE and enable the data owner to take fully control over data access. Compared to previous work [11,14,29,30], our scheme provides better scalability when providing fine-grained data access control because the complexity of most system operations in our scheme is linear to the number of attributes rather than the number of users/data files.

  In Cloud Computing, cloud servers are very powerful but cloud users could

be resource-constrained devices such as mobile phones. To reduce the computation load for cloud users, we combine various computation delegation techniques with ABE and securely offload computation-intensive tasks to powerful cloud servers. For example, we integrate the technique of proxy re-encryption into ABE and securely mitigate the laborious user revocation task from the data owner to cloud servers. Using another computation delegation technique, we reduce the computation load for data consumers to constant complexity and make it affordable to user devices such as mobile phones. The proposed scheme also significantly saves the computation load for cloud servers by exploiting the technique of lazy re-encryption [14]. Both performance analysis and security proof are provided.

- **Access Control for Distributed Data Storage in WSNs** In WSNs, storing data at local sensor nodes or at designated in-network nodes would greatly save the network-wide communication load and brings forth a lot of benefits such as energy-efficiency and ease of distributed data retrieval. However, unattended wireless sensor nodes are easily subject to strong attacks such as physical compromise and can not be trusted by the owner of the WSN in terms of data security. A secure data storage and retrieval scheme is required for distributed data storage in WSNs. Our proposed solution addresses this issue and provides a cryptographic-based access control mechanism – encrypting data on sensor nodes with ABE public keys and distributing decryption keys to authorized sensor users. To make the expensive ABE encryption operation affordable to resource-constrained sensor nodes, we divide the lifetime of sensor nodes into phases and then distribute the underlying mathematical operations in ABE over these phases. To minimize the communication and computation load on sensor nodes in case of user revocation, we revise an existing ABE

scheme and makes the user revocation complexity on sensor nodes constant. Formal security proof and experimental results shows that our proposed solution is provably secure and affordable to contemporary sensor nodes. To the best of our knowledge, the only existing work prior to ours that addresses the issue of secure distributed data storage and retrieval in WSNs is [83]. However, a recent work [84] shows that there is a severe security weakness with [83]. Our work is de facto the first that provides a secure mechanism for distributed fine-grained data access control in WSNs.

1.3 Roadmap

The organization of this dissertation is as follows.

Chapter 2 presents technical preliminaries pertained to this dissertation. In Section 2.1 we introduces Attribute-Based Encryption. Section 2.2 describes the concepts of miscellaneous techniques to be used in following chapters. In Section 2.3 gives complexity assumptions we will use.

Chapter 3 proposes a user revocation scheme for ABE. In Section 3.1 describe the problem as well as the main idea of our solution. Section 3.2 presents our algorithm definitions and the security model. In Section 3.3, we give a CPA-secure construction for CP-ABE, which is followed by a CCA-secure construction in Section 3.4. Section 3.5 discusses the applicability of our solution to KP-ABE. We summarize this chapter in Section 3.6.

In Chapter 4, we present our solution for key abuse attacks in ABE. Section 4.1 gives the main idea of our solution. Section 4.2 presents models and definitions for this work. In Section 4.3, we describe our construction in detail. Section 4.4 articulates the application scenarios for our proposed scheme. A summarization of

this chapter is given in Section 4.5.

Chapter 5 proposes two privacy enhancement solutions to ABE. Section 5.1 gives our construction under the generic group model. In Section 5.2, we presents our construction under the XDH assumption. We summarize this chapter in Section 5.3.

In Chapter 6, we give our solution for secure data sharing in Cloud Computing. Section 6.1 articulates the models and assumptions to be used in this work. In Section 6.2, we present our scheme in detail. Section 6.3 analyzes our proposed scheme on its security and performance. Section 6.4 summarizes this chapter.

Chapter 7 presents our solution for secure data sharing in Wireless Sensor Networks. Section 7.1 gives our models and assumptions for this work. We present our detailed construction in Section 7.2. Section 7.3 evaluates our proposed scheme on its security and performance. We summarize this chapter in Section 7.4.

Chapter 8 concludes this dissertation and presents several directions for future work.

Chapter 2

Technical Preliminaries

2.1 Attribute-Based Encryption

2.1.1 Definition

Sahai and Waters [19] first introduced the public-key cryptography attribute based encryption (ABE) for cryptographically enforced access control. In ABE both the user secret key and the ciphertext are associated with a set of attributes. A user is able to decrypt the ciphertext if and only if at least a threshold number of attributes overlap between the ciphertext and user secret key. Different from traditional public-key cryptography such as Identity-Based Encryption [5], ABE is intended for one-to-many encryption in which ciphertexts are not necessarily encrypted to one particular user. In Sahai and Waters ABE scheme, the threshold semantics are not very expressive to be used for designing more general access control system. To enable more general access control, Goyal et al. [12] proposed a key-policy attribute-based encryption (KP-ABE) scheme – a variant of ABE. The idea of a KP-ABE scheme is as follows: the ciphertext is associated with a set of attributes and each user secret key is embedded with an access structure which can be any monotonic tree-

access structure. A user is able to decrypt a ciphertext if and only if the ciphertext attributes satisfy the access structure embedded in her secret key. In the same work, Goyal et al. introduced the concept of another variant of ABE – ciphertext-policy attribute-based encryption (CP-ABE). CP-ABE works in the reverse way of KP-ABE in the sense that in CP-ABE the ciphertext is associated with an access structure and each user secret key is embedded with a set of attributes. Formally, KP-ABE and CP-ABE can be defined as follows.

Key-Policy Attribute-Based Encryption A KP-ABE scheme consists of the following four algorithms.

Setup This algorithm takes as input a security parameter κ. and returns the public key PK as well as a system master secret key MK. PK is used by message senders for encryption. MK is used to generate user secret keys and is known only to the authority.

Encryption This algorithm takes a message M, the public key PK, and a set of attributes γ as input. It outputs the ciphertext E.

Key Generation This algorithm takes as input an access structure T and the master secret key MK. It outputs a secret key SK that enables the user to decrypt a message encrypted under a set of attributes γ if and only if γ matches T.

Decryption It takes as input the user's secret key SK for access structure T and the ciphertext E, which was encrypted under the attribute set γ. This algorithm outputs the message M if and only if the attribute set γ satisfies the user's access structure T.

Ciphertext-Policy Attribute-Based Encryption A CP-ABE scheme also consists of four algorithms:

Setup This algorithm takes as input a security parameter κ. and returns the public key PK as well as a system master secret key MK. PK is used by message

senders for encryption. MK is used to generate user secret keys and is known only to the authority.

Encrypt This algorithm takes as input the public parameter PK, a message M, and an access structure T. It outputs the cihphertext CT.

KeyGen This algorithm takes as input a set of attributes γ associated with the user and the master secret key MK. It outputs a secret key SK that enables the user to decrypt a message encrypted under an access structure T if and only if γ matches T.

Decrypt This algorithm takes as input the ciphertext CT and a secret key SK for an attributes set γ. It returns the message M if and only if γ satisfies the access structure associated with the ciphertext CT.

In ABE, including KP-ABE and CP-ABE, the authority runs the algorithm *Setup* and *Key Generation* to generate system MK, PK, and user secret keys. Any user knowing the system public key PK is able to encrypt data by calling the algorithm *Encryption*. Only authorized users (i.e., users with intended access structures) are able to decrypt by calling the algorithm *Decryption*. In this dissertation, we just consider the case of one-writer-and-multiple-reader in untrusted storage for brevity. The only writer is the data owner, who also acts as the authority and is in charge of key generation. This means that the data owner takes the role of both the authority and the encryptor. In the following part of this dissertation, we will alternative call this party by "authority" or "data owner". The decryptor will be called as "data consumer", or just "user" for brevity.

2.1.2 Overview

Attribute-Based Encryption (ABE) was first proposed by Sahai and Waters [19] with the name of Fuzzy Identity-Based Encryption, with the original goal of providing an error-tolerant identity-based encryption [5] scheme that uses biometric identities. In [18], Pirretti et al. proposed an efficient construction of ABE under the Random Oracle model and demonstrated its application in large-scale systems. Goyal et al. enhanced the original ABE scheme by embedding a monotone access structure into user secret key. The scheme proposed by Goyal et al. is called Key-Policy Attribute-Based Encryption (KP-ABE) [12], a variant of ABE. In the same work, Goyal et al. also proposed the concept of Ciphertext-Policy Attribute-Based Encryption (CP-ABE) without presenting a concrete construction. CP-ABE is viewed as another variant of ABE in which ciphertexts are associated with an access structure. Both KP-ABE and CP-ABE are able to enforce general access policies that can be described by a monotone access structure. In [103], Ostrovsky et al. proposed an enhanced KP-ABE scheme which supports non-monotone access structures. Chase [104] enhanced Sahai-Waters ABE scheme and Goyal et al. KP-ABE scheme by supporting multiple authority. Further enhancements to multi-authority ABE can be found in [105, 106]. Bethencourt et al. [2] proposed the first CP-ABE construction with security under the Generic Group model. In [9], Cheung et al. presented a CCA-secure CP-ABE construction under the Decisional Bilinear Diffie-Hellman (DBDH) assumption (cf. Section 2.3). In [9], the CCA-secure scheme just supports AND gates in the access structure. Towards proposing a provably secure CP-ABE scheme supporting general access structure, Goyal et al. [108] proposed a CP-ABE construction with an exponential complexity which can just be viewed as theoretic feasibility. For the same goal, Waters [107] proposed another CP-ABE scheme under various security assumptions. Aside from providing

basic functionalities for ABE, there are also many works proposed to provide better security/privacy protection for ABE. These works include CP-ABE with hidden policy [13,17,40,111], ABE with user accountability [6,15,109], ABE with attribute hierarchy [110] and etc.

2.2 Miscellaneous Techniques

Bilinear Maps In this dissertation, some facts about groups with efficiently computable bilinear maps will be used.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$. A bilinear map is is an injective function $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ with the following properties:

1. *Bilinearity*: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. *Non-degeneracy*: $e(g, g) \neq 1$.
3. *Computability*: There is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}_0$.

Proxy Re-Encryption Proxy Re-Encryption (PRE) [3] is a cryptographic primitive in which a semi-trusted proxy is able to convert a ciphertext encrypted under Alice's public key into another ciphertext that can be opened by Bob's private key without seeing the underlying plaintext. More formally, a PRE scheme allows the proxy, given the proxy re-encryption key $rk_{a \leftrightarrow b}$, to translate ciphertexts under public key pk_a into ciphertexts under public key pk_b and vise versa.

2.3 Complexity Assumptions

This dissertation will use the following complexity assumptions.

Decisional Bilinear Diffie-Hellman (DBDH) Assumption Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_0$. The DBDH assumption [5] states that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuples $(A = g^a, B = g^b, C = g^c, e(g,g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g,g)^z)$ with non-negligible advantage.

The Decision Linear (D-Linear) Assumption Let $z_1, z_2, z_3, z_4, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}_0$. The D-Linear assumption [44] states that that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^{z_3+z_4})$ from the tuple $(g, g^{z_1}, g^{z_2}, g^{z_1 z_3}, g^{z_2 z_4}, g^z)$ with non-negligible advantage.

External Diffie-Hellman (XDH)[2] Let $\mathbb{G}_1, \mathbb{G}_2$ be two distinct groups. The XDH assumption [44, 112] implies the following properties: 1) the Discrete Logarithm problem (DLP) [115], the computational Diffie-Hellman problem (CDH) [117], and the Computational Co-Diffie-Hellman Assumption (co-CDH) [118] are all intractable in $\mathbb{G}_1$ and $\mathbb{G}_2$; 2) an efficiently computable bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ exists; 3) the decisional Diffie-Hellman problem (DDH) [116] is intractable in $\mathbb{G}_1$.

[2] In practice, it is believed that the XDH assumption may hold in certain subgroups of MNT elliptic curves [113, 114].

Chapter 3

User Revocation for Attribute-Based Encryption

In attribute based systems, user revocation is a challenge issue because each attribute is conceivably shared by multiple users. Revocation of any single user would affect others who share the same attributes. This chapter focuses on this important problem. Instead of addressing the issue in general settings, we particularly focus on practical application scenarios such as data storage and sharing, as shown by Fig.3, in which proxy servers are always available for providing various types of data services. Similar to previous work [11], these servers are assumed to be curious-but-honest instead of being totally untrusted. That is, they will honestly execute the tasks assigned by legitimate parties in the system. However, they have the incentive to learn the contents of encrypted data as much as possible. Based on this assumption, our solution uniquely integrates the proxy re-encryption technique [3] with ABE, and enables the authority to delegate most laborious tasks of user revocation to proxy servers without leaking any confidential information to them. On each revocation event, the authority just generates several proxy re-encryption keys

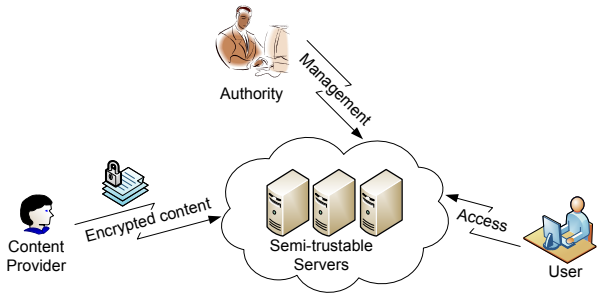


Figure 3.1: An example application scenario of data sharing.

and transmits them to proxy servers. Proxy servers will update secret keys for all users but the one to be revoked. In this way our construction places minimal load on the authority upon each revocation event.

Existing schemes [2, 18] suggest associating expiration time attributes to user secret keys. However, the expiration method just enables user revocation at a pre-arranged time, but is not able to efficiently revoke user attributes on the fly. In our proposed scheme the authority is able to freely revoke any attribute of users at any time. In [4], Boldyreva et al. proposed an efficient revocation scheme for IBE, which is also applicable to KP-ABE [12] and fuzzy IBE [19]. However, it is not clear whether or not the proposed scheme is applicable to CP-ABE. Moreover, this solution just addresses the case of realtime communication but not the case of data storage. Our technique can be used for both CP-ABE and KP-ABE and particularly designed for data storage. For brevity this chapter will mainly focus on presenting our construction for CP-ABE which is followed by a brief discussion on its applicability to KP-ABE.

3.1 Problem Description and Main Idea

We take the following imaginary example to demonstrate our problem: In the WPI campus data system, which might be outsourced to Amazon S3 [21] but remotely

managed by a technician (i.e., the authority) in WPI, each student is associated with a set of attributes such as (WPI, department, year of class, enrolled courses, club memberships, ...). Data files in the system are encrypted with access structures like "WPI $\wedge$ (department: ECE) $\wedge$ (year of class: 2010) $\wedge$ (club memberships: football $\vee$ baseball)" which enables WPI students of class 2010 from the ECE department who are registered with the football club or the baseball club to decrypt. When a student quits from a club, the system should disable the corresponding attribute in the student's attribute set. When a student graduates from WPI, the system should disable the "WPI" attribute in the student's attribute set to revoke the user from the system. Such operations should be remotely executed by the authority of the system.

In current CP-ABE constructions [2, 9], a master key component is defined for each attribute in the system. With these master key components, the system defines the public key and user secret key components each of which corresponds to one of the user's attributes. To disable attributes in a user's attribute set, it is necessary for the authority to redefine the corresponding master key and public key components for the corresponding attributes. Apparently, user secret keys should be updated accordingly for data access. New data files will be encrypted with the new public key, and existing data files should be re-encrypted to prevent revoked users from decrypting using their old version keys. However, these operations could represent a huge amount of computation and are not affordable to the authority if all executed by herself. Moreover, the authority should always stay online to provide key update services for users. To relieve the burden for the authority, we propose to mitigate the most computation-intensive tasks to powerful data servers (e.g., cloud servers in Amazon S3) as follows. On each user/attribute revocation event, the authority redefines the corresponding keys for the attributes and generates proxy

re-key's for the updated master key components, with which the proxy servers are able to securely update user secret keys to the latest version on behalf of the data authority without obtaining the users' decryption capabilities. To revoke a user or his attributes, the proxy servers update user secret keys for all the non-revoke users but refuse to update for the user to be revoked[3]. With the proxy re-key's the proxy servers are also able to to re-encrypt existing ciphertexts stored on them[4] to the latest version without learning the data contents. In this way the workload placed on the authority is minimal and the authority can go off-line after having submitted the proxy re-key's to the proxy servers.

3.2 Definitions and Models

3.2.1 Algorithm Definition

Our proposed scheme is composed of seven algorithms: $Setup$, Enc, $KeyGen$, $ReKeyGen$, $ReEnc$, $ReKey$, and Dec. $Setup$, $KeyGen$, and $ReKeyGen$ are performed by the authority while $ReEnc$ and $ReKey$ are executed by proxy servers. Enc and Dec are called by encryptors and decryptors respectively. The functionalities of $Setup$, Enc, $KeyGen$, and Dec are the same as previous CP-ABE schemes [2, 9]. $ReKeyGen$ is defined for the authority to generate proxy re-key's. $ReEnc$ and $ReKey$ will be used by the proxy servers to re-encrypt data files and update user secret keys respectively. In our scheme we also define a system wide version information ver indicating the evolution of the system master key as follows: initially it is set to one; whenever an attribute revocation event occurs and the sys-

[3]A user secret key is updated when the user accesses proxy servers. Aggregate update for successive events is possible when a user has not accessed the system for a long time.

[4]In practice, this can be done efficiently using the technique of lazy re-encryption [14] as we will discuss later.

tem master key is redefined, it increases by one. The system public key, ciphertexts, user secret keys, and proxy re-key's are all tagged with the version information indicating which version of system master key they comply with. The seven algorithms are defined as following:

Setup(1^λ) It takes as input the security parameter 1^λ and outputs the system master key MK and public parameters PK. ver is initialized as 1.

Enc(M, AS, PK) It takes as input a message M, an access structure AS, and current public parameters PK, and outputs a ciphertext CT.

KeyGen(MK, S) It takes as input current system master key MK and a set of attributes S that describes the key. It outputs a user secret key SK in the form of $(ver,\ S,\ D,\ \bar{D} = \{D_i, F_i\}_{i \in S})$.

ReKeyGen(γ, MK) It takes as input an attribute set γ that includes attributes for update, and current master key MK. It outputs the new master key MK', the new public key PK' (computation of PK' can be delegated to proxy servers), and a set of proxy re-key's rk for all the attributes in the attribute universe U. ver is increased by 1. Note that, for attributes in set $U - \gamma$, their proxy re-key's are set as 1 in rk.

ReEnc(CT, rk, β) It takes as input a ciphertext CT, the set of proxy re-key's rk having the same version with CT, a set of attributes β which includes all the attributes in CT's access structure with proxy re-key not being 1 in rk. It outputs a re-encrypted ciphertext CT' with the same access structure as CT.

ReKey$(\bar{D}, rk, \theta)$ It takes as input the component $\bar{D}$ of a user secret key SK, the set of proxy re-key's rk having the same version with SK, and a set of attributes θ which includes all the attributes in SK with proxy re-key not being 1 in rk. It outputs updated user secret key components $\bar{D}'$.

Dec(CT, PK, SK) It takes as input a ciphertext CT, public parameters PK, and the user secret key SK having the same version with CT. It outputs the message M if the attribute set of SK satisfies the ciphertext access structure. Otherwise, it returns $\perp$ with an overwhelming probability.

3.2.2 Security Definition

We first define the correctness of our proposed scheme by the following conditions:

(1) $Dec(Enc(M, AS, PK), PK, SK) = M$, if the attribute set S of SK satisfies AS.

(2) Let $CT' = ReEnc(Enc(M, AS, PK), rk, \beta)$, and $SK' = (ver + 1, S, D, \bar{D}' = ReKey(\bar{D}, rk, \theta))$, where ver is the version number of PK and rk. $Dec(CT', PK', SK') = M$, if $S' = S \backslash (\beta \backslash \theta)$ satisfies AS.

(3) Let $CT'' = ReEnc(CT', rk', \beta')$, and $SK'' = (ver+2, S', D, ReKey(\bar{D}', rk', \theta'))$. If $Dec(CT', PK', SK') = M$ and $S'' = S' \backslash (\beta' \backslash \theta')$ satisfies AS, $Dec(CT'', PK'', SK'') = M$.

(4) Inductively we get the statement for $(CT^{(n)}, PK^{(n)}, SK^{(n)})$ of any n.

CPA security of our proposed scheme under the selective-structure model [9] can be defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

CPA Security Game Let λ be a security parameter. We say that our scheme is secure against chosen plaintext attacks under selective-structure model if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage.

Init The adversary $\mathcal{A}$ chooses the challenge access structure AS^*, a version number ver^*, and $ver^* - 1$ attribute sets $\{\gamma^{(1)}, \gamma^{(2)}, \cdots, \gamma^{(ver^*-1)}\}$, and submits them to the challenger $\mathcal{B}$.

Setup The challenger $\mathcal{B}$ fist runs Setup(1^λ) to obtain MK and PK for version 1. He then runs $ReKeyGen(\gamma_i, MK)$ from $i = 1$ to $ver^* - 1$. Finally, $\mathcal{B}$ gives (PK, $\{rk^{(i)}\}_{2 \le i \le ver^*}$) to $\mathcal{A}$, where $rk^{(i)}$ denotes the proxy re-key set for version i [5]. Note that, $\mathcal{A}$ is able to derive PK for all the versions with $rk^{(i)}$'s.

Phase 1 The adversary $\mathcal{A}$ is allowed to issue polynomial times (in λ) of queries on generation of secret keys of any version within the range of $[1, ver^*]$. The only restrict is that the attribute set that $\mathcal{A}$ submits for each secret key query does not satisfy AS^*.

Challenge The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b by executing $CT^* \leftarrow Enc(M, AS^*, PK)$, where PK is the public parameter for version ver^*. The challenge ciphertext CT^* is passed to the adversary.

Phase 2 Phase 1 is repeated.

Guess The adversary $\mathcal{A}$ outputs his guess b_0 of b.

The adversary $\mathcal{A}$ is advantage in winning this CPA security game is defined as $ADV_{CPA} = Pr[b_0 = b] - \frac{1}{2}$.

Note that, In Phase 1, the adversary is also permitted to issue queries on re-encryption of ciphertexts and on update of secret keys. In our security game, however, the adversary has been given all the proxy re-key's. This means that he is able to answer the two queries by himself. For this sake, we do not include the two corresponding oracles in Phase 1. In fact, the adversary $\mathcal{A}$ has at least the same capability as proxy servers who passively collect secret keys of unauthorized users. Since we assume proxy servers are honest, we do not consider active attacks from proxy servers by colluding with revoked authorized users.

[5] In this chapter, the superscript (i) means that the component is of version i. When there is no confusion, we always remove the superscript for brevity. For example, we may just use rk or γ.

Definition 3.2.1 *(CPA SECURITY) We say that our scheme is CPA secure if* ADV_{CPA} *is negligible (in* λ*) for any polynomial time adversary.*

3.3 Our Construction

3.3.1 Overview

The basic idea of our construction is to combine the proxy re-encryption technique with CP-ABE. Instead of building a new CP-ABE scheme from scratch, we intend to enhance an existing construction by extending it with abilities of proxy update of secret key and proxy re-encryption of ciphertext. Our construction is partially based on but not limited to Cheung et al construction of CP-ABE [9].

Attribute and Access Structure In our construction, attributes are represented by their index values and the attribute universe is $U = \{1, 2, \cdots, n\}$ for a certain natural number n. Each attribute would have three occurrences: *positive*, *negative*, and "*don't care*". We just consider access structures consisting of a single AND gate, i.e., the gate $\bigwedge_{\tilde{i} \in I} \tilde{i}$, where I denotes the set of attributes of interest and $\tilde{i}$ is the literal of an attribute i, which can be positive (denoted by $+i$) or negative (denoted by $-i$). If an attribute does not appear in the AND gate, its occurrence is "don't care".

3.3.2 The Detailed Construction

As is defined in section 3.2.1, there are seven algorithms in our construction: *Setup*, *Enc*, $KeyGen$, $ReKeyGen$, $ReEnc$, $ReKey$, and *Dec*. Now we present the construction for each of them as follows.

Setup(1^λ) First choose a bilinear group $\mathbb{G}_0$ of prime order p with a generator g, and a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$. Next, select random numbers $y, t_1, \cdots, t_{3n} \in \mathbb{Z}_p$. Then, generate the public parameter as: $PK = (e, g, Y, T_1, \cdots, T_{3n})$, where $Y = e(g, g)^y$ and $T_i = g^{t_i}$ for $1 \leq i \leq 3n$. T_i, T_{n+i}, and T_{2n+i} are for the three occurrences of i, i.e., positive, negative, and "don't care", respectively. The system master key MK is: $MK = (y, t_1, \cdots, t_{3n})$. Finally, initialize version number as $ver = 1$ and publish (ver, PK). (ver, MK) is witheld by the authority.

Enc(M, AS, PK) Note that AS is a single AND gate of form $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$, and assume $M \in \mathbb{G}_1$. The algorithm chooses a random number $s \in \mathbb{Z}_p$ and outputs the ciphertext CT as: $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U})$, where ver is current version number, $\tilde{C} = MY^s$, $\hat{C} = g^s$. For each $i \in I$, C_i is T_i^s if $\tilde{i} = +i$; or T_{n+i}^s if $\tilde{i} = -i$. If $i \in U \backslash I$, $C_i = T_{2n+i}^s$.

KeyGen(MK, S) First choose a random number $r_i \in \mathbb{Z}_p$ for each $i \in U$. Let $r = \sum_{i=1}^{n} r_i$. User secret key is defined as $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i \in U})$, where ver is current version number, $D = g^{y-r}$. For each $i \in U$, $F_i = g^{\frac{r_i}{t_{2n+i}}}$, and $D_i = g^{\frac{r_i}{t_i}}$ if $i \in S$, or $D_i = g^{\frac{r_i}{t_{n+i}}}$ otherwise. Note that $i \notin S$ means negative occurrence of attribute i in S.

ReKeyGen(γ, MK) Each item $i \in \gamma$ is defined to be within the range of [1,2n]. Value less or equal to n means positive occurrence of the attribute, while value greater than n represents the negative occurrence of attribute $i - n$. The proxy re-key is computed as follows. For each $i \in \gamma$, randomly choose $t_i' \in \mathbb{Z}_p$ and compute $rk_i = \frac{t_i'}{t_i}$. For each $i \in \{1, \cdots, 2n\} \backslash \gamma$, $rk_i = 1$. Output proxy re-key as $rk = (ver, \{rk_i\}_{1 \leq i \leq 2n})$ where ver is current version number. Increase the system version number ver by 1 when everything is done.

ReEnc(CT, rk, β) Denote the access structure of CT as $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$. Similar to γ, each item in β is also defined to be within the range of [1,2n]. This algorithm

directly outputs CT if CT and rk contain different version numbers. Otherwise, re-encrypt CT as follows. For each $i \in \beta$, $C'_i = C_i^{rk_i}$ if $1 \leq i \leq n$, or $C'_{i-n} = (C_{i-n})^{rk_i}$ if $n < i \leq 2n$. For each $i \in U$, $C'_i = C_i$ if $i \notin \beta$ and $i + n \notin \beta$, or $i \notin I$. Ciphertext is output as follows: $CT' = (ver + 1, AS, \tilde{C}, \hat{C}, \{C'_i\}_{i \in U})$, where ver is the version number in CT.

ReKey$(\bar{D}, rk, \theta)$ Each item in θ is defined to be within the range of [1,2n]. This algorithm returns with $\bar{D}$ immediately if $\bar{D}$ and rk contain different version numbers. Otherwise, update $\bar{D}$ as follows. For each $i \in \theta$, $D'_i = D_i^{rk_i^{-1}}$ if $1 \leq i \leq n$, or $D'_{i-n} = (D_{i-n})^{rk_i^{-1}}$ if $n < i \leq 2n$. For each $i \in U$, $D'_i = D_i$ if $i \notin \theta$ and $i + n \notin \theta$. It outputs as follows: $\bar{D}' = \{D'_i, F_i\}_{i \in U}$. ver in the corresponding user secret key SK is increased by 1.

Dec(CT, PK, SK) If any two of CT, PK, and SK have different version numbers, return $\perp$. Otherwise, continue to decrypt as follows. Suppose $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i \in U})$, $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i \in U})$, and denote AS by $AS = \bigwedge_{\tilde{i} \in I} \tilde{i}$. For each $\tilde{i} \in I$, if $\tilde{i} = +i$ and $i \in S$,

$$e(C_i, D_i) = e(g^{t_i s}, g^{\frac{r_i}{t_i}}) = e(g, g)^{r_i s}.$$

if $\tilde{i} = -i$ and $i \notin S$,

$$e(C_i, D_i) = e(g^{t_{n+i} s}, g^{\frac{r_i}{t_{n+i}}}) = e(g, g)^{r_i s}.$$

For each $\tilde{i} \notin I$,

$$e(C_i, D_i) = e(g^{t_{2n+i} s}, g^{\frac{r_i}{t_{2n+i}}}) = e(g, g)^{r_i s}.$$

Ciphertext is decrypted as follows:

$$M = \tilde{C} / (e(\hat{C}, D) \prod_{i=1}^{n} e(g, g)^{r_i s}).$$

Its correctness can be verified easily.

3.3.3 CPA Security Proof

Now we prove the CPA security of our scheme. We show the CPA security of our scheme by a theorem.

Theorem 3.3.1 *If a PPT algorithm (the adversary $\mathcal{A}$) wins our CPA security game with non-negligible advantage ADV_{CPA}, we can use this algorithm to construct another PPT algorithm $\mathcal{B}$ to solve the DBDH problem with advantage $\frac{1}{2}ADV_{CPA}$.*

Proof: In the DBDH game, the challenger chooses random numbers a, b, c from $\mathbb{Z}_p$ and flips a fair coin μ. If $\mu = 0$, set $z = abc$; If $\mu = 1$, set z as a random value in $\mathbb{Z}_p$. $\mathcal{B}$ is given (A, B, C, Z)=$(g^a, g^b, g^c, e(g, g)^z)$ and asked to output μ. To answer this challenge, $\mathcal{B}$ then simulates our CPA security game as follows.

Init The adversary $\mathcal{A}$ chooses the challenge access structure $AS^* = \bigwedge_{\tilde{i} \in I} \tilde{i}$, a version number ver^*, and $ver^* - 1$ attribute sets $\{\gamma^{(1)}, \gamma^{(2)}, \cdots, \gamma^{(ver^*-1)}\}$, and submits them to the challenger.

Setup The challenger $\mathcal{B}$ first generates the public key of version 1 for $\mathcal{A}$ as follows. Y is defined as $e(A, B) = e(g, g)^{ab}$. For each $i \in U$, $\mathcal{B}$ randomly chooses δ_i, ζ_i, and η_i from $\mathbb{Z}_p$. It outputs public parameters as follows.

For $\tilde{i} \in I$, $T_i = g^{\delta_i}$, $T_{n+i} = B^{\zeta_i}$, and $T_{2n+i} = B^{\eta_i}$, if $\tilde{i} = +i$;

if $\tilde{i} = -i$, $T_i = B^{\delta_i}$, $T_{n+i} = g^{\zeta_i}$, and $T_{2n+i} = B^{\eta_i}$;

For $\tilde{i} \notin I$, $T_i = B^{\delta_i}$, $T_{n+i} = B^{\zeta_i}$, and $T_{2n+i} = g^{\eta_i}$.

Then, $\mathcal{B}$ generates ver^* versions and answers $ver^* - 1$ proxy re-key generation requests. Specifically, for each attribute set $\gamma^{(k)}$, $1 \leq k \leq ver^* - 1$, generate a PK for that version as follows:

for each element $j \in \gamma^{(k)}$, where $1 \leq j \leq 2n$, randomly choose $rk_j^{(k)}$ from $\mathbb{Z}_p$. if $1 \leq j \leq n$,

$$T_j^{(k+1)} = (T_j^{(k)})^{rk_j^{(k)}}, T_{n+j}^{(k+1)} = T_{n+j}^{(k)}, T_{2n+j}^{(k+1)} = T_{2n+j}^{(k)},$$

if $n < j \leq 2n$,

$T_{j-n}^{(k+1)} = T_{j-n}^{(k)}, T_j^{(k+1)} = (T_j^{(k)})^{rk_j^{(k)}}, T_{n+j}^{(k+1)} = T_{n+j}^{(k)}$,

where superscripts (k) and $(k+1)$ denote the version number of each attribute set, re-key, and public key parameter.

For each element $1 \leq j \leq 2n$, if $j \notin \gamma^{(k)}$, set $rk_j^{(k)} = 1$, and calculate public key components in the same way as above. Finally, $\mathcal{B}$ returns $rk^{(k)} = (k, rk_1^{(k)}, rk_2^{(k)}, \cdots, rk_{2n}^{(k)})$ to $\mathcal{A}$.

Phase 1 Without loss of generality, we assume the adversary $\mathcal{A}$ submits secret key query on a set $S \subseteq U$ for version k, $1 \leq k \leq ver^*$. Since S does not satisfy the challenge access structure AS^*, we know there is a witness attribute $i \in I$ that either $i \in S$ and $\tilde{i} = -i$, or $i \notin S$ and $\tilde{i} = +i$. Without loss of generality, we assume $i \notin S$ and $\tilde{i} = +i$. $\mathcal{B}$ first chooses a random number $r'_j \in \mathbb{Z}_p$ for each $j \in U$. Then, it sets $r_j = r'_j \cdot b$ for every $j \neq i$ (non-witness attribute), and $r_j = ab + r'_j \cdot b$. Finally, it calculates $r = \Sigma_{j \in U} r_j = ab + \Sigma_{j \in U} r'_j \cdot b$. Secret key components are then returned as follows:

$D = \Pi_{j=1}^{n} B^{-r'_j} = g^{-\Sigma_{j=1}^{n} r'_j \cdot b} = g^{ab-r}$.

Consider that for any $j \in U$,

$T_j^{(k)} = (T_j^{(1)})^{rk_j^{(2)} \cdot rk_j^{(3)} \cdots rk_j^{(k)}} = T_j^{\Pi_{i=2}^{k} rk_j^{(i)}}$, and

$T_{n+j}^{(k)} = (T_{n+j}^{(1)})^{rk_{n+j}^{(2)} \cdot rk_{n+j}^{(3)} \cdots rk_{n+j}^{(k)}} = (T_{n+j})^{\Pi_{i=2}^{k} rk_{n+j}^{(i)}}$,

we denote $R_j^{(k)} = \Pi_{i=2}^{k} rk_j^{(i)}$ and $R_{n+j}^{(k)} = \Pi_{i=2}^{k} rk_{n+j}^{(i)}$. For each $j \in U$ and $j \neq i$, D_j is calculated as follows.

`Case 1.` $j \in S$.

1) $D_j = B^{\frac{r'_j}{\delta_j \cdot R_j^{(k)}}} = g^{\frac{r_j}{\delta_j \cdot R_j^{(k)}}}$, if $j \in I$ and $\tilde{j} = +j$;

2) $D_j = B^{\frac{r'_j}{\delta_j \cdot R_j^{(k)}}} = g^{\frac{r_j}{\delta_j \cdot R_j^{(k)} \cdot b}}$, if $j \in I$ and $\tilde{j} = -j$, or $j \notin I$;

`Case 2.` $j \notin S$.

1) $D_j = g^{\frac{r'_j}{\zeta_j \cdot R^{(k)}_{n+j}}} = g^{\frac{r_j}{\zeta_j \cdot R^{(k)}_{n+j} \cdot b}}$, if $j \in I$ and $\tilde{j} = j$, or $j \notin I$;

2) $D_j = B^{\frac{r'_j}{\zeta_j \cdot R^{(k)}_{n+j}}} = g^{\frac{r_j}{\zeta_j \cdot R^{(k)}_{n+j}}}$, if $j \in I$ and $\tilde{j} = -j$.

D_i is calculated as:

$$D_i = A^{\frac{1}{\zeta_i \cdot R^{(k)}_i}} \cdot g^{\frac{r'_i}{\zeta_i \cdot R^{(k)}_i}} = g^{\frac{ab + r'_i \cdot b}{\zeta_i \cdot R^k_i \cdot b}} = g^{\frac{r_i}{\zeta_i \cdot R^{(k)}_i \cdot b}}.$$

For each attribute $j \in U$, F_j is calculated as follows. If $j \neq i$, then

1) $F_j = g^{\frac{r'_j}{\eta_j}} = g^{\frac{r_j}{\eta_j \cdot b}}$, if $j \in I$;

2) $F_j = B^{\frac{r'_j}{\eta_j}} = g^{\frac{r_j}{\eta_j}}$, if $j \notin I$;

F_i is calculated as follows:

$$F_i = A^{\frac{1}{\eta_i}} \cdot g^{\frac{r'_i}{\eta_i}} = g^{\frac{ab + r'_i \cdot b}{\eta_i \cdot b}} = g^{\frac{r_i}{\eta_i \cdot b}}.$$

Challenge. The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, sets $\tilde{C} = M_b \cdot Z$, and outputs the ciphertext CT^* as follows.

$$CT^* = (ver^*, AS^*, \tilde{C}, C, \{C^{\delta_i \cdot R^{(ver^*)}_i}\}_{i \in I \wedge \tilde{i} = +i}, \{C^{\zeta_i \cdot R^{(ver^*)}_{n+i}}\}_{i \in I \wedge \tilde{i} = -i}, \{C^{\eta_i}\}_{i \notin I}).$$

Phase 2. Phase 1 is repeated.

Guess. $\mathcal{A}$ submits a guess b_0 of b. If $b_0 = b$, $\mathcal{B}$ will output $\mu' = 0$, meaning that (A, B, C, Z) is a valid DBDH-tuple; otherwise, $\mathcal{B}$ outputs $\mu' = 1$, indicating that (A, B, C, Z) is just a random 4-tuple. In the case of $\mu = 1$, the adversary obtains no information about b. We thus have $Pr[b_0 \neq b | \mu = 1] = \frac{1}{2}$. $\mathcal{B}$ just randomly guesses $\mu' = 1$ when $b \neq b_0$, we have $Pr[\mu' = \mu] = \frac{1}{2}$. In the case of $\mu = 0$, the adversary obtains an encryption of m_b, and his advantage is ADV_{CPA} by definition. We thus have $Pr[b = b_0 | \mu = 0] = \frac{1}{2} + ADV_{CPA}$. Since $\mathcal{B}$ guesses $\mu' = 0$ whenever $b = b_0$, we have $Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + ADV_{CPA}$. The overall advantage of $\mathcal{B}$ in the DBDH game is $\frac{1}{2} Pr[\mu' = \mu | \mu = 0] + \frac{1}{2} Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + ADV_{CPA}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2} ADV_{CPA}$.

3.4 CCA Security Construction

We now proceed to discuss the construction of the chosen ciphertext secure scheme. For IBE schemes, a common practice of constructing a CCA secure scheme from a CPA secure one is to generate one-time signature keys (K_v, K_s) and sign the ciphertext with K_s with a strongly existentially unforgeable signature scheme, while K_v is viewed as the message receiver's identity. This technique was proposed by Canetti, Halvei, and Katz [7]. In [9], Cheung and Newport applied the similar technique to CP-ABE and constructed a CCA secure CP-ABE scheme from the CPA secure one. Their construction defines an attribute for each bit in the key space of K_v, each attribute having two occurrences for its binary values. Each user secret key contains two components for the both occurrences of each bit. Thereafter, these attributes are treated similarly as other normal attributes. For encryption, the encryptor chooses a pair (K_v, K_s) and encrypts the message with the attributes for K_v in addition to other normal attributes. The whole ciphertext is then signed with K_s. The ciphertext along with the signature is sent to receiver(s), who will verify the signature before decryption.

In our work, it seems to be a contradiction to construct a CCA secure scheme since we on one hand require the ciphertext to be non-malleable, and on the other hand give the proxy re-key's to proxy servers and allow them to re-encrypt ciphertexts. However, in our scheme ciphertext re-encryption is just limited to updating partial ciphertext components to the latest version. Modification of the underlying message or the access structure is not permitted. In terms of non-malleability, we just need to prevent adversaries from modifying the message or the access structure. Based on this observation, we adopt the same technique as [9] but just sign on partial ciphertext components.

3.4.1 CCA Secure Construction

The seven algorithms in the CCA secure construction are defined as follows, assuming that the signature verification key K_v has w bits. Denote the set $\{1, 2, \cdots, w\}$ as W.

Setup(1^λ) The same as the CPA secure construction except that, here $2w$ extra attributes are defined for K_v. Now the system master key is: $MK = (y, t_1, \cdots, t_{3n}, t_{3n+1}, \cdots, t_{3n+2w})$, and the public parameters are: $PK=(e, g, Y, T_1, \cdots, T_{3n}, T_{3n+1}, \cdots, T_{3n+2w})$. Initialize the system wide version number ver as 1 and publish (ver, PK). (ver, MK) is kept by the authority.

Enc(M, AS, PK) AS is defined to be an AND gate as before. The encryptor first chooses one-time signature key pair (K_v, K_s), and a random number $s \in \mathbb{Z}_p$. M is encrypted as: $(ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i\in U}, \{K_i\}_{i\in W}, K_v)$, where ver is current version number, $\tilde{C} = MY^s$, $\hat{C} = g^s$. For each $i \in I$, $C_i = T_i^s$ if $\tilde{i} = +i$; or $C_i = T_{n+i}^s$ if $\tilde{i} = -i$. If $i \in U \backslash I$, $C_i = T_{2n+i}^s$. For each $i \in W$, $K_i = T_{3n+i}^s$ if the ith bit of K_v is 0, otherwise, $K_i = T_{3n+w+i}^s$. The encryptor then signs on tuple $(AS, \tilde{C}, \hat{C}, \{K_i\}_{i\in W}, K_v)$ with K_s, and obtains a signature δ. Finally, the ciphertext of M is output as $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i\in U}, \{K_i\}_{i\in W}, K_v, \delta)$.

KeyGen(MK, S) First choose a random numbers $r_i \in \mathbb{Z}_p$ for each $i \in U \bigcup W$. Let $r = \sum_{i=1}^{w+n} r_i$. The secret key is defined as $SK = (ver, S, D, \bar{D} = \{D_i, F_i\}_{i\in U}, \hat{D} = \{\hat{D}_{i,0}, \hat{D}_{i,1}\}_{i\in W})$, where D and $\bar{D}$ are the same as the CPA secure construction. $\hat{D}$ is defined as: $\hat{D}_{i,0} = g^{\frac{r_{n+i}}{t_{3n+i}}}$ and $\hat{D}_{i,1} = g^{\frac{r_{n+i}}{t_{3n+w+i}}}$ for each $i \in W$. Note that this definition extends the one defined in section 3.2.1.

ReKeyGen(γ, MK) The same as the CPA secure construction.

ReEnc(CT, rk, β) Let CT be $(ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i\in U}, \{K_i\}_{i\in W}, K_v, \delta)$. The re-encrypted ciphertext is output as $CT' = (ver{+}1, AS, \tilde{C}, \hat{C}, \{C_i'\}_{i\in U}, \{K_i\}_{i\in W}, K_v, \delta)$, where each C_i' is generated in the same way as the CPA secure construction.

ReKey$(\bar{D}, rk, \theta)$ The same as the CPA secure construction.

Dec(CT, PK, SK) The decryptor first verifies the signature δ. On failure, return $\perp$; otherwise, proceed as in the CPA secure construction.

3.4.2 CCA Security Proof

We first give a definition on CCA security of our scheme. Then, we sketch the security proof.

CCA Game Let λ be a security parameter. We say that our scheme is secure against chosen ciphertext attacks under selective-structure model if no PPT adversary $\mathcal{A}$ can win the following game with non-negligible advantage.

Init and **Setup** Same as the CPA security game.

Phase 1 The adversary is allowed to adaptively make polynomial times (in λ) of any combination of secret key and decryption queries.

Query for Secret Key The adversary submits an attribute set S. The challenger returns a secret key SK for S, given that S does not satisfies AS^*.

Query for Decryption The adversary $\mathcal{A}$ submits a ciphertext CT. If CT is not a valid ciphertext, $\mathcal{A}$ loses the game; otherwise, the challenger $\mathcal{B}$ returns the plaintext M.

Challenge Same as the CPA security game.

Phase 2 Same as Phase 1. Similar to [8], ciphertexts submitted for decryption are not allowed to be derivatives of CT^*. A derivative of CT^* is defined as any CT that can be used to derive CT^* by repeatedly executing algorithm $ReEnc$ on proxy re-key's $rk^{(2)}, rk^{(3)}, \cdots, rk^{(ver^*)}$.

Guess The adversary $\mathcal{A}$ outputs his guess b_0 of b.

The adversary $\mathcal{A}$ is advantage in winning this CCA security game is defined as $ADV_{CCA} = Pr[b_0 = b] - \frac{1}{2}$.

Definition 3.4.1 *(CCA SECURITY) We say that our scheme is CCA secure if ADV_{CCA} is negligible (in λ) for any polynomial time adversary.*

CCA security of our scheme can be shown by the following theorem.

Theorem 3.4.2 *If a PPT algorithm (the adversary $\mathcal{A}$) wins our CCA security game with non-negligible advantage ADV_{CCA}, we can use this algorithm to construct another PPT algorithm $\mathcal{B}$ to solve the DBDH problem with advantage $\frac{1}{2}ADV_{CCA}$, assuming that the signature scheme is strongly existentially unforgeable.*

proof: The challenger of the DBDH game generates the tuple (A, B, C, Z) exactly as in the CPA security proof, and then sends it to $\mathcal{B}$. To answer this challenge, first choose a signature key pair (K_v^*, K_s^*) and then simulates our CPA security game as follows.

Init The same as the CPA security proof.

Setup In this phase, $\mathcal{B}$ generates $(Y, T_1, \cdots, T_{3n})$ and $(rk^{(2)}, rk^{(3)}, \cdots, rk^{(ver^*)})$ exactly the same as the CPA security proof. $\mathcal{B}$ generates $(T_{3n+1}, \cdots, T_{3n+2w})$ as follows. For each $i \in W$, select random numbers $\phi_i, \psi_i \in \mathbb{Z}_p$, and set $T_{3n+i} = g^{\phi_i}$ and $T_{3n+w+i} = B^{\psi_i}$ if the ith bit of K_v^* is 0, denoted by $K_{v,i}^* = 0$; otherwise, set $T_{3n+i} = B^{\phi_i}$ and $T_{3n+w+i} = g^{\psi_i}$.

Phase 1. $\mathcal{B}$ answers queries for secret key and for decryption.

`Case 1.` Query for secret key. $\mathcal{B}$ executes in the same way as Phase 1 of the CPA security game. When generating $(D_{j,0}, D_{j,1})$ for each $j \in W$, $\mathcal{B}$ deals in the same way as non-witness attributes in U except that, R_j^k or R_{n+j}^k are no longer needed when computing $D_{j,0}$ and $D_{j,1}$ since $\hat{D}$ part of a user secret key never needs update.

`Case 2.` Query for decryption. $\mathcal{A}$ submits a ciphertext $CT = (ver, AS, \tilde{C}, \hat{C}, \{C_i\}_{i\in U}, \{K_i\}_{i\in W}, K_v, \delta)$ for decryption. $\mathcal{B}$ first verifies the signature δ with K_v. If the signature is not valid, $\mathcal{B}$ terminates the DBDH simulation game without answering the DBDH challenger and start a new game. Otherwise, proceed. In this case, we know $K_v \neq K_v^*$ with overwhelming probability. Otherwise, K_v can be used to successfully verify δ and the signature contained in the challenge ciphertext, which is assumed to happen with negligible probability since the signature scheme is strongly existentially unforgeable. In case of $K_v \neq K_v^*$, we can assume the jth bits of them are different. Without loss of generality, we assume that the bit of K_v^* is 0. Therefore, $K_j = T_{3n+w+j}^s = g^{b\cdot\psi_j\cdot s}$. $\mathcal{B}$ then calculates $e(K_j, A) = e(g,g)^{abs\psi_j} = Y^{s\psi_j}$. Since ψ_j is known to $\mathcal{B}$, he gives $\tilde{C}/(e(K_j, A)^{\frac{1}{\psi_j}})$ to $\mathcal{A}$ as the message M.

Challenge. The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, sets $\tilde{C} = M_b \cdot Z$, and outputs the ciphertext CT^* as follows.

$$CT^* = (ver^*, AS^*, \tilde{C}, C, \{C^{\delta_i \cdot R_i^{(ver^*)}}\}_{i\in I\wedge\tilde{i}=+i}, \{C^{\eta_i}\}_{i\notin I},$$
$$\{C^{\zeta_i \cdot R_{n+i}^{(ver^*)}}\}_{i\in I\wedge\tilde{i}=-i}, \{C^{\phi_i}\}_{i\in W\wedge K_{v,i}^*=0}, \{C^{\psi_i}\}_{i\in W\wedge K_{v,i}^*=1}).$$

Phase 2. Repeat Phase 1. The only restriction is that, ciphertexts submitted for decryption are not allowed to be derivatives of CT^*. $\mathcal{B}$ is able to verify this by running algorithm *ReEnc* on proxy re-key's and CT^*, and compare the results with the ciphertexts he received from $\mathcal{A}$.

Guess. The same as the CPA security proof.

3.5 Applicability to KP-ABE

In KP-ABE, ciphertexts are associated with attributes, while user secret keys are defined with access structures on attributes. If only the ciphertext attributes satisfy

a user's access structure, can he decrypt. When CP-ABE is applicable in Role-Based Access Control like scenarios, KP-ABE is suitable for applications such as pay-per-view TV systems, in which user access privileges are defined over content attributes and could be based on the prices they paid. In these application scenarios, the issue of key revocation also exists. Fig. 2. shows such an example, in which a user currently is allowed to access any series with name "Hero", "Lost", or "Dexter"' provided by channel 4. The system administrator now wants to disable the user's access privilege on series with name "Lost" for some reason (maybe late payment). For this purpose, it is necessary to revoke the corresponding component of the user's secret key.

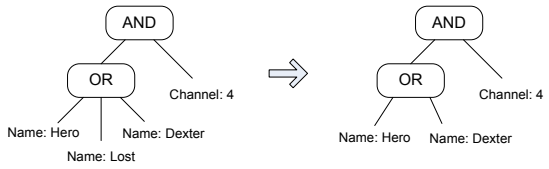


Figure 3.2: An example use of KP-ABE.

Similar to CP-ABE, the basic construction of current KP-ABE scheme [12] also defines a system master key component t_i for each attribute i. The corresponding public key component is defined as $T_i = g^{t_i}$. Encrypting a message with attribute i means including a component T_i^s into the ciphertext, where s is a random number for this ciphertext. In user secret key, the component for attribute i has the form of $g^{\frac{qx(0)}{t_i}}$, where $q_x(\cdot)$ is a polynomial uniquely defined for the user. Therefore, we can revoke a secret key component in the same way as we did for CP-ABE, i.e., the authority redefines the master key component as t_i' and give $\frac{t_i'}{t_i}$ to proxy servers as the proxy re-key. In the same way as our CP-ABE scheme, proxy servers, which are honest by our assumption, will use these proxy re-key's to re-encrypt ciphertexts stored on them and update secret keys for all but the user for revocation. Proof of

the new KP-ABE scheme is similar to that of our CP-ABE scheme.

Large Universe Construction In addition to the basic construction, [12] also provides a KP-ABE construction for large universe cases. One significant advantage of this construction is that the number of public parameter components is constant, no matter how many attributes the system may have. In this construction, however, our technique of generating proxy re-key's for CP-ABE and the basic KP-ABE scheme is not applicable any more. This is because the definitions of public parameter components and user secret key components are no longer in the same format as before. In this construction, it selects $n+1$ random points from $\mathbb{G}_1$ [6] and defines a function T to calculate the public key component for attribute X [7] as:

$$T(X) = g_2^{X^n} \cdot \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(X)},$$

where g_2 is another group element in $\mathbb{G}_1$, $N = \{1, 2, \cdots, n+1\}$, and $\Delta_{i,N}(X)$ is the Lagrange coefficient. Ciphertext component for each attribute i is still in the form of T_i^s as before. Each attribute has two components in user secret key:

$$D_i = g_2^{q_x(0)} \cdot T(i)^{r_i}, \text{ and } R_i = g^{r_i},$$

where $r_i \in \mathbb{Z}_p$ is a random number of attribute i in the user secret key. We refer to [12] for the detailed construction.

In this construction, we can not simply redefine the system master key component t_i to update the attribute i for key revocation as before. Instead, we need to change the construction of the *Setup*() algorithm of the original scheme as follows. We assume the bit string for each attribute is defined in a fixed format "(attribute description, version j)". The version number j of each attribute in the universe will be published.

Setup(n) Choose a random number $y \in \mathbb{Z}_p$ and let $g_1 = g^y$. Next, choose a

[6] Following the definition of [12], we assume the bilinear map is from $\mathbb{G}_1$ to $\mathbb{G}_2$, i.e., $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$.

[7] X is generated by applying a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ on the bit string representation of the attribute.

random element g_2 from $\mathbb{G}_1$. Then, select random numbers $w_1, w_2, \cdots, w_{n+1}$ from $\mathbb{Z}_p$. Define $t_i = g_2^{w_i}$ for $1 \leq i \leq n+1$. Define function T as:

$$T(X) = g_2^{X^n} \cdot \prod_{i=1}^{n+1} t_i^{\Delta_{i,N}(X)} = g_2^{X^n + p(X)},$$

where $p(\cdot)$ is a n degree polynomial defined by points $(1, w_1)$, $(2, w_2)$, $\cdots$, $(n+1, w_{n+1})$. The public parameters are output as: $PK = (g_1, g_2, t_1, \cdots, t_{n+1})$. The master key $MK = (y, w_1, w_2, \cdots, w_{n+1})$.

Algorithms *Encryption*, *Key Generation*, and *Decryption* are defined exactly the same as the original scheme. To enable the authority to generate proxy re-key's, we define algorithm *ReKeyGen* as follows.

ReKeyGen(γ, MK) γ is the set of attributes needing redefinition. For each attribute $X \in \gamma$, assuming its pre-image is a bit string "(attribute description, version j)", now redefine it as bit string "(attribute description, version $j+1$)". We hence obtain H(attribute name, version $j+1$) $= X'$ [8], where $H()$ is a cryptographic hash function. Since hash function $H(\cdot)$ is collision free, X' can not be used for any other attributes. The proxy re-key for attribute X will be output as $rk_X = \frac{(X')^n + p(X')}{X^n + p(X)}$. The set of proxy re-key's rk for attributes in γ are sent to proxy servers.

Proxy servers, on receiving the proxy key's, re-encrypt existing ciphertexts stored on them as follows.

ReEnc(E, rk, β) For each attribute $i \in \beta$, update E_i as $E_i^{rk_i}$.

To update user secret key, proxy servers update R_i components for users as follows.

ReKey(R_i, rk_i) Update R_i as $R_i^{(rk_i)^{-1}}$.

It is easy to verify that the updated user secret key will be able to decrypt

[8] When the new version $j+1$ is published, encryptors will use $T(X')$ as the public component for encryption thereafter.

ciphertexts encrypted with updated attributes if the ciphertext attributes satisfy the access structure of the secret key. Similar to our user revocation scheme for CP-ABE, our enhanced KP-ABE scheme places a minimal computation overhead on the authority when supporting user revocation, and enables the authority to manage her resources via resource constrained devices, e.g., iPhone, anywhere at any time.

3.6 Summary

In this chapter we addressed an important issue of user revocation for attribute based systems. In particular, we considered practical application scenarios in which semi-trustable proxy servers are available, and proposed a user revocation scheme for ABE. One nice property of our proposed scheme is that it places minimal load on authority upon attribute revocation events. We achieved this by uniquely combining the proxy re-encryption technique with ABE and enabled the authority to delegate most laborious tasks to proxy servers. Our proposed scheme is provably secure against chosen ciphertext attacks.

Chapter 4

Attribute-Based Encryption with User Accountability

This chapter addresses the issue of user accountability in ABE. In particular, we are interested in key abuse attacks in which authorized but malicious users duplicate and share their secret keys (and hence their access privileges) with unauthorized users. As we previously discussed, these kind of attacks are extremely harmful since key abusers are able to very easily execute these attacks at a very low cost. Complete prevention of these attacks is believed to be hard. In order for defending against these attacks, the common method, e.g., traitor tracing mechanisms [36–39] in conventional broadcast encryption, is to enable the authority (which could also be the data owner herself) to identify the original user(s) of any duplicated key(s) and hence obtain evidences of key abuse. In practical applications, it may not always be possible for the authority to obtain the duplicated key and test its legitimacy. This is because the authority may not have physical access to the key, or the unauthorized user would protect the duplicated key using various techniques such as hardware protection, re-randomization and etc. Therefore, the tracing mechanism should be

able to support black-box tracing, i.e., tracing the unauthorized user (a.k.a., pirate device) only by observing its outputs on given inputs.

The main challenge for this work is on how to trick the pirate device into decrypting the given tracing ciphertexts (inputs) and outputing the results. In order for doing so, the main difficulty is to make the tracing ciphertexts indistinguishable from normal (non-tracing) ciphertexts but uniquely correlated to the suspected user's identity. This chapter resolves these challenges and provides an abuse free scheme for ABE. More precisely, this scheme addresses the case for KP-ABE and proposes an abuse free KP-ABE scheme (AFKP-ABE). Our techniques used in AFKP-ABE can also be applied to CP-ABE for constructing an abuse free CP-ABE scheme (AFCP-ABE). For brevity, this chapter just presents our construction of AFKP-ABE. In this work, we formulate the security issue with the similar security model of traitor tracing mechanisms. Security of the proposed scheme is proved under the Decisional Bilinear Diffie-Hellman (DBDH) assumption and the Decision Linear (D-Linear) assumption. To the best of our knowledge, this work is among the first that addresses the key abuse attacks in ABE.

4.1 Main Idea

The intuition of our construction can be summarized as the follows. We define a n-bit user identity space and each bit of them is defined as an attribute with two occurrences, one for bit value 0 and the other for bit value 1. Each user is then assigned a unique ID from the identity space. The encryption algorithm will associate these identity-related attributes to the ciphertext in the following way: for normal (non-tracing) operations, all these n attributes are set as "don't care"; for tracing operations, they are set to represent the suspicious identity. In tracing operations, a

user is able to decrypt the ciphertext if and only if his identity equals the suspicious one. To make tracing ciphertexts indistinguishable from normal ciphertexts, we hide these identity-related attributes in the way that any user is not able to tell which and how many of them are set as "interested" (i.e., not "don't care"). In addition, we also hide some normal attributes so that upon a fail decryption the user can not tell if it is caused by the mismatch of his ID or by his access privilege (without considering his ID). Thus, he is not able to distinguish a tracing activity from a normal (non-tracing) one. The security goal of our construction is to build such a KP-ABE scheme in which 1) any user without the correct decryption key is not able to tell a single bit of the message, and 2) given a pirate device, the authority is able to trick it into decrypting tracing ciphertexts and thus discover the identity of the original owner of the decryption key held by this device.

4.2 Definitions and Models

In this section, we present the definition of our abuse-free KP-ABE (AFKP-ABE) scheme as well as its security definition. The security definition of our scheme is consistent to traitor tracing schemes [39].

4.2.1 Definition of AFKP-ABE

The AFKP-ABE scheme has the following five algorithms:

Setup(1^λ, n) The setup algorithm is a randomized algorithm. It takes as input the security parameter 1^λ and n, the length of a user identity. It outputs a master key MK and public parameters PK.

$Enc(M,\ \gamma, PK)$. The encryption algorithm is a randomized algorithm. It takes as input a message M, a set of attributes γ, and the public parameters PK. It outputs a ciphertext E. On different input γ, this algorithm can be used either for normal (non-tracing) operations of content distribution, or for the purpose of tracing.

$KeyGen(T,\ MK,\ PK)$. The key generation algorithm is a randomized algorithm. It takes as input an access structure T, the master secret key MK, and the public parameters PK. It outputs a user secret key SK.

$Dec(E,\ SK,\ PK)$. The decryption algorithm is a deterministic algorithm. It takes as input the ciphertext E for a set of attributes γ, a user secret key SK for an access structure T, and the public parameters PK. If $\gamma \models T$, i.e., γ satisfies T, it outputs the message M. Otherwise it outputs $\perp$ with overwhelming probability.

$Trace^{\mathcal{D}}(\varepsilon)$ This algorithm takes input a parameter ε (which should be polynomially related to λ), and has black-box access to an ε-useful decoder box $\mathcal{D}$ which is constructed by the adversary. It outputs a set of guilty colluders in polynomial time.

4.2.2 Definition of Attributes

We define three set of attributes: *public normal attributes*, *hidden normal attributes* and *hidden identity-related attributes*. We denote the universe of each of them by $\mathcal{U}_{PN}$,$\mathcal{U}_{HN}$, and $\mathcal{U}_{HID}$ respectively. The letter P in the subscription denotes the word "public", H means "hidden", N represents "normal", and ID is the abbreviation of "identity". $\mathcal{U}_{PN}$ and $\mathcal{U}_{HN}$ contain attributes to be used by normal encryptions.

$\mathcal{U}_{HID}$ contains identity-related attributes for describing the suspected user's identity and is particularly used for tracing. In ciphertexts, the associated attributes from $\mathcal{U}_{HN}$ and $\mathcal{U}_{HID}$ have to be hidden such that any receiver is not able to tell which and how many of them are used, while attributes from $\mathcal{U}_{PN}$ are public. Each attribute in $\mathcal{U}_{HID}$ has two occurrences, one for bit value 0 and the other for bit value 1. Similarly, we assume that attributes in $\mathcal{U}_{HN}$ also have binary values like those in $\mathcal{U}_{HID}$. This assumption is just for concise presentation of our scheme. Extending our scheme to support the non-binary case is trivial. From now on we will call the union of $\mathcal{U}_{HID}$ and $\mathcal{U}_{HN}$ as *hidden attributes* by capturing their common property of "hidden". We denote the universe of hidden attributes as $\mathcal{U}_H$, and thus $\mathcal{U}_H = \mathcal{U}_{HN} \cup \mathcal{U}_{HID}$. We denote the number of attributes in $\mathcal{U}_{HN}$ by m and that in $\mathcal{U}_{PN}$ by k. Therefore, the total number of hidden attributes is $m + n$.

According to the above discussion, it is clear that in a ciphertext there could be three types of attributes: attributes from $\mathcal{U}_{PN}$, attributes from $\mathcal{U}_{HN}$, and those from $\mathcal{U}_{HID}$. We denote the set of these three type of attributes in a ciphertext by γ_{PN}, γ_{HN}, and γ_{HID} respectively. Therefore, we have $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$, where γ represents the set of all the attributes interested by the encryptor.

Access Structure Our definition of the access structure (implemented using an access tree) is the same as KP-ABE [12], i.e., each interior node of the tree is a threshold gate and the leaves are associated with attributes. However, our construction has the following restrictions on the access structure: (1) each access structure should deal with all the hidden attributes and all of them should appear on the second layer of the tree; (2) the root node has to be an AND gate; (3) all the attributes from $\mathcal{U}_{PN}$ should appear in a subtree which we denote by T_R. Interior nodes of the subtree T_R could be any kind of threshold gates. The structure of the access tree in our construction is illustrated by Fig. 7.1. In addition, each non-root

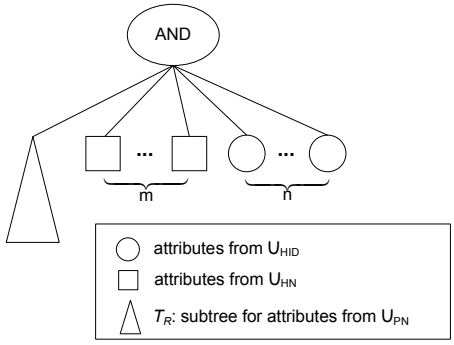


Figure 4.1: The form of access structure

node has a unique index given by its parent. For the convenience of representation, we will denote a node x's parent by x_{pa} and x's index by $idx(x)$.

4.2.3 Security Definition

The security of ABKP-ABE is defined by the following two security games.

Game 1. The first game captures the idea of *Semantic Security*. In our scheme we follow the definition of the standard game used by KP-ABE [12] which proceeds with the following steps.

- *Init* The adversary declares the set of attributes, γ, that he wishes to be challenged upon.
- *Setup* The challenger runs the Setup algorithm of AFKP-ABE and gives the public parameters to the adversary.
- *Phase 1* The adversary is allowed to issue queries for private keys for many access structures T_i , where $\gamma \nvDash T_i$ for all i.
- *Challenge* The adversary submits two equal length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with γ. The ciphertext is passed to the adversary.

- *Phase 2* Phase 1 is repeated.

- *Guess* The adversary outputs a guess b_0 of b.

The advantage of an adversary $\mathcal{A}$ winning this game is defined as $Adv_{SS} = Pr[b_0 = b] - \frac{1}{2}$.

Game 2. The second game captures the notion of *Traceability against partial collusion*. Our definition of the traceability game is based on that of [39]. Given λ, n, and ε, the game proceeds with the following steps.

- *Setup* The adversary $\mathcal{A}$ outputs a set $U = \{u_1, u_2, \ldots, u_t\}$ of colluding users with the only restriction that no pair of users have exactly the same access privilege. The access structure associated with user $u_i \in U$ is denoted by T_i.

- *Key Generation* The challenger runs the key generation algorithm $KeyGen$ to provide the user secret key for each user in U.

- The adversary $\mathcal{A}$ outputs a pirate device $\mathcal{D}$.

- The challenger runs $Trace^{\mathcal{D}}(\varepsilon)$ to obtain a set S.

We say that the adversary $\mathcal{A}$ wins the game if the following two conditions hold:

1. The decoder $\mathcal{D}$ is ε-useful, i.e., for a randomly chosen M in the finite message space, we have that $Pr[\mathcal{D}(Enc(M, \gamma, PK)) = M] \geq \varepsilon$ if there exists a user $u_i \in U$ with $\gamma \models T_i$, where γ is chosen in the way that makes Enc run under normal (non-tracing) operation.

2. The set S is either empty, or is not a subset of U.

We denote the probability that the adversary $\mathcal{A}$ wins this game by Adv_{TR}. If U contains exactly one user, this game captures the notion of *Traceability against single pirate.*

Definition 4.2.1 *We say that AFKP-ABE is secure if Adv_{SS} and Adv_{TR} are negligible (in λ) for any polynomial time adversary $\mathcal{A}$ and any constant $\varepsilon > 0$.*

To prove the security of AFKP-ABE in Game 2, another required security game is the *Indistinguishability Game* which captures the notion that, it is hard to distinguish ciphertexts generated by normal (non-tracing) operations from those generated by tracing operations. This game captures the idea that ciphetexts generated by tracing operations are indistinguishable from those generated by normal (non-tracing) operations. In AFKP-ABE, these two types of ciphertexts are generated by running our encryption algorithm over different sets of attributes. To differentiate these two types of ciphertexts is actually equal to telling which set of attributes are used in a given data encryption operation. In a tracing operation, we set γ_{HID} to represent the suspicious identity, while in a normal (non-tracing) operation we set γ_{HID} to represent the identity of "$*\,*\cdots*$", i.e., each bit if ID is set as "don't care". We define the *Indistinguishability Game* by the following steps: $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$

Indistinguishability Game

- *Init* The adversary $\mathcal{A}$ selects two sets of attributes to be challenged upon: $\gamma_0 = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ and $\gamma_1 = \gamma_{PN} \cup \gamma_{HN} \cup \gamma^*_{HID}$, where γ_{HID} represents a certain identity ID_i, and γ^*_{HID} denotes the identity of "$*\,*\cdots*$", i.e., each bit if ID is set as "don't care". $\mathcal{A}$ submits these two sets of attributes to the challenger $\mathcal{C}$.

- *Setup* The challenger $\mathcal{B}$ runs the setup algorithm of AFKP-ABE and give public parameters PK to $\mathcal{A}$.

- *Phase 1* $\mathcal{A}$ asks for the secret key of access structure T. If $(\gamma_0 \models T \wedge \gamma_1 \models T)$ or $(\gamma_0 \not\models T \wedge \gamma_1 \not\models T)$, the challenger $\mathcal{B}$ answers the query and gives $\mathcal{A}$ the corresponding secret key SK_T. The adversary $\mathcal{A}$ can repeat this step polynomially many times.

- *Challenge* $\mathcal{A}$ submits two equal length messages M_0 and M_1 to $\mathcal{B}$. If $\mathcal{A}$ a secret key SK_T for which $(\gamma_0 \models T \wedge \gamma_1 \models T)$, it is required that $M_0 = M_1$. $\mathcal{B}$ flips a binary fair coin b and encrypts M_b using attribute set γ_b. The ciphertext is given to $\mathcal{A}$.

- *Phase 2* Repeat Phase 1. If $M_0 \neq M_1$, $\mathcal{A}$ can not submit secret key query for access structure T for which $(\gamma_0 \models T \wedge \gamma_1 \models T)$.

- *Guess* The adversary $\mathcal{A}$ outputs a guess b' of b.

4.3 Our Construction

In this section, we present our construction of the secure AFKP-ABE scheme.

4.3.1 AFKP-ABE Scheme

In the description, $\mathbb{G}_0$ is a bilinear group of prime order p and g is a generator of $\mathbb{G}_0$. We use $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ to represent a bilinear map. The Lagrange coefficient $\Delta_{i,S}(x)$ is defined as follows, where $i \in \mathbb{Z}_p$, $x \in \mathbb{Z}_p$ are variables, and $S \subset \mathbb{Z}_p$ is some

set.

$$\Delta_{i,S}(x) := \prod_{j \in S \setminus \{i\}} \frac{x-j}{i-j}.$$

We use strings of length n to represent user IDs. "don't care" bit of an ID is represented by a "$*$".

Setup$(1^\lambda, n)$ Define $\mathcal{U}_H = \{1, \cdots n, n{+}1, \cdots m{+}n\}$, where the first n elements are for $\mathcal{U}_{HID}$ and the last m for $\mathcal{U}_{HN}$, and $\mathcal{U}_{PN} = \{1, 2, \cdots k\}$. For each attribute $i \in \mathcal{U}_{PN}$, choose a random number t_i from $\mathbb{Z}_p$. Then for each hidden attribute $j \in \mathcal{U}_H$, choose random numbers $\{a_{j,t}, b_{j,t}\}_{t=0,1}$ from $\mathbb{Z}_p$ and random points $\{A_{j,t}\}_{t=0,1}$ from $\mathbb{G}_0$. Finally, choose a random number y from $\mathbb{Z}_p$. The public parameters PK are published as

$$PK = (Y = e(g,g)^y, \{T_i = g^{t_i}\}_{i \in \mathcal{U}_{PN}}, \{A_{j,t}^{a_{j,t}}, A_{j,t}^{b_{j,t}}\}_{j \in \mathcal{U}_H, t=0,1})$$

and the master key MK is

$$MK = (y, \{t_i\}_{i \in \mathcal{U}_{PN}}, \{a_{j,t}, b_{j,t}\}_{j \in \mathcal{U}_H, t=0,1})$$

Enc(M, γ, PK) Define $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ as mentioned before. Let the ID represented by γ_{HID} be $X_n X_{n-1} \cdots X_1$, where $X_i = 0, 1$ or $*$ for each $1 \leq i \leq n$. The encryptor generates ciphertext components for γ_{HID} as follows. First choose a random number s from $\mathbb{Z}_p$. Then for each $1 \leq i \leq n$, pick random numbers $r_{i,0}$ and $r_{i,1}$ from $\mathbb{Z}_p$, and compute tuples $\{[\hat{E}_{i,t}, \check{E}_{i,t}]\}_{t=0,1}$ as follows.

(1) If $X_i = b$, where $b = 0|1$, the encryptor sets $[\hat{E}_{i,1-b}, \check{E}_{i,1-b}]$ as random (*mal-formed*), and $[\hat{E}_{i,b}, \check{E}_{i,b}] = [(A_{i,b}^{b_{i,b}})^{r_{i,b}}, (A_{i,b}^{a_{i,b}})^{s-r_{i,b}}]$ (*well-formed*).

(2) If $X_i = *$, for $t = 0, 1$ the encryptor sets $[\hat{E}_{i,t}, \check{E}_{i,t}] = [(A_{i,t}^{b_{i,t}})^{r_{i,t}}, (A_{i,t}^{a_{i,t}})^{s-r_{i,t}}]$ (*well-formed*).

Ciphertext components for γ_{HN} are generated in the same way as γ_{HID}. The encryptor generates ciphertext components for γ_{PN} as follows. For each $i \in \gamma_{PN}$, compute $E_i = T_i^s$. Finally, the ciphertext is output as follows.

$$E = (\gamma_{PN}, \tilde{E} = MY^s, E_0 = g^s, \\ \{E_i\}_{i\in\gamma_{PN}}, \{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{i\in\gamma_{HN}\cup\gamma_{HID}})$$

KeyGen(T, MK, PK) The access structure T is defined as mentioned before: the root node of the tree is an AND gate, all the hidden attributes appear on the second layer of the tree, and all the public normal attributes are in the subtree T_R. The trusted authority generates the user secret key as follows.

(1) For the subtree T_R, choose a polynomial q_x for each node x, including all the leaf nodes, of the tree in the top-down manner as follows. Starting from the root node r of T_R (with the threshold value k_r), choose a random number u from $\mathbb{Z}_p$ and set $q_r(0) = u$. Then randomly choose $k_r - 1$ other points to define the $(k_r - 1)$-degree polynomial q_r completely. For any other node x, q_x is generated in the same way and $q_x(0) = q_{x_{pa}}(idx(x))$.

After having defined the polynomials, the following secret key component is generated for each leaf node x in T_R:

$$D_x = g^{\frac{qx(0)}{t_i}}$$

where i denotes the attribute in $\mathcal{U}_{PN}$ associated with node x. We use $\mathcal{L}_{TR}$ to represent the set of all the leaf nodes in T_R.

(2) Secret key components for attributes from $\mathcal{U}_{HID}$ are generated as follows.

Assume the user is assigned a unique identity $ID = X_n X_{n-1} \cdots X_1$, where $X_i = 0|1$ for each $1 \leq i \leq n$. Then for each attribute i in $\mathcal{U}_{HID}$, the authority chooses random numbers v_i and λ_i from $\mathbb{Z}_p$ and outputs a triple $[\tilde{D}_i, \hat{D}_i, \check{D}_i]$ as follows.

$$\tilde{D}_i = g^{v_i}(A_{i,X_i})^{a_{i,X_i} b_{i,X_i} \lambda_i}, \quad \hat{D}_i = g^{a_{i,X_i}\lambda_i}, \quad \check{D}_i = g^{b_{i,X_i}\lambda_i}.$$

(3) Secret key components for attributes from $\mathcal{U}_{HN}$ are generated in the same way as $\mathcal{U}_{HID}$.

(4) The authority sets $v = \sum_{i \in \mathcal{U}_H} v_i$ and generates a secret key component $D_0 = g^{y-u-v}$.

Finally, the authority outputs the following as the user secret key (SK):

$$SK = (D_0, \{D_i\}_{i \in \mathcal{L}_{TR}}, \{\tilde{D}_i, \hat{D}_i, \check{D}_i\}_{i \in \mathcal{U}_H})$$

Dec(E, SK, PK) The receiver decrypts the ciphertext E by applying his secret key components to the ciphertext as follows.

(1) Apply secret key components for public normal attributes to the ciphertext. For each leaf node x of T_R, assuming x is associated with attribute $i \in \mathcal{U}_{PN}$, calculate the following (the result is denoted by F_x):

$$F_x = \begin{cases} e(D_i, E_i) = e(g,g)^{sq_x(0)}, & \text{if } x \in \gamma_{PN} \ ; \\ \perp, & \text{otherwise.} \end{cases} \tag{4.1}$$

Then execute recursively for each non-leaf node z of T_R in the bottom-up manner as follows. For each child node x of z, if $F_x \neq \perp$ add x into a set S_z until S_z has k_z elements, where the set S_z is initialized to empty. If not able to construct such a

k_z-sized set S_z, let $F_z = \perp$. Otherwise, calculate F_z as follows.

$$\begin{aligned} F_z &= \prod_{x \in S_z} F_x^{\Delta_{x,Sz}(0)} \\ &= \prod_{x \in S_z} (e(g,g)^{sq_x(0)})^{\Delta_{x,Sz}(0)} \\ &= e(g,g)^{sq_z(0)} \end{aligned}$$

where derivation of the last two steps holds because $q_x(0) = q_z(idx(x))$ and $q_z(0) = \sum_{x \in S_z}(q_z(idx(x)) \cdot \Delta_{x,S_z}(0))$.

This recursion ends up with outputting $F_r = e(g,g)^{sq_r(0)}$ if $\gamma_{PN} \models T_R$. Since $q_r(0) = u$, we have $F_r = e(g,g)^{su}$.

(2) Apply secret key components for hidden attributes to the ciphertext. If the set of hidden attributes in the access structure contains all the attributes in γ_{HN} and γ_{HID}, output the result F_H as follows.

$$\begin{aligned} F_H &= \prod_{i \in \mathcal{U}_H} \frac{e(E_0, \tilde{D}_i)}{e(\hat{E}_i, \hat{D}_i) e(\check{E}_i, \check{D}_i)} \\ &= e(g,g)^{sv} \end{aligned}$$

The message can be output as follows

$$\begin{aligned} M &= \frac{\tilde{E}}{e(E_0, D_0) F_r F_H} \\ &= \frac{M e(g,g)^{ys}}{e(g^s, g^{y-u-v}) e(g,g)^{su} e(g,g)^{sv}} \end{aligned}$$

Trace$^{\mathcal{D}}(\varepsilon)$ This algorithm takes as input ε and a ε-useful pirate device $\mathcal{D}$. We first show how to trace $\mathcal{D}$ which just holds one decryption key as follows. The tracing algorithm repeats the following steps $\frac{1}{\varepsilon}$ times for each identity ID_i in the system identity list:

- Step 1. Choose a set of attributes $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ such that γ satisfies the access structure of ID_i and γ_{HID} just contains the attributes corresponding to bits of ID_i.

- Step 2. Choose a random message M from the finite message space. Let $E \leftarrow Enc(M, \gamma, PK)$.

- Step 3. Test if $\mathcal{D}$ correctly decrypts E. If it does, stop and return with ID_i. Otherwise continue.

If at the end of these repetitions the algorithm does not return with any identity, return FAIL and stop the experiment. Tracing $\mathcal{D}$ which holds more than one decryption keys is similar with the exception that, in step 3 add ID_i into the guilty user set S instead of returning immediately, where S is initialized as empty. If at the end of these repetitions S is empty, return FAIL and stop the experiment.

4.3.2 Security Proof

We show the security of our scheme as follows.

Lemma 4.3.1 *If a polynomial-time adversary $\mathcal{A}$ can win Game 1 with non-negligible advantage Adv_{SS}, then we can build a simulator $\mathcal{B}$ that is able to solve the DBDH problem with advantage $\frac{1}{2}Adv_{SS}$.*

Proof: In the DBDH game, the challenger chooses random numbers a, b, c from $\mathbb{Z}_p$ and flips a fair coin μ. If $\mu = 0$, set $z = abc$; If $\mu = 1$, set z as a random value in $\mathbb{Z}_p$. $\mathcal{B}$ is given (A, B, C, Z)=$(g^a, g^b, g^c, e(g, g)^z)$ and asked to output μ. To answer this challenge, $\mathcal{B}$ then simulates Game 1 as follows.

Init $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the set of attributes $\gamma = \gamma_{PN} \cup \gamma_{HN} \cup \gamma_{HID}$ it wants to be challenged upon. We denote the identity represented by γ_{HID} by $X_n X_{n-1} \cdots X_0$, where $X_i = 0, 1$ or $*$, for $1 \leq i \leq n$.We denote the set $\gamma_{HN} \cup \gamma_{HID}$ by γ_H.

Setup $\mathcal{B}$ creates public parameters as follows. First, set $Y = e(A, B) = e(g, g)^{ab}$. Then, for each attribute $i \in \mathcal{U}_{PN}$, generate T_i by the following steps:

- choose a random number $t_i \in \mathbb{Z}_p$.
- if $i \in \gamma_{PN}$, sets $T_i = g^{t_i}$; otherwise, set $T_i = g^{bt_i} = B^{t_i}$.

For each attribute $i \in \mathcal{U}_{HID}$, choose two random numbers $h_{i,0}$ and $h_{i,1}$ from $\mathbb{Z}_p$. Then proceed as follows.

- if $X_i = *$, $A_{i,t} = g^{h_{i,t}}$, $t = 0, 1$; otherwise, $A_{i,X_i} = g^{h_{i,X_i}}$ and $A_{i,1-X_i} = g^{bh_{i,1-X_i}} = B^{h_{i,1-X_i}}$.
- choose random numbers $\{a_{i,t}, b_{i,t}\}_{t=0,1}$ from $\mathbb{Z}_p$.

Attributes in $\mathcal{U}_{HN}$ are processed in the same way as $\mathcal{U}_{HID}$. Finally, output PK as in the real scheme.

Phase I $\mathcal{A}$ submits a query for secret key of access structure T, where $\gamma \nvDash T$. Note that T has the structure of 7.1. $\mathcal{B}$ differentiates the following two cases and answers the query accordingly:

Case 1: In this case, $\gamma_{PN} \nvDash T_R$. $\mathcal{B}$ generates secret key components for hidden attributes as in the real scheme. To generate secret key components for attributes attached to T_R, $\mathcal{B}$ defines a recursive function $PolyDef(x)$ and runs it over the root node r of T_R. For each node x in T_R, use k_x and p_x to represent the node's threshold value and the number of its satisfied children respectively (the satisfied child is a child node of x that returns true over γ_{PN}).

`PolyDef(x)`: It is defined by the following steps:

- Define q_x as follows.
  - If x is not r, set $q_x(0) = q_{x_{pa}}(idx(x))$; otherwise, set $q_x(0) = ab + br'$, r' is randomly chosen from $\mathbb{Z}_p$.

– Select d $(= k_x - 1)$ children of x. For each selected child i, choose a random number r'_i from $\mathbb{Z}_p$ and let $q_x(idx(i)) = br'_i$. This completes the construction of polynomial q_x. Note that, if $p_x \leq d$, the set of selected children should include all the p_x satisfied ones; otherwise, all the d selected children should be satisfied ones. We denote the set of these selected children of x plus x itself by X_s.

- For each remaining child j (not selected by the above step), calculate $q_x(j) = \sum_{i \in X_s} q_x(idx(i))\Delta_{i,S_x}(j)$.

- For each child i of x, run $PolyDef(i)$.

When $PolyDef(r)$ terminates, $\mathcal{B}$ completes the construction of the polynomials for all the nodes in T_R. In particular, $p_r(0) = ab + br'$. Note that, in our construction of polynomials, for each node x, the polynomial values have the following properties:

(1) If $q_x(0)$ has the form of $R_x b$, then for each of its children i, $q_i(0)$ $(= q_x(idx(i)))$ has the form of $R_i b$.

(2) If $q_x(0)$ has the form of $C_x ab + R_x b$, then for each of its children i, (i) if $i \in X_s$ (selected), $q_i(0)$ has the form of $R_i b$; otherwise, (ii) $q_i(0)$ has the form of $C_i ab + R_i b$.

(3) In (1) and (2), C_x, R_x, C_i, and R_i are functions of Lagrange coefficients and random numbers (i.e., r'_j's), and independent of a and b.

From these properties, we may categorize a leaf nodes x into one of the following three types:

(1) Type A: $x \in \gamma_{PN}$, i.e., x is a satisfied node. $q_x(0)$ has the form of $R_x b$.

(2) Type B: $x \notin \gamma_{PN}$ but one of x's ancestors (including x itself) is selected by its parent. $q_x(0)$ has the form of $R_x b$.

(3) Type C: all the other leaf nodes, $q_x(0)$ has the form of $C_x ab + R_x b$.

Therefore, the secret key component corresponding to each leaf node x of T_R is given as follows

$$D_x = \begin{cases} g^{\frac{Rxb}{tx}} = B^{\frac{Rx}{tx}}, & x \text{ in Type A.} \\ g^{\frac{Rxb}{txb}} = g^{\frac{Rx}{tx}}, & x \text{ in Type B.} \\ g^{\frac{Cxab+Rxb}{txb}} = A^{\frac{Cx}{tx}} g^{\frac{Rx}{tx}}, & x \text{ in Type C.} \end{cases}$$

The secret key component D_0 of SK is output as follows

$$g^{y-u-v} = g^{ab-q_r(0)-v} = g^{-br'} g^{-v} = B^{-r'} g^{-v}$$

where v is generated when constructing secret key components for hidden attributes. All the other components are generated as in the real scheme.

Case 2: In this case, $\gamma_{PN} \models T_R$, but the hidden attributes of T do not match with γ_H. Let a hidden attribute j that is not intended by γ_{HID} be the witness. $\mathcal{B}$ generates secret key components corresponding to T_R as in the real scheme. $\mathcal{B}$ generates secret key components for hidden attributes as follows.

- For hidden attributes $1 \leq i \leq m+n$, pick v'_i randomly from $\mathbb{Z}_p$. Set $v_j = ab+v'_j$ and $v_i = v'_i$ for every $i \neq j$. Finally set $v = \sum_{i=1}^{m+n} v_i = ab + \sum_{i=1}^{m+n} v'_i$.

- compute the secret key components $[\tilde{D}_j, \hat{D}_j, \check{D}_j]$ of attribute j as follows.

$$\begin{aligned} \tilde{D}_j &= g^{v_j} (A_{j,X_j})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{ab+v'_j} (A_{j,X_j})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{ab+v'_j} (g^{bh_{j,X_j}})^{a_{j,X_j} b_{j,X_j} \lambda_j} \\ &= g^{v'_j} (g^{bh_{j,X_j}})^{a_{j,X_j} b_{j,X_j} \lambda'_j} \end{aligned}$$

where λ'_j is chosen by $\mathcal{B}$ and $\lambda_j = \frac{a}{h_{j,X_j} a_{j,X_j} b_{j,X_j}} + \lambda'_j$. $\mathcal{B}$ calculates $[\hat{D}_j, \check{D}_j]$ and

$[\tilde{D}_i, \hat{D}_i, \check{D}_i]$ for $i \neq j$ as in the real scheme.

- Output D_0 of SK as: $D_0 = g^{ab-u-v} = g^{-u-\sum_{i=1}^{m+n} v'_i}$, where u is generated when constructing secret key components for T_R.

All the other components are generated as in the real scheme.

From the above description, we can see that $\mathcal{B}$ is able to construct a secret key of T in both cases. Furthermore, the distribution of the secret key of T is the same as that in the original scheme. The adversary $\mathcal{A}$ can repeat this step for polynomial times.

Challenge The adversary $\mathcal{A}$ submits two equal length challenge messages m_0 and m_1 to $\mathcal{B}$. $\mathcal{B}$ flips a fair binary coin v and picks out m_v. The ciphertext of m_v is output as: $E = (\gamma_{PN}, \tilde{E} = m_v Z, E_0 = C, \{E_i = C^{t_i}\}_{i \in \gamma_{PN}}, \{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{i \in \gamma_H})$. Note that $\mathcal{B}$ can construct $\{\{\hat{E}_{i,t}, \check{E}_{i,t}\}_{t=0,1}\}_{i \in \gamma_H}$ because if the occurrence t of attribute i is in γ_H, $A_{i,t}$ does not contain the unknown value b, and if the occurrence t of i is not in γ_H, $\{\hat{E}_{i,t}, \check{E}_{i,t}\}$ are just chosen at random. If $\mu = 0$ it is easy to show that the ciphertext is a valid random encryption of message m_v. Otherwise, if $\mu = 1$, then $Z = e(g,g)^z$ and $\tilde{E} = m_v e(g,g)^z$. Since z is random, $\tilde{E}$ is just a random element of $\mathbb{G}_1$ from the adversary's view and contains no information about m_v.

Phase II The simulator acts exactly as it did in Phase I.

Guess The adversary $\mathcal{A}$ submits a guess v' of v. If $v' = v$, $\mathcal{B}$ outputs $\mu' = 0$, indicating that the given DBDH-tuple is a valid one. Otherwise it outputs $\mu' = 1$, indicating that the given DBDH-tuple is just a random quadruple. In the case of $\mu = 1$, the ciphertext E contains no information about m_v. Therefore, v' is just a random guess of v, and thus μ' is just a random guess of μ. Thus, we have $Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, the ciphertext E is a valid encryption of m_v. Since by definition $\mathcal{A}$ has the advantage of Adv_{SS} to output a correct guess, i.e.,

$v' = v$, $\mathcal{B}$ outputs $\mu' = 0$ with the probability of $\frac{1}{2} + Adv_{SS}$, i.e., $Pr[\mu' = \mu|\mu = 0] = \frac{1}{2} + Adv_{SS}$. Therefore, the overall advantage of $\mathcal{B}$ in the DBDH game is $\frac{1}{2}Pr[\mu' = \mu|\mu = 0] + \frac{1}{2}Pr[\mu' = \mu|\mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + Adv_{SS}) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2}Adv_{SS}$.

Lemma 4.3.2 *If a polynomial-time adversary $\mathcal{A}$ can win our Indistinguishability Game (see Appendix B) with advantage Adv_{IND}, then we can build a simulator $\mathcal{B}$ that is able to solve the D-Linear problem with advantage $\frac{1}{2}Adv_{IND}$.*

Proof: We use a series of games to prove the security of this game as [40]. *Game Ind_1* is defined in the same way as the original game except that in γ_0, γ_{HID} represents the identity of "$**\cdots*X_1$", i.e., the upper $n-1$ bits are set as "don't care" but keep the first bit the same as in the original game. *Game Ind_2* is defined in the same way that γ_{HID} represents the identity of "$**\cdots*X_2X_1$", i.e., the upper $n-2$ bits are set as "don't care" but keep the first bit the same as in the original game, so on and so forth. Our original game is thus *Game Ind_n*. To prove the security of our scheme, it is enough to prove that it is indistinguishable between *Game Ind_i* and *Game Ind_{i+1}*. We can use the similar technique used by [40] to prove this. For brevity, we do not present the complete proof here.

Lemma 4.3.3 *If Adv_{IND} and Adv_{SS} are negligible, Adv_{TR} is negligible.*

Proof: Given a pirate device $\mathcal{D}$, our tracing algorithm $Trace^{\mathcal{D}}(\varepsilon)$ will try with each identity ID_i in the system identity list. We denote the attribute set chosen for testing ID_i by $\gamma^i = \gamma^i_{PN} \cup \gamma^i_{HN} \cup \gamma^i_{HID}$. We define the corresponding attribute set used for normal (non-tracing) encryption as $\bar{\gamma}^i = \gamma^i_{PN} \cup \gamma^i_{HN} \cup \bar{\gamma}^i_{HID}$. The only difference between the two sets of attributes is that, in γ^i_{HID} all the attributes

corresponding to bits of ID_i are set as "interested", but in $\bar{\gamma}^i_{HID}$ all the identity-related attributes are set as "don't care". Based on this definition, we define the following two probabilities:

$$\begin{aligned} p_i &= \Pr[\mathcal{D}(Enc(M, \gamma^i, PK)) = M] \\ p &= \Pr[\mathcal{D}(Enc(M, \bar{\gamma}^i, PK)) = M] \end{aligned}$$

where M is a random message picked from the message space. We distinguish between the following three types of ε-useful pirate devices that the pirate can generate, where ε is some fixed constant:

1. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is non-negligible for some identity ID_i.

2. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is negligible for each identity ID_i, but the tracing algorithm $Trace^{\mathcal{D}}(\varepsilon)$ outputs an empty set.

3. Pirate device $\mathcal{D}$ for which $|p - p_i|$ is negligible for each identity ID_i, but the tracing algorithm $Trace^{\mathcal{D}}(\varepsilon)$ outputs a set which is not contained in the set of colluding users U.

It is obvious that we can use any pirate producing type 1) devices to win the *Indistinguishability Game* with non-negligible advantage. We now show the rough idea of how we can use any pirate producing type 2) devices to win the *Indistinguishability Game* with non-negligible advantage. Assume the set of colluding users that the pirate claims to be able to collect is $U = \{u_1, u_2, \cdots, u_t\}$. Now denote the challenger of the *Indistinguishability Game* as $\mathcal{C}$, the simulator we want to build is $\mathcal{B}$, and the pirate is $\mathcal{A}$. Then the simulator we build executes as follows.

- Init. $\mathcal{B}$ presents $\mathcal{C}$ two attribute sets $\gamma_0 = \gamma^i$ and $\gamma_1 = \bar{\gamma}^i$ to be challenged upon, where γ^i is the attribute set that can be used to test user $u_i \in U$ by our

tracing algorithm.

- Setup. $\mathcal{C}$ generates public parameters and give them to $\mathcal{B}$.
- Phase 1. $\mathcal{B}$ asks $\mathcal{C}$ to give him secret keys for all the users in U. Then $\mathcal{B}$ gives all these keys to $\mathcal{A}$ to answer key queries in the key generation phase of Game 2.
- Challenge. $\mathcal{B}$ submits two equal length messages M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ flips a coin and encrypts M_b with γ_b. Then the ciphertext is given to $\mathcal{B}$.
- Phase 2. $\mathcal{B}$ submits more secret key queries.
- Guess. $\mathcal{B}$ asks $\mathcal{A}$ to decrypt the ciphertext given by $\mathcal{C}$. If the message returned by $\mathcal{A}$ is one of M_1 and M_0, $\mathcal{B}$ answers $b_0 = 1$. Otherwise, $\mathcal{B}$ answers $b_0 = 0$.

The advantage for our simulator $\mathcal{B}$ to win the *Indistinguishability Game* is $\frac{1}{\varepsilon}$ times the advantage that the type 2) devices, which are generated by $\mathcal{A}$, output the empty set.

It is easy to show that type 3) devices can be used to win Game 1 (the semantic security game). The intuition is that, type 3) devices can correctly decrypt a message which is encrypted for users whose secret keys are not known to type 3) devices with non-negligible advantage.

4.3.3 Efficiency Analysis

In AFKP-ABE, both the ciphertext size and the secret key size are linear to n, where n is the number of bits in the identity space. As the maximum number of users it can represent is $N = 2^n$, the complexity can be written as $\mathcal{O}(logN)$, where N is the total number of users. To trace a pirate, AFKP-ABE needs to try with every user's identity in the system list. When the number of users in a system is large, the

tracing algorithm would be inefficient. To resolve this issue, we can first test with some normal ciphertexts using combinations of normal attributes. For example, we can use different combinations of attributes like location, age, etc. In practice, this process will hopefully rule out a significant portion of users. Our tracing algorithm can just test over the remaining set of users.

4.4 Applications

In general, our proposed scheme is applicable to systems where 1) data can be categorized by their attributes and a user access privilege should be defined in the way that just allows the user to access certain intended subset of resources; 2) abuse of the access privilege should be prohibited. As we mentioned before, one important application scenario of our abuse free KP-ABE scheme is the area of copyright-sensitive targeted broadcast, especially commercial media broadcast systems. In these systems, contents usually have their commercial values and abuse of the access privilege usually causes legal concerns. Another important application scenario of our proposed scheme would be audit log systems. As these systems would be widely used in applications such as network management, audit logs may contain sensitive information and disclose of them to unauthorized parties would cause security concerns or privacy violations. Recently, we also witnessed application of KP-ABE in wireless networks environment. In [58], Yu et al. proposed a fine-grained data access control scheme for wireless sensor networks for mission-critical applications. In this work, data access control is well resolved by combining KP-ABE with some other cryptographic primitives. However, the issue of access privilege abuse is not addressed since it is yet another serious issue if we consider the application of mission-critical scenarios such as battle fields. We believe our AFKP-ABE can serve to enhance

their proposed scheme as the complexity of AFKP-ABE in terms of ciphertext size and secret key size is just $\mathcal{O}(logN)$, where N is the total number of users.

4.5 Summary

In this chapter, we focus on the key abuse attacks in attribute-based systems and proposed an abuse free KP-ABE (AFKP-ABE) scheme. To defend against the key abuse attacks, we introduce hidden attributes in the system such that the tracing algorithm can use them to identify any single pirate or partial colluding users. Our design enables black boxing tracing and does not require the well-formness of the user secret key. The complexity of AFKP-ABE in terms of ciphertext size and user secret keys size is just $\mathcal{O}(logN)$, where N is the total number of users. Our scheme is provably secure under DBDH assumption and D-Linear assumption. Notably, our techniques used in AFKP-ABE are also applicable to CP-ABE for providing an abuse free CP-ABE (AFCP-ABE) scheme.

Chapter 5

Privacy-Preserving Attribute-Based Encryption

In ABE data access policies are enforced by encryption. To facilitate user decryption, current constructions of ABE [2, 9, 12, 107, 108] attach the plaintexts of data attributes (in KP-ABE) or data access structures (in CP-ABE) to ciphertexts. These plaintexts, particularly data access structures in CP-ABE, reveal the data owner's data access policies when disclosed to untrusted servers, and hence have privacy concerns. In order for providing better privacy protection for CP-ABE, this chapter proposes two novel CP-ABE constructions under different security models. In our proposed schemes the access structure is hidden to both the untrusted server and the users, no matter authorized or unauthorized. To the best of our knowledge, these two schemes are among the first CP-ABE schemes with hidden access policies.

5.1 Our Construction under Generic Group Model

5.1.1 Definitions

Recall that CP-ABE works in the following way: users are assigned a set of attributes; data is associated with an access structure via encryption; a user is able to decrypt the data if and only if the user attributes satisfy the access structure of the data. In order for hiding the access structure of data, this work defines user attributes and the data access structure as follows.

Attributes Definition We differentiate two kinds of attributes: *application level attributes* and *algorithm level attributes*. Application level attributes refer to those meaningful to human being, e.g., skill, occupation, rank etc. Algorithm level attributes refer to those suitable for computer to interpret. Application level attributes can be mapped to algorithm level attributes. The mapping method is another interesting topic which is out of the scope of this work. In this work, an attribute refers to an algorithm level attribute which is defined in such a way that it has two possible occurrences: *positive* and *negative* which are denoted with symbols $Att_{i,1}$ and $Att_{i,0}$, where $i \in \mathbb{Z}_n$ is the index of attribute i and n is the total number of attributes in the system. We do not consider *don't care* case for any attribute and assume each algorithm level attribute is meaningful to every user. The set of attributes that each user possesses is $\{Att_{i,b} | \forall i \in \mathbb{Z}_n, b = 0|1\}$. In our scheme description, we also use the binary string $X_{n-1}X_{n-2}\cdots X_0$ to denote the set of attributes that the user possesses, where bit value 1 and 0 represent positive and negative occurrences of the attribute respectively.

Access Structure We use 1-level AND logic over algorithm level attributes to represent the access structure. For example, in the case of $n = 4$, an access structure may have the form (*attribute* 3 *is positive*) $\wedge$ (*attribute* 1 *is negative*).

If we use a product term to represent this access structure, it would be $X_3\bar{X}_1$. OR logic can be simulated using concatenation. Complex access structures over application level attributes can be easily realized using these logics. For example, in military scenarios we can realize the access structure "$rank > second\ lieutenant$" as follows: First, we map the application level attribute $rank$ to a set of algorithm level attributes by enumerating all the ranks: $\{\cdots, lieutenant, captain, \cdots\}$. Then, the logic "$rank > second\ lieutenant$" can be implemented by AND all the negative algorithm level attributes for ranks lower than lieutenant. In this way, we can realize access structures over application level attributes such as "$(rank > second\ lieutenant) \wedge (service\ year < 5\ years) \wedge (gender = female)$". In the remaining part of this work, we just consider the access structure over algorithm level attributes which can be represented via one product term, i.e., AND logic only. The term *attribute* in the remaining part of this chapter refers to algorithm level attribute.

5.1.2 Scheme Description

Our scheme is composed of four algorithms: *Setup*, *KeyGen*, *Encryption*, and *Decryption*. The functionality of each algorithm is almost the same as that in the standard CP-ABE scheme [2, 9]. The main difference is that our proposed scheme hides the access structure while current CP-ABE schemes do not. To achieve this goal our scheme omits the access structure T from the ciphertext CT. To enable decrypting without explicitly knowing the access structure, our scheme is designed as follows: the ciphertext in our construction comprises components for both positive and negative occurrence of all the attributes. It is designed in the way that it is hard for all but the authority herself to tell which attributes are used in the access structure. Data decryption requires the user to use secret key components

of all his attributes. After decryption, the user know nothing about which or how many attributes grant or decline him the access. This prevents the authorized users from knowing the access policy information of the authority (who is also the data owner). As our current design is not a public-key solution, only the authority can encrypt data and servers as the data owner. We leave the public-key construction as a future work.

The four algorithms of our scheme are defined as follows.

Setup This algorithm chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g. Each attribute is then mapped to an element of group $\mathbb{G}_0$. Let $h_{i,b}$ denote the corresponding element in group $\mathbb{G}_0$ of attribute $Att_{i,b}$. We have $h_{i,0} = g^{a_i}$ and $h_{i,1} = g^{b_i}$, where a_i and b_i are randomly generated from $\mathbb{Z}_p$. Let $\gamma_i = a_i + b_i$. a_i and b_i should be chosen in the way that a_i, b_i, and γ_i are all non-trivial. This algorithm also chooses other two random numbers $\alpha, \beta \in \mathbb{Z}_p$. The system master key (MK) is output as follows

$$MK = (\alpha, \beta, \{a_i, b_i\}_{\forall i \in \mathbb{Z}_n})$$

MK, $h_{i,0}$ and $h_{i,1}$ are only known to the authority.

KeyGen This algorithm takes as input a user's attribute set $X_{n-1}X_{n-2}\cdots X_0$ and generates her secret key as follows

$$SK = (D = g^{(\alpha+r)/\beta}, D' = g^r, D'' = g^{\beta r}, \{D_i = h_{i,\bar{X}_i}^r\}_{\forall i \in \mathbb{Z}_n})$$

where r is a random numbers chosen from $\mathbb{Z}_p$. X_i is the value of the i^{th} attribute and $\bar{X}_i$ is its inverse.

Encryption This algorithm takes as input the message (M), the access structure

in a product term, and the master key (MK). It outputs the ciphertext with the following format:

$$CT = (\tilde{C}, \check{C}, \{\hat{C}_j\}_{j=0,1}, \{C_i\}_{\forall i \in \mathbb{Z}_n})$$

where $\tilde{C} = (M||MAC) \cdot X$ and X is a blind factor used to hide $(M||MAC)$. MAC is the message authentication code for M. "||" means concatenation. $\check{C}$, $\hat{C}_j$'s, and C_i's are ciphertext components to help decryptors reconstruct X and thus derive M. Each bit of the user's attribute set $X_{n-1}X_{n-2}\cdots X_0$ corresponds to a ciphertext component C_i which is a triple as we will describe later in this section. Before presenting the detailed construction of CT, we define notation as follows: Each symbol in a *product term* is called a *literal*, denoted by X'_i if it is for the i^{th} bit of user attribute set. If the symbol has the form $\bar{X}$, $X'_i = 0$. Otherwise $X'_i = 1$. We denote the product term by S. The string "there is a literal in S for the i^{th} bit of a user's attribute set X_i" is represented by "$X_i \in S$". CT is constructed by the following steps:

• *Step 1.* Random Number Generation. Chooses random numbers $s_0, s_1, ..., s_{n-1}$, k_0, $k_1 \in \mathbb{Z}_p$, and set $\delta = \sum_{j=0}^{n-1} \gamma_i s_i$.

• *Step 2.* C_i Computation. C_i is a triple of the form $C_i = (g^{s_i}, C_{i,0}, C_{i,1})$ and s_i is a random number generated in *step 1.* Both $C_{i,0}$ and $C_{i,1}$ are elements of group $\mathbb{G}_0$. If $X_i \in S$, choose a random number $t_i \in \mathbb{Z}_p$ and calculates $C_{i,X'_i} = h_{i,X'_i}^{s_i+t_i}$ and $C_{i,1-X'_i} = h_{i,1-X'_i}^{s_i}$. Otherwise, output $C_{i,0} = h_{i,0}^{s_i}$ and $C_{i,1} = h_{i,1}^{s_i}$.

• *Step 3.* $\check{C}$ Computation and C_i update. First compute a value $g^{s'}$ as follows:

$$g^{s'} = \prod_{i=0}^{n-1} (C_{i,0} C_{i,1}) = g^{\delta + x}$$

	g^{s_i}	$C_{i,0}$	$C_{i,1}$
C_3	g^{s_3}	$g^{k_0} h_{3,0}^{s_3+t_3}$	$g^{k_1} h_{3,1}^{s_3}$
C_2	g^{s_2}	$g^{k_0} h_{2,0}^{s_2}$	$g^{k_1} h_{2,1}^{s_2+t_2}$
C_1	g^{s_1}	$g^{k_0} h_{1,0}^{s_1}$	$g^{k_1} h_{1,1}^{s_1}$
C_0	g^{s_0}	$g^{k_0} h_{0,0}^{s_0}$	$g^{k_1} h_{0,1}^{s_0+t_0}$

Table 5.1: Vector for the product $\bar{X}_3 X_2 X_0$

where x is some number in $\mathbb{Z}_p$ such that

$$g^x = \prod_{\forall j,\ X'_j \in S} h_{j,X'_j}^{t_j} \tag{5.1}$$

Then, compute $\check{C} = g^{\beta s'}$. Finally, update $C_{i,0}$ and $C_{i,1}$ for all $i \in \mathbb{Z}_p$ as follows:

$$C_{i,0} = g^{k_0} C_{i,0}, \quad C_{i,1} = g^{k_1} C_{i,1}$$

Table 5.1 illustrates an example vector (C_3, C_2, C_1, C_0) for product term $\bar{X}_3 X_2 X_0$, where $n = 4$.

• *Step 4.* Ciphertext Generation. Ciphertext is output as follows:

$$CT = (\tilde{C} = (M||MAC)e(g,g)^{\alpha s'}, \check{C} = g^{\beta s'}, \{\hat{C}_j = g^{\frac{k_j}{\beta}}\}_{j=0,1}, \{C_i\}_{\forall i \in \mathbb{Z}_n}) \tag{5.2}$$

where $MAC = hash(M)$. $hash(\cdot)$ is an one way hash function using algorithms such as SHA-1 [53].

Decryption This algorithm takes as input the ciphertext CT and the user's attribute set. It returns the message M if the user's attributes satisfy the access structure. Otherwise, it returns an error symbol $\perp$.

• *Step 1.* Credential Pairing. Assume the user's attribute set is $X_{n-1} X_{n-2} ... X_0$.

It first calculates

$$B_j = e(\hat{C}_j, D'') = e(g^{\frac{k_j}{\beta}}, g^{\beta r}) = e(g,g)^{rk_j}, \quad j = 0, 1.$$

Then, for each $i \in \mathbb{Z}_n$, pick C_{i,X_i} from C_i and compute a value F_i for bit i of her attribute set as follows:

$$\begin{aligned} F_i &= e(D_i, g^{s_i}) e(C_{i,X_i}, D') / B_{X_i} \\ &= e(h_{i,\bar{X}_i}^r, g^{s_i}) e(g^{k_{X_i}} h_{i,X_i}^{s_i+t_i}, g^r) / B_{X_i} \\ &= e(g,g)^{r\gamma_i s_i} e(g, h_{i,X_i})^{rt_i} \end{aligned} \tag{5.3}$$

In (3), $t_i = 0$ if $X_i \notin S$. Otherwise, $t_i \neq 0$. In the last step of derivation, B_{X_i} and $e(g,g)^{rk_{X_i}}$ are cancelled as they are equal.

• *Step 2.* Pairing Aggregation. Aggregate the F_i's and compute another value F as follows:

$$F = \prod_{i=0}^{n-1} F_i = \prod_{i=0}^{n-1} e(g,g)^{r\gamma_i s_i} e(g, h_{i,X_i})^{rt_i} = e(g,g)^{r\delta} e(g,g)^{rx'}$$

x' is some number (unknown) in $\mathbb{Z}_p$ such that

$$g^{x'} = \prod_{i=0}^{n-1} h_{i,X_i}^{t_i} = \prod_{\forall i,\ X_i \in S} h_{i,X_i}^{t_i}, \quad (t_i = 0,\ if\ X_i \notin S) \tag{5.4}$$

Therefore,

$$e(g,g)^{rx'} = \prod_{i=0}^{n-1} e(g, h_{i,X_i})^{rt_i}, \quad x' \in \mathbb{Z}_p \tag{5.5}$$

Lemma 5.1.1 *$x = x'$ if and only if the user's ID contains all the literals of the*

product, i.e., the user is in the set of intended users.

Proof: By Eq. (1) and Eq. (4), it is easy to see that if the user's ID contains all the literals of the product, $x = x'$. On the other hand, if the ID does not contain all the literals of the product, the user has negligible probability to output F in which $x' = x$ since t_i's are all random numbers.

• *Step 3.* Message Derivation. The user derives the M as follows

$$M' = \frac{\tilde{C}}{e(\check{C}, D)/F} = \frac{(M||MAC)e(g,g)^{\alpha s'}}{e(g,g)^{(\alpha s' + rs')}/e(g,g)^{r(\delta + x')}} = (M||MAC)e(g,g)^{r(x'-x)} \tag{5.6}$$

• *Step 4.* Message Verification. We assume M and MAC have fixed lengths of n_1 and n_2 bits respectively. To verify if she is in an intended recipient, each user takes the first n_1 bits and the remaining n_2 bits from M', denoted by M_1 and M_2 respectively, and checks if $M_2 = hash(M_1)$.

From Theorem 5.1.1 and Eq. (6), we know that only the intended users can recover $(M||MAC)$ correctly. Therefore, only for the intended users, does equation $M_2 = hash(M_1)$ hold and M_1 equal M.

5.1.3 Security Analysis

We analyze security of our scheme in terms of its correctness and fulfillment of our security goals. Correctness of our design can be shown by the following theorems:

Lemma 5.1.2 *A user can correctly decrypt M if and only if she holds all the intended attributes in the data access structure.*

Proof: To decrypt M, blind factor $e(g,g)^{\alpha s'}$ should be removed from $\tilde{C}$ as illustrated by Eq. (6). The only way to construct $e(g,g)^{\alpha s'}$ is to perform a bilinear

mapping between $g^{\beta s'}$ and $g^{(\alpha+r)/\beta}$, i.e., $e(g^{\beta s'}, g^{(\alpha+r)/\beta})$, which introduces another blind factor $e(g,g)^{rs'}$. As shown in Eq. (6), cancelling this blind factor requires $x = x'$ holds. By Theorem 5.1.1, $x = x'$ holds if and only if the user holds all the intended attributes in the data access structure.

Lemma 5.1.3 *Except for the authority, it is hard for any other parties to generate a valid secret key component D_i for attribute Att_{i,X_i} even if they have already known secret key components of other attributes.*

Proof: As defined in Section 5.1.2, $h_{i,\bar{X}_i} = g^{a_i}$ or g^{b_i}, where $a_i, b_i \in \mathbb{Z}_p$ are two independent random numbers. Without lose of generality, we assume $h_{i,\bar{X}_i} = g^{a_i}$. Therefore, the secret key component for attribute Att_{i,X_i} is $D_i = h^r_{i,\bar{X}_i} = g^{ra_i}$, where r and a_i are not known to any user. Any user not assigned the attribute Att_{i,X_i} only knows g^r, $g^{\beta r}$, and g^{rb_i}. Without knowing a_i and b_i, it is hard to generate g^{ra_i} given g^r, $g^{\beta r}$ and g^{rb_i} since a_i and b_i are independent. Therefore, this theorem holds.

From above lemmas, we can conclude that: (1) only the users with intended attributes can decrypt M; (2) Any user can not generate valid credentials for those attributes which are not assigned to him. Therefore, our design is correct.

Our proposed scheme meets the following security goals:

Data Confidentiality As is shown above, only intended users are able to decrypt the message M. Moreover, it can be shown that collusion does not help the unintended users decrypt the message since each user's SK is blinded by a blind factor r which is unique to each user.

Confidentiality of Access Structure First, we show that from the ciphertext the eavesdroppers are not able to derive the access structure information as follows. In the ciphertext, the intended attributes are secretly marked with a random number

$t_j \in \mathbb{Z}_p,\ j \in \mathbb{Z}_n$. Assume $C_{i,0}$ and $C_{i,1}$ of attribute i have the following form: $C_{i,0} = g^{k_0} h_{i,0}^{s_i+t_i}$ and $C_{i,1} = g^{k_1} h_{i,1}^{s_i}$. Since $h_{i,0}$ and $h_{i,1}$ are not publicly known, $C_{i,0}$ and $C_{i,1}$ appear as the form of $C_{i,0} = g^{k_0} g^{a_i(s_i+t_i)}$ and $C_{i,1} = g^{k_1} g^{b_i s_i}$ from the eavesdroppers' viewpoint. As a_i and b_i are randomly and independently chosen for any attribute i, $C_{i,0}$ and $C_{i,1}$ appear to be independent and random for the eavesdroppers. Therefore, they are not able to tell which one is marked and how many attributes are actually used in the access structure. Next, we show that the intended recipients are not able to derive the access structure information. This can be shown by observing the steps in the *Decryption* algorithm. As is shown in the steps, the user does not know if she is an intended receiver until she has aggregated the secret key components of all her attributes and decrypted the ciphertext in *Step* 4. Since her attributes take effort only when they are aggregated, the user can not tell which attributes grant or decline her access to the message M, nor how many attributes contribute to the access grant or declination. Therefore, any user, no matter authorized or unauthorized, can not tell, even partially, which or how many attributes are actually used in the access structure. Collusion does not help reveal this information because of the unique blind factor r in each user's SK. In addition, any user is not able to derive the number of associated attributes from the ciphertext size because it is constant in our proposed scheme.

Backward Secrecy For backward secrecy, any new user can not decrypt the messages sent before she joined the group. To achieve this goal, we can update the master key (MK) α before any new user joins. Similar to the process of delivering the message M, we can deliver $g^{\alpha'/\beta}$ to all users. Upon $g^{\alpha'/\beta}$, each user updates α as follows: $g^{(\alpha+r)/\beta} \cdot g^{\alpha'/\beta} = g^{(\alpha+\alpha'+r)/\beta}$. In this way, α is updated as $(\alpha + \alpha')$ securely. Member revocation can be realized in the same way except that we now update MK α to all users but those to be revoked.

5.1.4 Performance Evaluation

This section presents our evaluation results for the proposed scheme in terms of computation and communication loads as well as the storage load. We will present both numerical results and the experimental results. Finally, we give a brief discussion as well as the comparison between our proposed scheme and existing work. In the following part of this section, we assume the total number of attributes is n. We denote one scalar multiplication on the elliptic curve by an EXP, and one point addition operation by a MUL.

1. Numerical Results

Computation Load on the Authority The authority is responsible for execution of three algorithms: *Setup*, *KeyGen*, and *Encryption*. The main computation load of the algorithm *Setup* is caused by the calculation of $h_{i,0}$ and $h_{i,1}$, which involves $2n$ EXP operations in total. The algorithm *KeyGen* is responsible for computing the secret key SK. The main computation load is caused by the calculation of $\{D_i = h^r_{i,\bar{X}_i}\}_{\forall i \in \mathbb{Z}_n}$. It accounts for n EXP operations. *KeyGen* consumes $(n+3)$ EXP operations in total as the secret key components D, D', and D'' each accounts for one EXP. The main computation load of the algorithm *Encryption* comes from items $\{C_i = (g^{s_i}, C_{i,0}, C_{i,1})\}_{\forall i \in \mathbb{Z}_n}$ which represent $3n$ EXP operations[9]. In total the number of EXP operations required by *Encryption* is $(3n+3)$ since $\check{C}$ and $\hat{C}$ consume 1 and 2 EXP operations respectively. In addition, *Encryption* requires one one-way hash operation. The bilinear pairing operation required by the item $\tilde{C}$ can be ignored since we can pre-compute $e(g,g)^{\alpha}$. We do not count the integer field operations into our computation load because they account for a trivial

[9]Note that since the authority knows the master secret key MK, she can first calculate the exponents and then compute $C_{i,j}$, $j = 0|1$, by one EXP operation. We present the calculation of $C_{i,j}$ by two EXP's and one MUL in the algorithm just for clear description of our scheme. It is the similar case for $g^{s'}$.

	MUL	EXP	pairing
Setup [10]	0	2n	0
KeyGen[10]	0	n+3	0
Encryption[11]	0	3n+3	0
Decryption[11]	n	0	n+4

Table 5.2: Summary of cryptographic operations

part as compared to operations over elliptic curves.

Computation Load for Users Computation load on the user side mainly caused by the *Decryption* operation. According to our scheme description, following operations are required to decrypt a ciphertext: $(2n + 3)$ pairings and one hash. In most cases, a pairing operation is more expensive than an EXP operation or a MUL operation. Therefore, it is desirable to minimize the number of pairing operations. For this purpose, we can modify our decryption algorithm a little bit. We show this by expanding the calculation of F in *Step* 2 of *Decryption*:

$$\begin{aligned} F &= \prod_{i=0}^{n-1} F_i \\ &= \prod_{i=0}^{n-1} e(D_i, g^{s_i}) e(C_{i,X_i}, D') / B_{X_i} \\ &= e(\prod_{i=0}^{n-1} C_{i,X_i}, D') \cdot \prod_{i=0}^{n-1} (e(D_i, g^{s_i}) / B_{X_i}) \end{aligned} \tag{5.7}$$

Instead of computing $e(C_{i,X_i}, D')$ for each attribute as shown in *Step* 1, users can first multiply all C_{i,X_i} for all i in $\mathbb{Z}_n$, and then apply one pairing between D' and the product of the multiplication as is shown in Eq. (7). This revision saves $(n - 1)$ pairings and causes n MUL operations on the user side. The computation load of *Decryption* algorithm now becomes $(n+4)$ pairings, n MUL operations and

[10]Operations performed by the authority.
[11]Operations performed by users.

one hash. We do not take into count the computation of dividing B_{X_i}'s as they are trivial as compared to operations on elliptic curves. Table 5.2 concludes the main cryptographic operations required by our design[12].

Communication Load Our ciphertext is composed of four parts: $\tilde{C}$, $\check{C}$, $\{\hat{C}_j\}_{j=0,1}$, $\{C_i\}_{\forall\ i\in\mathbb{Z}_n}$. Each C_i has three parts: $g^{s_i}, C_{i,0}, C_{i,1}$. Therefore, the ciphertext contains $(3n+3)$ G_0 and 1 G_T group elements in total.

Storage Load for Users The main storage load of each user is for the secret key SK which represents $(n+3)$ G_0 group elements in total.

2. Experimental Results

In our experiment, we implement our algorithms based on the Pairing-Based Crypto (PBC) library [96]. We test our program on Ubuntu 8.04 with Intel Pentium D 3.40GHz CPU.

The Choice of Elliptic Curves In our experiment, we test on three types of elliptic curves: supersingular curves (type A), MNT curves (type D), and type F curves as is named in [96]. Type A curves enable fast pairing, while type D and type F curves require short group element. In particular, type F curves provides 1920-bit RSA security with only 160-bit group element. For type D curves, we test on two kinds of curves, namely $d159$ and $d201$. The detailed description of these curves can be found in [96].

To understand how the choice of elliptic curves affects our ciphertext size, it's necessary to discuss two kinds of groups characterized by bilinear maps: symmetric bilinear groups and asymmetric bilinear groups. The bilinear map in the former case has the form: $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_T$, while the latter has the form: $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, $\mathbb{G}_1 \neq \mathbb{G}_2$. Type A curves are characterized by having symmetric bilinear groups.

[12]Hash operations are not counted in the table since they just cause a negligible computational overhead

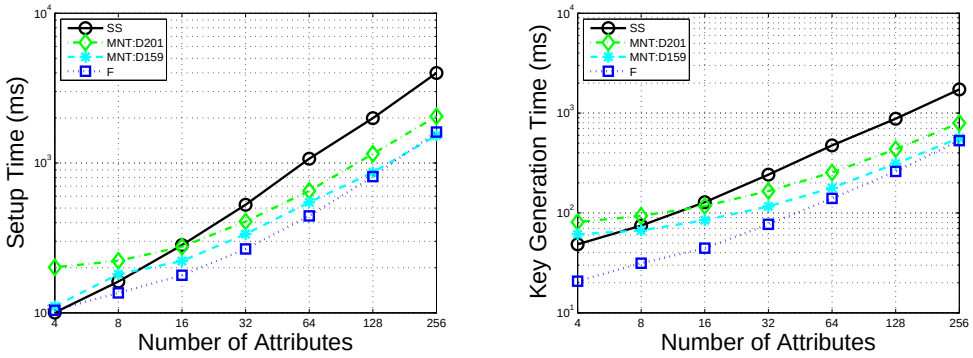
Type D and type F curves are characterized by having asymmetric bilinear groups. For fast pairing, we usually choose elliptic curves from symmetric bilinear groups with small embedding degrees. For example, type A curves have embedding degree of 2 and base field size of $\mathbb{G}_0$ is 512 bits. On the other hand, for short group size, we usually choose elliptic curves from asymmetric bilinear groups with high embedding degrees. For example, type F curves have embedding degree of 12. This turns out that the base field size of $\mathbb{G}_1$ is just 160 bits. The embedding degree of type D curves is 6. The base field size of $\mathbb{G}_1$ is just 170 bits for 1020-bit RSA security.

Ciphertext Size Our design uses symmetric bilinear groups by default. Therefore, the ciphertext size is $512(3n + 4)$ bits in the case of type A curves as both $\mathbb{G}_0$ and $\mathbb{G}_T$ can be represented by 512 bits. To achieve short ciphertext size, we can easily modify our design by using asymmetric bilinear groups. After this revision the ciphertext would just contain the following group elements: $(2n + 3)$ $\mathbb{G}_1$, n $\mathbb{G}_2$, and 1 $\mathbb{G}_T$. Table 5.3 gives a summary of our ciphertext size under the selected curves. As shown in Table 5.3, type A curves (supersingular) exhibits the longest ciphertext while type F curves provides the shortest ciphertext.

type A	type d159	type d201	type F
512(3n+4)	159(5n+6)	201(5n+6)	160(4n+9)

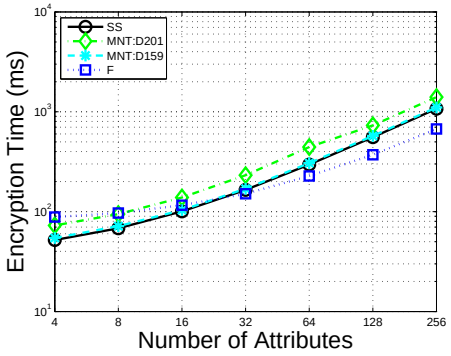
Table 5.3: Ciphertext sizes (bits) for different elliptic curves

Computation Load In our experiment, we run our algorithms over elliptic curves of type A (SS), $d159$, $d201$, and type F respectively. The number of attributes is chosen to be 4, 8, 16, 32, 64, 128, and 256. The experiment results are shown in Fig.5.1(a)-Fig.5.1(d). From these figures we can see that the computation load of our scheme is linear to the number of attributes, which verifies our numerical analysis results in Table 5.2. As is analyzed in Table 5.2, computation overhead of algorithm *Setup*, *KeyGen*, and *Encryption* is dominated by scalar multiplication

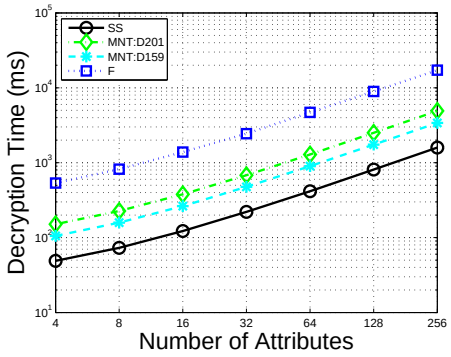


(a) Setup

(b) KeyGen



(c) Encryption



(d) Decryption

Figure 5.1: Experiment results on computation load.

operations on elliptic curves. Therefore, the elliptic curves with small base field size are more efficient than others. This is verified by Fig.5.1(a)-Fig.5.1(c) which show that type F curves are the most efficient. Type A (SS) curves are the least efficient as is shown in Fig.5.1(a) and Fig.5.1(b). In Fig.5.1(c), however, type A (SS) curves outperform type $d201$ curves and exhibit comparable encryption efficiency with $d159$ curves. This is because *Encryption* algorithm also involves n scalar multiplication operations on group $\mathbb{G}_2$. The base field size of $\mathbb{G}_2$ in the case of type F, $d159$, type A, and $d201$ are 320 bits, 477 bits, 512 bits, and 603 bits respectively. As the base field size of $\mathbb{G}_2$ is much larger than that of $\mathbb{G}_1$, these n scalar multiplication operations dominate the computation load of *Encryption* algorithm. Fig. 5.1(d) shows that type A (SS) curves have the best decryption performance while type F curves are

the worst. This coincides with our previous analysis.

Specifically, from these figures we can see that, in the case of 256 attributes our *Encrytion* algorithm takes about 0.7 seconds under type F curves and 1 second under type D and F curves. Our *Decryption* algorithm takes less than 2 second for type A curves in the case of 256 attributes. In the case of 64 attributes, *Decryption* takes about 0.3 seconds for type A curves and 1 second for type D curves.

3. Discussion and Comparison

From above numerical and experimental results, we can see that our scheme exhibits an acceptable computation load if the number of attributes are carefully chosen. Actually, the computation overhead for both ECC and bilinear pairing operations can be further reduced to the magnitude of μs under hardware implementations [52]. Moreover, as the computing power of processors is increasing rapidly, computation overload should not be a problem. What actually matters is the communication overload as bandwidth is a limited resource under environments such as wireless networks. Fortunately, the communication overload of our scheme grows linearly to the number of attributes only. Since attributes are shared by unlimited number of users, communication overload of our scheme can be well controlled even in the case of large-scale application scenarios. As a matter of fact, even in large-scale systems, the number of attributes required could be relatively small. To evaluate the performance of our scheme, we can compare it with current work. To the best of our knowledge, the only existing work that addresses the similar issue is proposed by Barth et al. [45]. In that scheme, identities of the recipients are protected by encrypting the message using every user's public key. The computational load as well as the ciphertext size grow linear to the group size. Assume we are using 1024 bit RSA, the ciphertext size would be $1024N$ bits, where N is the total

number of users. This represent a huge communication load in large-scale applications and not applicable for large-scale application scenarios. Moreover, our scheme also protects the number of the intended users while [45] does not.

5.2 Our Construction under the XDH assumption

This section proposes our construction under the XDH assumption, which is believed to hold for some MNT elliptic curves. This construction is more efficient than the previous one.

5.2.1 Scheme Description

Instead of building a CP-ABE scheme from scratch we construct our privacy-preserving CP-ABE by enhancing Cheung's construction [9]. In this work, we also follow the same definition of notation as [9]. For completeness, we present them as follows: The set of attributes are defined as $N := 1, \ldots, n$ for some natural number n. Attribute i and their negations $\neg i$ are referred to as *literals*. Let I denote the set of attributes that are needed for decryption. The scheme considers access structures that consist of a single AND gate whose inputs are literals, denoted by $\bigwedge_{i \in I} \underline{i}$, where every $\underline{i}$ is a literal (i.e., i or $\neg i$).

SETUP. This algorithm selects bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p with generator g_1 and g_2 respectively. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ is defined on them. Next, it chooses random exponents $y, t_1, \ldots, t_{2n} \in \mathbb{Z}_p$. The public key is

published as:

$$PK = (e, g_1, g_2, Y, T_1, \ldots, T_{2n})$$

where $Y = e(g_1, g_2)^y$, $\forall i \in \mathbb{Z}_{2n} : \ T_i = g_1^{t_i}$. The master secret key is $MK = (y, t_1, \ldots, t_{2n})$.

In our construction, each attribution only has two occurrences: positive and negative. *don't care* element is discarded, while it is a key element in the original construction.

ENCRYPT. Given a message $M \in \mathbb{G}_T$ and an AND gate $W = \bigwedge_{i \in I} \underline{i}$, the ciphertext is output as $CT = (\tilde{C}, \hat{C}, \{C_{i,0}, C_{i,1} | i \in N\})$, where $\tilde{C} = M \cdot Y^s$, $\hat{C} = g^s$, and s is a random number in $\mathbb{Z}_p$.

For each $i \in I$, $C_{i,0}$ and $C_{i,1}$ are computed as follows.

- if $\underline{i} = i$, $C_{i,0} = T_i^s$, $C_{i,1} = T_{n+i}^x$.
- if $\underline{i} = \neg i$, $C_{i,0} = T_i^x$, $C_{i,1} = T_{n+i}^s$.

  x is a random number in $\mathbb{Z}_p$.

For each $i \notin I$, $C_{i,0} = T_i^s$ and $C_{i,1} = T_{n+i}^s$.

KEYGEN. Let S denote the input attribute set. Every $i \notin S$ is considered a negative attribute. The secret key is defined as $SK = (\hat{D}, \{D_i | i \in N\})$, where $\hat{D} = g_2^{y-r}$, $r = \sum_{i=1}^{n} r_i$, r_i is randomly selected from $\mathbb{Z}_p$. For each $i \in N$, $D_i = g_2^{\frac{r_i}{t_i}}$ if $i \in S$; otherwise, $D_i = g_2^{\frac{r_i}{t_{n+i}}}$.

DECRYPT. Suppose the input ciphertext is of form $CT = (\tilde{C}, \hat{C}, \{C_{i,0}, C_{i,1} | i \in N\})$. Let $SK = (\hat{D}, \{D_i | i \in N\})$. For each $i \in N$, if the user's attribute is positive,

then

$$F_i = e(C_{i,0}, D_i) = e(g_1^{t_i \cdot s}, g_2^{\frac{r_i}{t_i}} = e(g_1, g_2)^{r_i \cdot s})$$

If the user's attribute is negative, then

$$F_i = e(C_{i,1}, D_i) = e(g_1^{t_{n+i} \cdot s}, g_2^{\frac{r_i}{t_{n+i}}} = e(g_1, g_2)^{r_i \cdot s})$$

Decrypt finishes as follows: $M = \frac{\tilde{C}}{Y^s} = \frac{\tilde{C}}{e(g_1, g_2)^{y \cdot s}}$, where

$$e(g_1, g_2)^{y \cdot s} = e(g_1^s, g_2^{y-r}) \cdot e(g_1, g_2)^{r \cdot s} = e(\hat{C}, \hat{D}) \cdot \prod_{i=1}^{n} F_i.$$

Above equations demonstrate how an intended user can decrypt the ciphertext. If the user is not the intended recipient, there is at least one attribute for which the user gets F_i with the form $e(g_1, g_2)^{r_i \cdot x}$. Therefore, she can not calculate $e(g_1, g_2)^{y \cdot s}$ as shown in the above equation.

5.2.2 Scheme Analysis

Ciphertext in our construction does not include the access policy. In decrypt algorithm, the user uses all her attributes to decrypt the ciphertext. If the user's i^{th} attribute is positive while $C_{i,0}$ has the form T_i^x, this user can not decrypt the ciphertext. However, because the user can not distinguish between T_i^x and T_i^s according to XDH assumption, she is not able to know which attributes are desired by the encryptor. Therefore, she can not derive any information about the access policy. For the same reason, an intended user only knows if she can decrypt the ciphertext while not knowing which attributes grant her the access. Therefore, the access policy is hidden to all the users.

5.3 Summary

In this chapter, we addressed an important problem of privacy-preserving construction of ABE. We proposed two privacy-enhanced CP-ABE schemes in which data access structures are well protected from both the untrusted servers and all the users even under powerful attacks, e.g., colluding attacks. Numerical and experimental results show that our scheme is suitable for large-scale applications since its complexity is just linear to the number of attributes rather than the number of users. Our proposed schemes are secure under the generic group model and the XDH assumptions respectively.

Chapter 6

Secure Data Sharing with ABE in Cloud Computing

Cloud Computing is a promising next-generation IT architecture which provides elastic and unlimited resources, including storage, as services to cloud users. In Cloud Computing cloud users and cloud service providers are almost certain to be from different trust domains. It turns out that on one hand sensitive data should be encrypted before uploading to cloud servers; on the other hand, a secure user-enforced data access control mechanism must be provided before cloud users have the liberty to outsource sensitive data to the cloud for storage. Similar to any untrusted storage case, we can resolve the issue using a cryptographic-based data access control mechanism as discussed in Chapter 1. In doing so, we need to address challenge issues such as fine-grained access control with scalability, user dynamics and etc. In addition to these, another main challenge pertained to Cloud Computing is system efficiency. In Cloud Computing, Cloud users (including both data owners and users[13]) could access the system via various low-end devices such as mobile

[13]Here users refer to data consumers.

phones, which do not have much computation power. Therefore, the proposed access control mechanism should be efficient enough in the sense that the computation load addressed on both the data owner and data consumers should be affordable to these low-end devices.

Keeping these challenges in mind, in this chapter we propose a cryptographic-based data access control mechanism for Cloud Computing with ABE. In this work, we address the issue of user revocation by applying our technique in Chapter 3 to Cloud Computing. By introducing a dummy attribute into the system, we enable users to delegate most data decryption operations to cloud servers and reduce the computation load on users to a constant complexity. We also greatly reduce the computation load on cloud servers through using the technique of lazy re-encryption [14]. Compared to previous work [11, 14, 29, 30], our scheme provides better scalability when providing fine-grained data access control since the complexity of most system operations in our scheme is just linear to the number of attributes rather than the number of users/data files.

6.1 Models and Assumptions

6.1.1 System Models

Similar with [31], we assume that the system is composed of the following parties: Cloud Users, Cloud Servers, and maybe Third Party Auditor. Cloud users include both the data owner and data consumers. The data owner stores his encrypted data files on Cloud Servers either for sharing or for personal use. To access data files shared by the data owner, data consumers need to download them from the Cloud Servers and then decrypt. Cloud users, be it the data owner or data consumers, will not be always online. They come online just on the necessity basis. For simplicity, we

assume that the only access privilege for data consumers is data file read. Extending our proposed scheme to support data file writing is trivial by asking the data writer to sign the new file on each update as [29] does. As we did in previous chapters, in this chapter we also call data consumers by *users* for brevity. Cloud Servers are always online and operated by Cloud Service Provider (CSP). They are assumed to have abundant storage capacity and computation power. The Third Party Auditor, if any, is an online party which is used for auditing every file access event. In addition, we also assume that the data owner can not only store data files but also run his own code on Cloud Servers to manage his data files. This assumption coincides with the unified ontology of cloud computing which is recently proposed by Youseff et al. [32].

6.1.2 Security Models

In this work, we just consider Honest but Curious Cloud Servers as [11] does. That is, Cloud Servers will follow our proposed protocol in general, but try to find out as much secret information as possible based on their inputs. More specifically, we assume Cloud Servers are more interested in contents of data files and user access privilege information than other secret information. Cloud Servers might collude with a small number of malicious users for the purpose of harvesting the data file contents when it is highly beneficial. Communication channel between Clients and Cloud Servers are assumed to be secured under existing security protocols such as SSL. Users would try to access data files either within or out of the scope of their access privileges. To achieve this goal, unauthorized users may work independently or cooperatively. In addition, each party is preloaded with a public/private key pair and the public key can be easily obtained by other parties when necessary.

6.1.3 Design Goals

Our main design goal is to help the data owner enforce fine-grained access control over data in large-scale data centers outsourced to Cloud Servers. In terms of fine-grained access control, we want to enable the data owner to enforce a unique access structure on each user whenever necessary, which precisely designates the type/set of files that the user is allowed to access. We also want to prevent Cloud Servers from learning either the plaintexts of data files or user access privilege information. In addition, the proposed scheme should be able to support user dynamics and achieve security goals such as user accountability. All these security goals should be achieved efficiently in the sense that the computation load should be affordable to Cloud users with low-end portable devices.

6.2 Our Proposed Scheme

6.2.1 Main Idea

In order to achieve secure, scalable and fine-grained data sharing on outsourced data in the cloud, we utilize and uniquely combine the following three advanced cryptograhphic techniques: KP-ABE, Proxy Re-Encryption (PRE) and lazy re-encryption. More specifically, we associate each data file with a set of attributes, and assign each user an expressive access structure which is defined over these attributes. To enforce this kind of access control, we utilize KP-ABE to escort data encryption keys of data files. Such a construction enables us to immediately enjoy fine-grainedness of access control. However, this construction, if deployed alone, would introduce heavy computation overhead and cumbersome online burden towards the data owner, as he is in charge of all the operations of data/user manage-

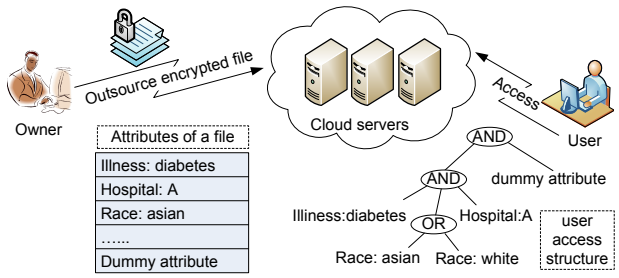


Figure 6.1: An example case in the healthcare scenario

ment. Specifically, such an issue is mainly caused by the operation of user revocation, which inevitabily requires the data owner to re-encrypt all the data files accessible to the leaving user, or even needs the data owner to stay online to update secret keys for users. To resolve this challenging issue and make the construction suitable for the cloud computing, we apply our technique in Chapter 3 and uniquely combine PRE with KP-ABE to enable the data owner to delegate most of the computation-intensive operations to Cloud Servers without disclosing the underlying plaintexts. In order for saving the computation load on users, we enable users to delegate most computation operations for decryption to cloud servers through the use of a dummy attribute. Such a construction allows both the data owner and users to participate in the system with a minimal overhead in terms computation effort and online time, and fits well into the cloud environment. For further reducing the computation overhead on Cloud Servers and thus saving the data owner's investment, we take advantage of the lazy re-encryption technique and allow Cloud Servers to "aggregate" computation tasks of multiple system operations. Notably, the computation complexity on Cloud Servers is just proportional to the number of system attributes rather than the number of users or data files in the system. Scalability and efficiency is thus achieved. In addition, user accountability can also be achieved by using our technique in Chapter 4.

6.2.2 Definition and Notation

In our scheme, each data file is assigned a set of attributes which are meaningful and necessary for access control. Different data files can have a subset of overlapped attributes. Each attribute is associated with a version number for the purpose of attribute update as we will discuss later. Cloud Servers keep an attribute history list AHL which records the version evolution history of each attribute and PRE keys used. In addition to these meaningful attributes, we also define one dummy attribute, denoted by symbol Att_D for the purpose of key management as well as computation delegation. Att_D is required to be included in every data file's attribute set and will never be updated. The access structure of each user is implemented by an access tree. Interior nodes of the access tree are threshold gates. Leaf nodes of the access tree are associated with data file attributes. For the purpose of key management and computation delegation, we require the root node to be an AND gate (i.e., n-of-n threshold gate) with one child being the leaf node which is associated with the dummy attribute, and the other child node being any threshold gate. The dummy attribute will not be attached to any other node in the access tree. Fig.6.1 illustrates our definitions by an example. In addition, Cloud Servers also keep a user list UL which records IDs of all the valid users in the system. Fig.6.2 gives the description of notation to be used in our scheme.

6.2.3 Scheme Description

We present our proposed scheme in two levels: *System Level* and *Algorithm Level*. At system level, we are interested in the following high level operations: *System Setup*, *File Creation*, *User Grant*, and *User Revocation*, *File Access*, *File Deletion*, and the interaction between involved parties. At algorithm level, we focus on the

Notation	Description
PK, MK	system public key and master key
T_i	public key component for attribute i
t_i	master key component for attribute i
SK	user secret key
sk_i	user secret key component for attribute i
E_i	ciphertext component for attribute i
I	attribute set assigned to a data file
DEK	symmetric data encryption key of a data file
P	user access structure
L_P	set of attributes attached to leaf nodes of P
Att_D	the dummy attribute
UL	the system user list
AHL_i	attribute history list for attribute i
$rk_i^{(k)}$	proxy re-key to update attribute i from version $k-1$ to version k
$\delta_{O,X}$	the data owner's signature on message X

Figure 6.2: Notation used in our scheme description

mathematical implementation of algorithms that are invoked by system level operations.

1. System Level Operations

System level operations in our proposed scheme are designed as follows.

System Setup In this operation, the data owner chooses a security parameter κ and calls the algorithm level interface $ASetup(\kappa)$, which outputs the system public parameter PK and the system master key MK. The data owner then signs each component of PK and sends PK along with these signatures to Cloud Servers.

File Creation Before uploading a file to Cloud Servers, the data owner processes the data file as follows.

- select a unique ID for this data file;
- randomly select a symmetric data encryption key $DEK \xleftarrow{R} \mathcal{K}$, where $\mathcal{K}$ is the

key space, and encrypt the data file using DEK;

- define a set of attribute I for the data file and encrypt DEK with I using KP-ABE, i.e., $(\tilde{E}, \{E_i\}_{i \in I}) \leftarrow AEncrypt(I, DEK, PK)$.

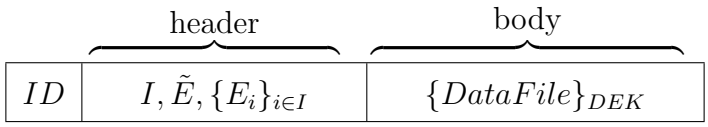


Figure 6.3: Format of a data file stored on the cloud

Finally, each data file is stored on the cloud in the format as is shown in Fig.6.3.

User Grant When a new user wants to join the system, the data owner needs to assign an access structure and the corresponding secret key to this user as follows.

- assign the new user a unique identity w and an access structure P;
- generate a secret key SK for w, i.e., $SK \leftarrow AKeyGen(P, MK)$;
- encrypt the tuple $(P, SK, PK, \delta_{O,(P,SK,PK)})$ with user w's public key, denoting the ciphertext by C;
- send the tuple $(T, C, \delta_{O,(T,C)})$ to Cloud Servers, where T denotes the tuple $(w, \{j, sk_j\}_{j \in L_P \setminus Att_D})$, and L_P represents the leaf nodes on the access structure P. The tuple T contains all the secret key components in SK but the one for the dummy attribute Att_D.

On receiving the tuple $(T, C, \delta_{O,(T,C)})$, Cloud Servers processes as follows.

- verify $\delta_{O,(T,C)}$ and proceed if correct;
- store T in the system user list UL;
- forward C to the user.

// to revoke user v

// stage 1: attribute update.

The Data Owner		**Cloud Servers**
1. $D \leftarrow AMinimalSet(P)$, where P is v's access structure;		remove v from the system user list UL;
2. for each attribute i in D $(t'_i, T'_i, rk_i^{(k)}) \leftarrow AUpdateAtt(i, MK)$;	$\xrightarrow{AT}$	for each attribute $i \in D$ store $(i, T'_i, \delta_{O,(i,T'_i)})$;
3. $AT = (v, D, \{i, T'_i, \delta_{O,(i,T'_i)}, rk_i^{(k)}\}_{i \in D})$.		add $rk_i^{(k)}$ to i's history list AHL_i.

// stage 2: data file and key update.

Cloud Servers		**User**(u)
1. on receiving RQ, proceed if $u \in UL$;		
2. get the tuple $(u, \{j, sk_j\}_{j \in L_P \backslash Att_D})$;		1. generate data file access request RQ;
for each attribute $j \in L_P \backslash Att_D$ $sk'_j \leftarrow AUpdateSK(j, sk_j, AHL_j)$;	$\xleftarrow{RQ}$	2. wait for the response from Cloud Servers;
for each requested file f in RQ for each attribute $k \in I_f$	$\xrightarrow{RP}$	3. on receiving RP, verify each $\delta_{O,(j,T'_j)}$ and sk'_j; proceed if all correct;
$E'_k \leftarrow \mathrm{A}UpdateAtt4File(k, E_k, AHL_k)$;		4. replace each sk_j in SK with sk'_j;
3. $RP = (\{j, sk'_j, T'_j, \delta_{O,(j,T'_j)}\}_{j \in L_P \backslash Att_D}, FL)$.		5. decrypt each file in FL with SK.

Figure 6.4: Description of the process of user revocation

On receiving C, the user first decrypts it with his private key. Then he verifies the signature $\delta_{O,(P,SK,PK)}$. If correct, he accepts (P, SK, PK) as his access structure, secret key, and the system public key.

As described above, Cloud Servers store all the secret key components of SK except for the one corresponding to the dummy attribute Att_D. Such a design allows Cloud Servers to update these secret key components during user revocation as we will describe soon. As there still exists one undisclosed secret key component (the one for Att_D), Cloud Servers can not use these known ones to correctly decrypt ciphertexts. Actually, these disclosed secret key components, if given to any unauthorized user, do not give him any extra advantage in decryption as we will

discuss in our security analysis. For the same reason, with these secret key components Cloud Servers can even help users execute most bilinear pairing operations pertained to data decryption as we will discuss.

User Revocation We start with the intuition of the user revocation operation as follows. Whenever there is a user to be revoked, the data owner first determines a minimal set of attributes without which the leaving user's access structure will never be satisfied. Next, he updates these attributes by redefining their corresponding system master key components in MK. Public key components of all these updated attributes in PK are redefined accordingly. Then, he updates user secret keys accordingly for all the users except for the one to be revoked. Finally, DEKs of affected data files are re-encrypted with the latest version of PK. The main issue with this intuitive scheme is that it would introduce a heavy computation overhead for the data owner to re-encrypt data files and might require the data owner to be always online to provide secret key update service for users. To resolve this issue, we apply our technique in Chapter 3 and combine the technique of proxy re-encryption with KP-ABE to delegate tasks of data file re-encryption and user secret key update to Cloud Servers. More specifically, we divide the user revocation scheme into two stages as is shown in Fig.6.4.

In the first stage, the data owner determines the minimal set of attributes, redefines MK and PK for involved attributes, and generates the corresponding PRE keys. He then sends the user's ID, the minimal attribute set, the PRE keys, the updated public key components, along with his signatures on these components to Cloud Servers, and can go off-line again. Cloud Servers, on receiving this message from the data owner, remove the revoked user from the system user list UL, store the updated public key components as well as the owner's signatures on them, and record the PRE key of the latest version in the attribute history list AHL for each

updated attribute. AHL of each attribute is a list used to record the version evolution history of this attribute as well as the PRE keys used as shown in figure 6.5. Every attribute has its own AHL which is represented by the corresponding column in figure 6.5. With AHL, Cloud Servers are able to compute a single PRE key that enables them to update the attribute from any historical version to the latest version. This property allows Cloud Servers to update user secret keys and data files in the "lazy" way as follows. Once a user revocation event occurs, Cloud Servers just record information submitted by the data owner as is previously discussed. Only if there is a file data access request from a user, do Cloud Servers check the attribute version information and when necessary re-encrypt the requested files and update the requesting user's secret key. This statistically saves a lot of computation overhead since Cloud Servers are able to "aggregate" multiple update/re-encryption operations into one if there is no data access request occurring across multiple successive user revocation events.

Version #	**attr 1**	**attr 2**	$\cdots$	**attr** n
k	$rk_1^{(k)}$	$rk_2^{(k)}$	$\cdots$	$rk_n^{(k)}$
$k-1$	$rk_1^{(k-1)}$	$rk_2^{(k-1)}$	$\cdots$	$rk_n^{(k-1)}$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$
$k-W$	$rk_1^{(k-W)}$	$rk_2^{(k-W)}$	$\cdots$	$rk_n^{(k-W)}$

Figure 6.5: Attribute History Lists (AHLs)

One issue is how many versions should be recorded in the AHLs. In this work, we leave this number as a system parameter, denoted by W. In practice, each re-key in AHLs could be several hundred bits, e.g., 160 bits. The total storage complexity for AHLs also depends on the number of attributes in the system. In Cloud Computing, storage would not be an issue and the system has the ability to record a large number of versions. More importantly, as the version numbers are

ordered, query over this table should be very efficient.

File Access This is also the second stage of user revocation. In this operation, Cloud Servers respond user request on data file access, and update user secret keys and re-encrypt requested data files if necessary. As is depicted in Fig. 6.4, Cloud Servers first verify if the requesting user is a valid system user in UL. If true, they update this user's secret key components to the latest version and re-encrypt the DEKs of requested data files using the latest version of PK. Notably, Cloud Servers will not perform update/re-encryption if secret key components/data files are already of the latest version. Finally, Cloud Servers send updated secret key components as well as ciphertexts of the requested data files to the user. On receiving the response from Cloud Servers, the user first verifies if the claimed version of each attribute is really newer than the current version he knows. For this purpose, he needs to verify the data owner's signatures on the attribute information (including the version information) and the corresponding public key components, i.e., tuples of the form (j, T'_j) in Fig. 6.4. If correct, the user further verifies if each secret key component returned by Cloud Servers is correctly computed. He verifies this by computing a bilinear pairing between sk'_j and T'_j and comparing the result with that between the old sk_j and T_j that he possesses. If verification succeeds, he replaces each sk_j of his secret key with sk'_j and update T_j with T'_j. Finally, he decrypts data files by first calling $ADecrypt(P, SK, E)$ to decrypt DEK's and then decrypting data files using DEK's.

In the last step of the above process, the user can also choose to delegate the major part of data decryption operation $ADecrypt(P, SK, E)$ to cloud servers. This is because in KP-ABE, data decryption mainly involves following steps: 1) select a set of attributes from the header of the ciphertext (cf. figure 6.3.) that satisfy the user's access structure; 2) do bilinear pairing between the ciphertext component

E_i and secret key component sk_i for each attribute i in the selected attribute set, and compute a blind factor with pairing results; 3) cancel the blind factor in the ciphertext component $\tilde{E}$ and recover DEK. In the decryption process, step 2) represents the most computation-intensive part since a number of bilinear pairing operations are needed, and the number is related to the complexity of the user access structure. In our scheme, cloud servers possesses all the user secret key components except for the one for the dummy attribute Att_D. This makes it possible for the cloud servers to do the bilinear pairing operations in step 2) on behalf of the user. After having obtained the pairing results from the cloud servers, the user just need to do another bilinear pairing for the dummy attribute Att_D, compute the blind factor, and recover DEK following step 3). If the user would like to disclose his access structure to cloud servers, the later can even help the user compute the blind factor. It turns out that the user just needs to perform three operations: one bilinear pairing for the dummy attribute and two multiplications over an integer field (one for computing the blind factor and the other for recovering DEK). Such a computation task can be efficiently executed on contemporary portable devices such as mobile phones.

File Deletion This operation can only be performed at the request of the data owner. To delete a file, the data owner sends the file's unique ID along with his signature on this ID to Cloud Servers. If verification of the owner's signature returns true, Cloud Servers erase the data file from the storage space.

2. Algorithm level operations

Algorithm level operations include eight algorithms: $ASetup$, $AEncrypt$, $AKeyGen$, $ADecrypt$, $AUpdateAtt$, $AUpdateSK$, $AUpdateAtt4File$, and $AMinimalSet$. As the first four algorithms are just the same as $Setup$, $Encryption$, $Key\ Generation$,

and *Decryption* of the standard KP-ABE respectively, we focus on our implementation of the last four algorithms. Fig.6.6 depicts two of the four algorithms.

$AUpdateAtt(i, MK)$
  // assume current version of attribute i is $k-1$;
  randomly pick $t'_i \overset{R}{\leftarrow} \mathbb{Z}_p$;
  compute $T'_i \leftarrow g^{t'_i}$, and $rk_i^{(k)} \leftarrow \frac{t'_i}{t_i}$;
  output t'_i, T'_i, and $rk_i^{(k)}$.

$AUpdateAtt4File(i, E_i, AHL_i)$
  if i has the latest version, exit;
  search AHL_i and locate the old version of i;
  // assume the old version of attribute i is j
  // and its latest version is k.
  $rk_i^{(j)\rightarrow(k)} \leftarrow rk_i^{(j+1)} \cdot rk_i^{(j+2)} \cdots rk_i^{(k)} = \frac{t_i^{(k)}}{t_i^{(j)}}$;
  compute $E_i^{(k)} \leftarrow (E_i)^{rk_i^{(j)\rightarrow(k)}} = g^{(t_i^{(k)})s}$;
  output $E_i^{(k)}$.

Figure 6.6: Pseudo-code of algorithm level algorithms

AUpdateAtt This algorithm updates an attribute to a new version by redefining its system master key and public key component. It also outputs a proxy re-encryption key between the old version and the new version of the attribute.

AUpdateAtt4File This algorithm translates the ciphertext component of an attribute i in the header of a data file from an old version into the latest version. It first checks the attribute history list of this attribute and locates the position of the old version. Then it multiplies all the PRE keys between the old version and the latest version and obtains a single PRE key. Finally it apply this single PRE key to the ciphertext component E_i and returns $E_i^{(k)}$ which coincides with the latest definition of attribute i.

AUpdateSK This algorithm translates the secret key component of attribute i in the user secret key SK from an old version into the latest version. Its implementation is similar to *AUpdateAtt4File* except that, in the last step it applies $(rk_i^{(j)\rightarrow(k)})^{-1}$ to

SK_i instead of $rk_i^{(j)\to(k)}$. This is because $t_i^{(j)}$ [13] is the denominator of the exponent part of SK_i while in E_i it is a numerator.

AMinimalSet This algorithm determines a minimal set of attributes without which an access tree will never be satisfied. For this purpose, it constructs the conjunctive normal form (CNF) of the access tree, and returns attributes in the shortest clause of the CNF formula as the minimal attribute set.

6.2.4 Discussion

In our proposed scheme, we exploit the technique of hybrid encryption to protect data files, i.e., we encrypt data files using symmetric DEKs and encrypt DEKs with KP-ABE. Using KP-ABE, we are able to enjoy fine-grained data access control and efficient operations such as file creation/deletion and new user grant. To resolve the challenging issue of user revocation, we combine the technique of proxy re-encryption with KP-ABE and delegate most of the burdensome computational task to Cloud Servers. We achieve this by letting Cloud Servers keep a partial copy of each user's secret key, i.e., secret key components of all but one (dummy) attributes. When the data owner redefines a certain set of attributes for the purpose of user revocation, he also generates corresponding proxy re-encryption keys and sends them to Cloud Servers. Cloud Servers, given these proxy re-encryption keys, can update user secret key components and re-encrypt data files accordingly without knowing the underlying plaintexts of data files. This enhancement releases the data owner from the possible huge computation overhead on user revocation. The data owner also does not need to always stay online since Cloud Servers will take over the burdensome task after having obtained the PRE keys. To further save computation overhead of Cloud Servers on user revocation, we use the technique of lazy

[13] The superscript (j) means t_i is defined for attribute i of version j.

re-encryption and enable Cloud Servers to "aggregate" multiple successive secret key update/file re-encryption operations into one, and thus statistically save the computation overhead. Moreover, we reduce the computation load on user side in terms of data decryption to a constant complexity by letting them securely delegate most bilinear pairing operations in data decryption to cloud servers.

6.3 Analysis of Our Proposed Scheme

6.3.1 Security Analysis

We first analyze security properties of our proposed scheme, starting with the following immediately available properties.

Fine-grainedness of Access Control In our proposed scheme, the data owner is able to define and enforce expressive and flexible access structure for each user. Specifically, the access structure of each user is defined as a logic formula over data file attributes, and is able to represent any desired data file set.

User Secret Key Accountability This property can be immediately achieved by using the enhanced construction of KP-ABE in Chapter 4.

Data Confidentiality We analyze data confidentiality of our proposed scheme by comparing it with an intuitive scheme in which data files are encrypted using symmetric DEKs, and DEKs are direclty encrypted using standard KP-ABE. In this intuitive scheme just ciphertexts of data files are given to Cloud Servers. Assuming the symmetric key algorithm is secure, e.g., using standard symmtric key algorithm such as AES, security of this intuitive scheme is merely relied on the security of KP-ABE. Actually, the standard KP-ABE is provably secure under the attribute-based Selective-Set model [12] given the Decisional Bilinear Diffie-Hellman (DBDH) problem is hard. Therefore, the intuitive scheme is secure under the same

model. Our goal is to show that our proposed scheme is as secure as the intuitive scheme. As is compared to the intuitive scheme, our scheme discloses the following extra information to Cloud Servers: a partial set of user secret key components (except for the one for the dummy attribute which is required for each decryption), and the proxy re-encryption keys. Based on this observation, we sketch the security proof of our proposed scheme using a series of games as follows.

Game 0: This is the security game of the intuitive scheme.

Game 1: The difference between this game and *Game 0* is that, in this game more than one (MK, PK) pairs are defined, and the adversary is given the PK's as well as the secret keys for each access structure he submits. In addition, the adversary is also given the proxy re-encryption keys (between any two (MK, PK) pairs).

Game 2: This game is for our proposed scheme. The only difference between this game and *Game 1* is that, in this game the partial set of user secret key components are disclosed to the adversary.

Lemma 6.3.1 *The advantage of the adversary in Game 1 is the same as that in Game 0.*

Proof: Our first goal is to show that, if there is a polynomial time algorithm $\mathcal{A}$ that wins the semantic security game of *Game 1* with non-negligible advantage, we can use it to build a polynomial time algorithm $\mathcal{B}$ to win the semantic security game of *Game 0*, i.e., the game under the attribute-based Selective-Set model. In the semantic security game, the adversary submits two equal length challenge message m_0 and m_1. The challenger flips a random coin $b \leftarrow \{0, 1\}$ and encrypts m_b. The challenge cihpertext E is then given to the adversary. The adversary is asked to output his guess b' of the random coin b. If $b' = b$ the adversary wins.

During this game, the adversary is given public parameters and allowed to query many user secret keys except for the one for the challenge ciphertext. Assuming the algorithm $\mathcal{A}$ (i.e., the adversary) can win the semantic security game, i.e., $b \leftarrow \mathcal{A}(E_A, \{PK_i\}_{1\le i\le k}, \{SK_j\}_{1\le j\le q_S}, \{rk\})$, with non-negligible advantage, where $\{PK_i\}$ and $\{SK_j\}$ denotes the set of all PK's and the set of all the secret keys given to $\mathcal{A}$ respectively, $\{rk\}$ representing the set of all the proxy re-encryption keys, q_S denoting the number of secret key queries, and k representing the number of PK's, A polynomial time algorithm $\mathcal{B}$ can be built as is shown in Fig.6.7. Therefore, the advantage of the adversary in *Game 1* is not higher than that in *Game 0*. However, the advantage of the adversary in *Game 1* can not be lower than that in *Game 0* since the adversary is given more information in *Game 1* than in *Game 0*. Therefore, the advantages in the two games are the same.

$\mathcal{B}(E_B, PK, \{SK'_j\}_{1\le j\le q_S})$
assume $PK = (Y, T_1, T_2, \cdots, T_N)$;
$E_B = (I, \tilde{E}, \{E_j\}_{j\in I})$;
$SK'_j = \{sk_{j,i}\}_{i\in L}$ for all $j \in \{1, \cdots, q_S\}$;
for u from 1 to k:
for v from 1 to N:
random choose $r_{uv} \xleftarrow{R} \mathbb{Z}_p$;
if $u > 1$, $rk_{u-1\leftrightarrow u} = \frac{r_{(u-1)v}}{r_{uv}}$;
add $rk_{u-1\leftrightarrow u}$ to {rk};
$PK_u = (Y, T_1^{r_{u1}}, T_2^{r_{u2}}, \cdots, T_N^{r_{uN}})$;
$E_A \leftarrow (I, \tilde{E}, \{E_j^{r_{uj}}\}_{j\in I})$, where $u \xleftarrow{R} \{1, \cdots, k\}$;
$SK_j = \{(sk_{j,i})^{\frac{1}{r_{ui}}}\}_{i\in L}$, u is the same as above;
$b' \leftarrow \mathcal{A}(E_A, \{PK_i\}_{1\le i\le k}, \{SK_j\}_{1\le j\le q_S}, \{rk\})$.

Figure 6.7: Construction of algorithm $\mathcal{B}$ from $\mathcal{A}$

Lemma 6.3.2 *The advantage of the adversary in Game 2 is the same as that in Game 1.*

Proof: As is described, the extra information disclosed to the adversary in *Game 2*

are the partial user secret keys. These partial user secret keys are actually equivalent to the secret keys queried by the adversary in *Game 1*. Therefore, the view of the adversary in the two games are the same. This proves this lemma.

According to the above two lemmas, we can conclude that our proposed scheme is as secure as the intuitive scheme, which is provably secure. This proves data confidentiality of our proposed scheme, even under collusion attacks between Cloud Servers and malicious users.

6.3.2 Performance Analysis

This section numerically evaluates the performance of our proposed scheme in terms of the computation overhead introduced by each operation as well as the ciphertext size.

Computation Complexity We analyze the computation complexity for the following six operations: *system setup*, *new file creation*, *file deletion*, *new user grant*, *user revocation*, and *file access*.

System Setup In this operation, the data owner needs to define underlying bilinear groups, and generate PK and MK. For KP-ABE, the main computation overhead for the generation of PK and MK is introduced by the N group multiplication operations on $\mathbb{G}_1$.

New File Creation The main computation overhead of this operation is the encryption of the data file using the symmetric DEK as well as the encryption of the DEK using KP-ABE. The complexity of the former depends on the size of the underlying data file and inevitable for any cryptographic method. The computation overhead for the latter consists of $|I|$ multiplication operations on $\mathbb{G}_1$ and 1 multiplication operation on $\mathbb{G}_2$, where I denotes the attribute set I of the data file. All these operations are for the data owner.

File Deletion This operation just involves the data owner and Cloud Servers. The former needs to compute one signature and the latter verifies this signature.

New User Grant This operation is executed interactively by the data owner, Cloud Servers, and the user. The computation overhead for the data owner is mainly composed of the generation of the user secret key and encryption of the user secret key using the user's public key. The former accounts for $|L|$ multiplication operations on $\mathbb{G}_1$, where L denotes the set of leaf nodes of the access tree. The latter accounts for one PKC operation, e.g., RSA encryption. The main overhead for Cloud Servers is one signature verification. The user needs to do two PKC operations, one for data decryption and the other for signature verification.

User Revocation This operation is composed of two stages. The second stage can actually be amortized as the file access operation. Here we just counts the operation overhead for the first stage. That for the second stage will be included in the file access operation. The first stage occurs between the data owner and Cloud Servers. The computation overhead for the data owner is caused by the execution of $AMinimalSet$ and $AUpdateAtt$ as well as the generation of his signatures for the public key components. The complexity of algorithm $AMinimalSet$ is actually mainly contributed by the CNF conversion operation which can be efficiently realized by existing algorithms such as [33] (with the complexity linear to the size of the access structure). Assuming the size of the minimal set returned by $AMinimalSet$ is D, $D \leq N$, the computation overhead for $AUpdateAtt$ is mainly contributed by D multiplication operations on $\mathbb{G}_1$. In addition, the data owner also needs to compute D signatures on public key components. The computation overhead on Cloud Servers in this stage is negligible. When counting the complexity of user revocation, we use N instead of the size of the access structure since in practical scenarios $AMinimalSet$ is very efficient if we limit the size of access structure (without af-

fecting system scalability), but each signature or multiplication operation on $\mathbb{G}_1$ is expensive.

File Access This operation occurs between Cloud Servers and the user. For Cloud Servers, the main computation overhead is caused by the execution of algorithm $AUpdateSK$ and algorithm $AUpdateAtt4File$. In the worst case, the algorithm $AUpdateSK$ would be called $|L| - 1$ times, which represents $|L| - 1$ multiplication operations on $\mathbb{G}_1$. Each execution of the algorithm $AUpdateAtt4File$ accounts for one multiplication operation on $\mathbb{G}_1$. In the worst case, Cloud Servers need to call $AUpdateAtt4File$ N times per file access. Our lazy re-encryption solution will greatly reduce the average system-wide call times of these two algorithms from statistical point of view. File decryption needs $|L|$ bilinear pairing in the worst case. Fig.6.8 summarizes the computation complexity of our proposed scheme.

Operation	Complexity
File Creation	$\mathcal{O}(\|I\|)$
File Deletion	$\mathcal{O}(1)$
User Grant	$\mathcal{O}(\|L\|)$
User Revocation	$\mathcal{O}(N)$
File Access	$\mathcal{O}(max(\|L\|, N))$

Figure 6.8: Complexity of our proposed scheme

Ciphertext Size As depicted in Section 6.2.3, the ciphertext is composed of an ID, a header, and a body. The body is just the data block. The header for each data file is composed of an attribute set I, one group element on $\mathbb{G}_2$, and $|I|$ group elements on $\mathbb{G}_1$.

6.3.3 Related Work

Existing work close to ours can be found in the areas of "shared cryptographic file systems" and "access control of outsourced data".

In [14], Kallahalla et al proposed Plutus as a cryptographic file system to secure file storage on untrusted servers. Plutus groups a set of files with similar sharing attributes as a file-group and associates each file-group with a symmetric lockbox-key. Each file is encrypted using a unique file-blcok key which is further encrypted with the lockbox-key of the file-group to which the file belongs. If the owner wants to share a file-group, he just delivers the corresponding lockbox-key to users. As the complexity of key management is proportional to the total number of file-groups, Plutus is not suitable for the case of fine-grained access control in which the number of possible "file-groups" could be huge.

In [29], Goh et al proposed SiRiUS which is layered over existing file systems such as NFS but provides end-to-end security. For the purpose of access control, SiRiUS attaches each file with a meta data file that contains the file's access control list (ACL), each entry of which is the encryption of the file's file encryption key (FEK) using the public key of an authorized user. The extension version of SiRiUS uses NNL broadcast encryption algorithm [34] to encrypt the FEK of each file instead of encrypting it with each individual user's public key. As the complexity of the user revocation solution in NNL is proportional to the number of revoked users, SiRiUS has the same complexity in terms of each meta data file's size and the encryption overhead, and thus is less scalable.

Ateniese et al [30] proposed a secure distributed storage scheme based on proxy re-encryption. Specifically, the data owner encrypts blocks of content with symmetric content keys. The content keys are all encrypted with a master public key, which can only be decrypted by the master private key kept by the data owner. The

data owner uses his master private key and user's public key to generate proxy re-encryption keys, with which the semi-trusted server can then convert the ciphertext into that for a specific granted user and fulfill the task of access control enforcement. The main issue with this scheme is that collusion between a malicious server and any single malicious user would expose decryption keys of all the encrypted data and compromise data security of the system completely. User secret key accountability is neither supported.

In [11], Vimercati et al proposed a solution for securing data storage on untrusted servers based on key derivation methods [35]. In this proposed scheme, each file is encrypted with a symmetric key and each user is assigned a secret key. To grant the access privilege for a user, the owner creates corresponding public tokens from which, together with his secret key, the user is able to derive decryption keys of desired files. The owner then transmits these public tokens to the semi-trusted server and delegates the task of token distribution to it. Just given these public tokens, the server is not able to derive the decryption key of any file. This solution introduces a minimal number of secret key per user and a minimal number of encryption key for each file. However, the complexity of operations of file creation and user grant/revocation is linear to the number of users, which makes the scheme less scalable. User access privilege accountability is also not supported.

6.3.4 Discussion

According to the above analysis, we can see that our proposed scheme is able to realize the desired security goals, i.e., fine-grained access control, data confidentiality, and user secret key accountability. The goal of scalability is also achieved since the complexity for each operation of our proposed scheme, as is shown in Fig. 6.8, is independent to the number of users and that of data files in the system. Therefore,

our proposed scheme can serve as a promising candidate for data access control in the emerging cloud computing environment. On the contrary, existing access control schemes in related areas, when applied to the cloud computing environment, have various limitations in terms of system scalability, fine-grainedness of access policies, and adequate protection over data confidentiality.

6.4 Summary

This chapter aims at fine-grained data access control in cloud computing. One challenge in this context is to achieve fine-grainedness, data confidentiality, and scalability simultaneously, which is not well supported by current work. In this work we propose a scheme to achieve this goal by exploiting KP-ABE and uniquely combining it with techniques of proxy re-encryption and lazy re-encryption. Our proposed scheme can enable both the data owner and users to delegate most of computation overhead to powerful cloud servers. User secret key accountability can also be achieved. Formal security proofs show that our proposed scheme is secure under standard cryptographic models.

Chapter 7

Secure Data Sharing with ABE in Wireless Sensor Networks

This chapter addresses the issue of secure data sharing for distributed data storage in Wireless Sensor Networks (WSNs) [59–61]. In WSNs, storing data at local sensor nodes or at designated in-network nodes greatly saves the network-wide communication load and has a lot of benefits such as energy-efficiency [65–70, 85]. However, unattended wireless sensor nodes are very likely subject to strong attacks such as physical compromise. In this sense a storage node in WSNs can be viewed as an untrusted storage since the owner of the WSN may have concerns on data security in mission-critical applications if data are stored without proper protection. A secure data storage and retrieval scheme is required for distributed data storage in WSNs. When previous works [71–82] focus on data confidentiality and integrity protection or communication security, the issue of fine-grained data access control in WSNs is seldom addressed. In this chapter we address this issue and provides a cryptographic-based access control mechanism with ABE. The main challenge in this work is to make the expensive ABE operations affordable to resource-constrained

sensor nodes. We resolve this issue by dividing the lifetime of sensor nodes into phases and then distribute the underlying mathematical operations in ABE over these phases. To minimize the communication and computation load on sensor nodes in case of user revocation, we revise an existing ABE scheme and makes the user revocation complexity on sensor nodes constant. Formal security proof and experimental results shows that our proposed solution is provably secure and affordable to real sensor nodes. To the best of our knowledge, our work is de facto the first that provides a secure mechanism for distributed fine-grained data access control in WSNs.

7.1 Models and Assumptions

7.1.1 Network Model

In this work, we consider a wireless sensor network composed of a network controller which is a trusted party, a large number of sensor nodes, and many users. Throughout this chapter, we will denote the network controller with the symbol $\mathcal{T}$. Symbol $\mathcal{U}$ and $\mathcal{N}$ are used to represent the universe of the users and the sensor nodes respectively. Both users and sensor nodes have their unique IDs. Symbol $\mathcal{U}_i$ will be used to denote user i, and $\mathcal{N}_j$ to represent sensor node j. The trusted party $\mathcal{T}$ can be online or off-line. It comes online merely on necessity basis, e.g., in the case of intruders detected. Each sensor could be a high-end sensor node such as iMote2 which has greater processing capability and a larger memory than conventional sensor nodes. Sensor data could be stored locally or at some designated in-network locations [68–70,85]. As is conventionally assumed, we consider each user $\mathcal{U}_i$ to have sufficient computational resources to efficiently support expensive cryptographic operations such as bilinear map. In addition, we assume there is a loose

time synchronization among the sensor nodes, and the lifetime of the network is further divided into phases based on the time synchronization. Such an assumption can also be found in previous works such as [83].

7.1.2 Adversary Model

This work considers attackers whose main goal is to learn about the contents of sensor data that they are not authorized to. The adversaries could be either external intruders or unauthorized network users. Due to lack of physical protection, sensor nodes are usually vulnerable to strong attacks. In particular, we consider the adversary with both passive and active capabilities, which can (1)eavesdrop all the communication traffics in the WSN, and (2) compromise and control a small number of sensor nodes. In addition, (3)unauthorized users may collude to compromise the encrypted data.

7.1.3 Security Requirements

For the purpose of securing distributed data storage in WSNs, the main goal of this work is to protect contents of sensor data from being learned by attackers, including external intruders and unauthorized network users. With respect to data access control in WSNs, we recognize the following unique but not necessarily complete security requirements.

Fine-grained Data Access Control: In many application scenarios, especially mission-critical cases, disclosure of sensitive data should be well controlled such that different users many have access privileges over different types of data. For this purpose, we need to define and enforce a flexible access policy for each individual user based on the user's role in the system. In particular, the access policy should be able to define a unique set of data that the user is authorized to access, and must

be enforced via a cryptographic method since sensor nodes are vulnerable to strong attacks like physical compromise.

Collusion Resilience: As described by our adversary model, unauthorized users may cooperate for the purpose of learning about the contents of sensitive data. This requires our data access control scheme to be resilient to collusion attacks in the sense that the collaboration of unauthorized users will not give them additional advantages over what they can directly obtain from executing attacks individually.

Sensor Compromise Resistance: Due to lack of compromise-resistant hardware, a small number of sensor nodes could be physically compromised by the adversary in hostile environments. Now that the adversary can always obtain the sensor data generated by a sensor node after it is compromised, we should at least secure sensor data such that, (1) compromising the sensor node does not disclose the sensor data generated before the sensor is compromised, and (2) compromising one sensor node does not give the adversary any assistance to obtain sensor data generated by other sensor nodes.

Backward Secrecy: User management is an important functionality required by most application scenarios. In particular, the system should be able to handle user revocation in the case of user leaving request or malicious behavior detected. To support such a functionality, the data access control mechanism should guarantee that the revoked users are not able to access the sensor data generated after they are revoked.

7.2 Our Proposed Scheme

This section presents our data access control scheme for distributed data storage in WSNs. We first introduce our access control strategy. Next, we give an overview of

our proposed scheme. Then, we present the detailed description of our basic scheme, which is followed by an advanced design.

7.2.1 Access Control Strategy

For the purpose of achieving fine-grained data access control in WSNs, we first explore some inherent natures of WSNs. In general, the deployments of most WSNs are aimed at collecting certain types of data for specific application(s). Therefore, we are able to specify individual sensors (and hence the data collected by them) through a set of predefined attributes. For example, in the battlefield, sensor nodes are usually deployed to collect military information in certain geographic location. Each sensor node may be responsible for collecting specific types of data such as vibration, smoke, so on and so forth. Sensor nodes may also have their owners, i.e., persons or units who are in charge of them. In particular, some nodes may be jointly owned by different units. Hence, we may specify sensor nodes using these attributes such as {location = *village*, data type = (*vibration*, *smoke*), owner = (*explosion experts*, *officers*, *scouts*)}. This further enables us to specify data access privileges of users based on these attributes. In the above example, we may designate the access structure of a user as "(location is village) AND (type is vibration)", which allows the users to obtain vibration data within the village area. We may also define more sophisticated access structures such as "(location is village) AND (type is vibration OR smoke) AND (at least owned by 2 of the following: explosion experts, officers, scouts)". In this case, the user can only access vibration and smoke data collected within the village area. In addition, the last condition implicitly requires the user to belong to at least two of the three designated groups. To enforce these access structures, we predefine a public key component for each of the attributes, and encrypt sensor data with public key components of the corresponding attributes

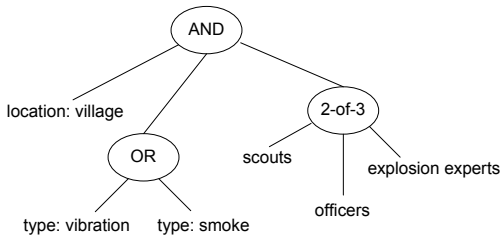


Figure 7.1: An example access structure in the battlefield scenario

such that only the users with "satisfiable" access structures[14] are able to decrypt.

Having discussed the intuitive idea of our data access control strategy, we further present it more formally as follows. In our proposed scheme, we associate each sensor node (and hence its collected data) with a set of attributes, for each of which we define a public key component. Each user is assigned an access structure, which is implemented via an access tree and embedded in the user's secret key. Every leaf node of the access tree is labelled with an attribute and the interior nodes are defined as threshold gates[15]. This kind of definition of user access structure is able to represent very expressive logic expressions over attributes, and thus specify data access privileges of users in the fine-grained manner. Actually, we are able to represent any general (monotone or non-monotone) access structures if we define the NOT of each attribute as a separate attribute, which in turn will double the number of attributes in our system. Fig 7.1 illustrates the aforementioned access structure in the battlefield scenario.

Formally, in this work we will denote the universe of all the sensor attributes in a WSN by a symbol $\mathcal{I}$. The set of attributes owned by each single sensor node is denoted by a symbol $\mathcal{I}_i$, where i is the sensor node ID. We have $\mathcal{I} = \bigcup_{\forall i \in \mathcal{N}} \mathcal{I}_i$. Let $k = \max_{\forall i \in \mathcal{N}} |\mathcal{I}_i|$. k will be a system parameter used by our scheme. The access

[14]That is to say, the logic expression represented by the access structure returns TRUE over the data attributes.

[15]A t-of-n threshold gate outputs TRUE if and only if at least t out of the n inputs are TRUE. Two extreme examples are AND gates (n-of-n) and OR gates (1-of-n).

structure is generally denoted by $\mathcal{P}$.

7.2.2 Scheme Overview

In our basic scheme, each sensor node is preloaded with a set of attributes as well as the public key PK. Each user is assigned an access structure and the corresponding secret key SK. As mentioned in section 7.1.1, the lifetime of the sensor network is divided into *phases*, each of which has the same time duration. Based on this, we further define each n consecutive phases as a *stage*, where $n < k$ and $k = \max_{\forall i \in \mathcal{N}} |\mathcal{I}_i|$ is a system parameter. Therefore, the lifetime of the sensor network can also be represented by a series of consecutive *stages*, numbering as $1, 2, \cdots, m$, where m is a system parameter. Sensor nodes encrypt sensed data using a symmetric-key algorithm, e.g., AES. Over each phase every sensor node updates its data encryption key once in the way that the data encryption keys during one stage form a one-way key chain. The key update algorithm could be any standard one-way hash function such as SHA-1. We call the first key on this key chain by the *master key*, denoted by K. The master key of each stage is always generated during its preceding stage, and encrypted under the preloaded attributes. Upon request for sensor data, the sensor node responds with the encrypted master key as well as the ciphertext of the sensor data. If the user is an intended receiver, he is able to decrypt the master key and derive the data encryption keys for phases of his interest, and thus decrypt the sensor data. Based on the basic scheme, our advanced scheme goes one step further by providing the functionality of user revocation, which is demanded by most WSN application scenarios. In the advanced scheme, $\mathcal{T}$ is able to revoke any user via broadcasting a user revocation message to all the users and all the sensor nodes respectively. In particular, the user revocation message for the sensor nodes contains merely one group element of $\mathbb{G}_T$.

7.2.3 The Basic Scheme

The construction of our basic scheme based on KP-ABE [12] and the one-way key chain is as follows.

1. System Initialization

On initialization, $\mathcal{T}$ executes the following steps:

a) Select two multiplicative cyclic groups $\mathbb{G}_1$ and $\mathbb{G}_T$ of prime order p as well as a bilinear map $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Let g be the generator of $\mathbb{G}_1$.

b) Choose a number t_i uniformly at random from $\mathbb{Z}_p$ for each attribute $i \in \mathcal{I}$, and y randomly from $\mathbb{Z}_p$. Output the public key as follows:

$$PK = \langle G_1, g, Y = e(g,g)^y, T_1 = g^{t_1}, \cdots, T_{|\mathcal{I}|} = g^{t_{|\mathcal{I}|}} \rangle$$

The master secret key is $MK = (y, t_1, \cdots, t_{|\mathcal{I}|})$.

c) Choose a secure one-way hash function, denoted as $h(\cdot)$. Pre-load the following information to each sensor node $\mathcal{N}_i$:

$$\mathcal{T} \rightarrow \mathcal{N}_i: \ \mathcal{I}_i, \ h(\cdot), \ PK$$

d) For each user $\mathcal{U}_j$, $\mathcal{T}$ generates an access structure $\mathcal{P}$ and computes his secret key SK as follows. Starting from the root node r of $\mathcal{P}$ and in the top-down manner, construct a random polynomial q_x of degree $d_x + 1$ using Lagrange interpolation for each node x in $\mathcal{P}$, where d_x is the threshold value of node x. For each non-root node x in $\mathcal{P}$, set $q_x(0) = q_{parent(x)}(index(x))$, where $parent(x)$ is the parent of x and $index(x)$ is the unique index number of x

given by its parent. In particular, set $q_r(0) = y$. SK is output as follows

$$SK = \langle \{D_i = g^{\frac{q_i(0)}{t_i}}\}_{i \in \mathcal{L}} \rangle$$

where $\mathcal{L}$ denotes the set of leaf nodes in $\mathcal{P}$. Then, $\mathcal{U}_j$ is pre-loaded with the following information

$$\mathcal{T} \rightarrow \mathcal{U}_j : \mathcal{P},\ SK,\ h(\cdot),\ PK$$

2. Master Key Encryption

During each stage $v \in [1, m]$, $\mathcal{N}_i$ generates a new master key for stage $v + 1$ and encrypts it as follows:

a) Select a number s uniquely at random from $\mathbb{Z}_p$.

b) On each phase of stage v, calculate one item $E_i = T_i^s$ for attribute $i \in \mathcal{I}_i$. After $|\mathcal{I}_i|$ phases, $|\mathcal{I}_i| \leq k$, $\mathcal{N}_i$ has the complete set $\{E_i = T_i^s\}_{i \in \mathcal{I}_i}$.

c) Randomly select a number $K \in \mathcal{K}$ as the master key of the key chain, where $\mathcal{K}$ denotes the key space. Then, compute $E' = KY^s$. Finally, store the ciphertext as follows:

$$E^{v+1} = \langle v + 1,\ \mathcal{I}_i,\ KY^s,\ \{E_i = T_i^s\}_{i \in \mathcal{I}_i} \rangle$$

where E^{v+1} represents the encrypted master key for the $(v + 1)^{th}$ stage.

3. Data Storage

$\mathcal{N}_i$ encrypts and stores the sensor data generated in the current phase, say phase $t \in [1, n]$ of stage $v \in [1, m]$, as follows:

a) Calculate the data encryption key $K_t = h(K_{t-1})$. In particular, we set $K_0 = K$.

b) Encrypt the sensor data, denoted by D, with current data encryption key K_t. Then, store the item $\langle v, t, \{D\}_{K_t} \rangle$, where $\{D\}_{K_t}$ represents the encrypted sensor data.

c) Erase K_{t-1} from the memory.

For each sensor node, all the data encryption keys used during one stage form a one-way key chain. The sensor node just keeps the latest data encryption key in its memory, while erasing all the previous ones.

4. Data Access

Assume user $\mathcal{U}_j$ is requesting for sensor data generated by sensor node $\mathcal{N}_i$ during phase t of stage v. $\mathcal{N}_i$ responds the data query request with the following message:

$$\mathcal{N}_i \rightarrow \mathcal{U}_j : \langle E^v,\ \{D\}_{K_t} \rangle$$

On receiving the response from $\mathcal{N}_i$, $\mathcal{U}_j$ executes the following steps to obtain the sensor data:

a) Decrypt the master key K of stage v from E^v. The decryption process starts from the leaf nodes and in the bottom-up manner. First, $\mathcal{U}_j$ computes the value F_i for each leaf node i in $\mathcal{P}$ as follows.

$$F_i = \begin{cases} e(D_i, E_i) = e(g,g)^{sq_i(0)}, & \text{if } i \in \mathcal{I}_i \ ; \\ \perp, & \text{otherwise.} \end{cases}$$

Then, it proceeds in the recursive way from the second last layer as follows: for node x which is a d_x-of-n gate, if more than $n - d_x$ children returns $\perp$,

$F_x = \perp$. Otherwise,

$$F_x = \prod_{i \in S_x} F_i^{\delta_i(0)} = \prod_{i \in S_x} e(g,g)^{sq_i(0)\delta_i(0)} = e(g,g)^{sq_x(0)}$$

where S_x denotes the set of x's children and $\delta_i(0)$ is the Lagrange coefficient which can be calculated by the user himself. If $\mathcal{P}$ "accepts" $\mathcal{I}_i, \mathcal{U}_j$ will finally obtain $e(g,g)^{sq_r(0)} = e(g,g)^{sy}$ and thus decrypt the master key K. Otherwise, the decryption algorithm returns $\perp$.

b) If the decryption algorithm returns $\perp$, terminate. Otherwise, $\mathcal{U}_j$ calculates the data encryption K_t from K by $K_t = h^t(K)$, and finally decrypts the sensor data with K_t.

In this basic scheme, we assign each sensor node a set of attributes and each user an access structure. Sensor data are encrypted under the attributes such that only the users with "satisfiable" access structures are able to decrypt. As the access structure is very expressive, we are able to precisely control the access privilege of each user, and thus enjoy fine-grained data access control. The access policies in the basic scheme are actually enforced by using KP-ABE [12]. To alleviate the computation overload, we divide the lifetime of sensor nodes into stages and phases. On each stage a master key is generated to serve as the "seed" for the data encryption keys of the underlying phases. For the purpose of access control, we just need to encrypt the master key of each stage under the attribute-based encryption algorithm. Sensor data are encrypted using symmetric-key encryption such as AES which is very efficient. As master keys are generated at a relative low frequency, we are able to distribute the computation overload of attribute-based encryption into each phase and thus make the expensive operations affordable to the sensor nodes.

5. User Revocation

Another fundamental functionality of WSNs is user management. In particular, we stress that the network operator should be able to revoke the user's access privilege when necessary. In our basic scheme, we can use the following approaches to revoke users from the system: one approach is to define some time attributes [2,18], and embed an expiration date to each user's access structure based on the time attributes. Sensor nodes can then associate a time stamp to each ciphertext using the time attributes. If sensor nodes always associate the current time stamp to ciphertexts, users will be automatically revoked after their designated expiration dates. Another approach for user revocation is to define some "identity attributes" [13], e.g., defining a binary attribute for each bit of user identity, and associate the corresponding identity attributes to each user's access structure. Sensor nodes can then associate any intended user list with each ciphertext using the "identity attributes". To revoke a user, sensor nodes can encrypt data using a selected set of "identity attributes" which exclude the revoked user's identity. The advantage of the two approaches is that they do not involve extra communication with users. However, the limitation of them is also obvious. For the first approach, users can only be revoked at a pre-defined time. It does not support user revocation on the fly. The second approach is "stateful", i.e., every ciphertext (and hence the sensor nodes) needs to remember all the revoked users in the history. The ciphertext size would keep increasing as more and more random users are revoked, which ends up with a heavy computation and communication overhead on each sensor node after several rounds of user revocation. To resolve this issue, in this work we propose to update secret keys of all the users but the one(s) to be revoked. More specifically, we will update a common master key component which is embedded into every user's secret key as we will discuss in detail. The benefits of this key update method can be

summarized as follows. First, this approach is "stateless" and sensor nodes do not need to "remember" any revoked user in the history. Second, the user revocation process does not introduce too much communication or computation overhead on each sensor node. Actually, each sensor node just needs to update one of its public key components which can be efficiently achieved by broadcasting the common public key component to all the sensor nodes. Consequently, the affect of user revocation on each sensor node is minimal. The main issue with the proposed solution, however, is that it needs every user to communicate with the authority via unicast to update his secret key. To resolve this issue, we revise the original KP-ABE construction so that we can update secret keys for all non-revoked users by broadcasting a common element to them, which can be efficiently realized by existing broadcast encryption techniques. We also prove that our revision to KP-ABE has the same security strength as the original construction in terms of semantic security of data.

7.2.4 The Advanced Scheme

The basic idea of our advanced user revocation solution is to separate the master secret key y from the user access structure in the user secret key SK. Update of user secret keys can thus be realized by updating the embedded secret y which is common to every user's secret key. As a result, we can update user secret keys via broadcasting the incremental of y while excluding the leaving user from the recipient list. Based on this general idea, we present our advanced scheme as follows. For brevity, we just present the parts that need to be changed as compared to our basic scheme.

1. System Initialization

$\mathcal{T}$ executes the following steps.

a) The same as step a) of 1) in the basic scheme.

b) In addition to the elements generated by step b) of 1) in the basic scheme, $\mathcal{T}$ selects a number β uniquely at random from $\mathbb{Z}_p$. The public key PK and the master secret key MK are then output as follows.

$$PK = \langle G_1,\ g,\ Y,\ \{T_i = g^{t_i}\}_{i\in\mathcal{I}},\ g^{\beta}\rangle$$
$$MK = \langle y, t_1, \cdots, t_{|\mathcal{I}|}, \beta\rangle$$

c) The same as step c) of 1) in the basic scheme.

d) Sensor node $\mathcal{N}_i$ is pre-loaded with the following

$$\mathcal{T} \rightarrow \mathcal{N}_i :\ \mathcal{I}_i,\ h(\cdot),\ PK$$

e) The process of key generation is similar to step d) of 1) in the basic scheme. $\mathcal{T}$ outputs the user secret key SK as follows.

$$SK = \langle g^{\frac{y-\theta}{\beta}}, \{D_i = g^{\frac{q_i(0)}{t_i}}\}_{i\in\mathcal{L}}\rangle$$

Compared to the basic scheme, this algorithm introduces a new element $g^{\frac{y-\theta}{\beta}}$ into SK, where $\theta = q_r(0)$ is randomly selected from $\mathbb{Z}_p$, and q_r denotes the polynomial for the root node r in $\mathcal{P}$. $\mathcal{U}_j$ is then preloaded with $\langle \mathcal{P},\ SK,\ h(\cdot),\ PK\rangle$.

2. Master Key Encryption

Similar to 2) in the basic scheme. The advanced scheme introduces a new element $g^{\beta s}$ into the ciphertext as follows:

$$E^{v+1} = \langle v+1, \mathcal{I}_i, KY^s, \{E_i = T_i^s\}_{i\in\mathcal{I}_i},\ g^{\beta s}\rangle$$

3. Data Storage The same as 3) in the basic scheme.

4. Data Access

This part is the same as 4) in the basic scheme except for step a).

a) The decryption process is similar to that in the basic scheme. When the data attributes satisfy the user's access structure $\mathcal{P}$, the user obtains $e(g,g)^{\theta s}$. Then, he decrypts the message as follows.

$$M = \frac{Me(g,g)^{ys}}{e(g^{\frac{y-\theta}{\beta}}, g^{\beta s})e(g,g)^{\theta s}}$$

In this advanced scheme, $\mathcal{T}$ is able to update the master secret key y embedded in the user secret key SK by broadcasting $g^{\frac{\Delta y}{\beta}}$ to the users, where Δy is the incremental of y. With the above enhancement, we can present our user revocation scheme as follows.

5. User Revocation

To revoke a user $\mathcal{U}_j$, $\mathcal{T}$ needs to update the master secret key y for the sensor nodes as well as the remaining users. The process can be illustrated as follows.

$$\mathcal{T}: \ y' \leftarrow \mathbb{Z}_p, \ \Delta y \leftarrow y' - y, \ Y' \leftarrow e(g,g)^{y'}, \ \ g^{\frac{\Delta y}{\beta}}$$
$$\mathcal{T} \rightarrow \mathcal{N}: \ Y'$$
$$\mathcal{T} \rightarrow \mathcal{U} \backslash \mathcal{U}_j: \ g^{\frac{\Delta y}{\beta}}$$

First, $\mathcal{T}$ chooses a random number $y' \in \mathbb{Z}_p$ as the new value of the master secret key y. The incremental is set as $\Delta y = y' - y$. Then, it calculates the new public key $Y' = e(g,g)^{y'}$ and the group element $g^{\frac{\Delta y}{\beta}}$. Finally, $\mathcal{T}$ broadcasts Y' to all the sensor nodes and $g^{\frac{\Delta y}{\beta}}$ to all the users excluding the one to be revoked. Upon receiving the master secret key update message, each sensor node simply replaces the public key

Y with Y'[16]. The master key for the next stage will be encrypted under the new public key. Each user updates his secret key as follows: $g^{\frac{y-\theta}{\beta}} g^{\frac{\Delta}{\beta}} = g^{\frac{y'-\theta}{\beta}}$. The master secret key y is thus updated as y'. In this user revocation scheme, one challenging issue is to selectively broadcast $g^{\frac{\Delta y}{\beta}}$ such that all but the leaving users are able to receive it. Fortunately, there are plenty of off-the-shelf selectively broadcast schemes available for different application scenarios. [34] is able to broadcast any $n-r$ out of n users with ciphertext size of $O(r)$ and private key size of $O(log^2 n)$, which is further reduced to $O(logn)$ by [86]. This scheme is suitable for application scenarios where the number of revoked users each time is small. In [46], Boneh et al. proposed a scheme which is able to broadcast to arbitrary subset of users with constant ciphertext size (only two group elements). This scheme is extremely suitable for bandwidth-critical applications. One drawback of this basic scheme of [46] is that the public key size is of $O(n)$. To balance the size between the public key and the ciphertext, a revised scheme is presented in which both the ciphertext and the public key are of size $O(\sqrt{n})$. In [48], Cheung et al. proposed a collusion-resistant broadcast encryption scheme based on flat table scheme [56] and attribute-based encrytpion [2]. Both the ciphertext and the user secret key are of size $O(logn)$. In [13], Yu et al. further improved [48] by supporting receiver anonymity. [48] and [13] are suitable for scenarios in which the system wants to revoke users of some common attributes, or the number of revoked users each time is small. In our proposed scheme, we do not designate any particular selective broadcast scheme for user secret key update. The system designer can pick an appropriate broadcast scheme from the above candidates according to the requirement of the actual system.

6. Further Enhancement

[16]Note that, the new public key Y' should be signed by the authority so that the sensor nodes can verify its authenticity. The signature scheme can be off-the-shelf algorithms such as ECDSA. For brevity, we do not explicitly include the signature in our proposed scheme.

In the above user revocation scheme, $g^{\frac{\Delta y}{\beta}}$ is an update message common to all non-revoked users, which opens the door for a non-revoked user to collude with revoked users and help them decrypt the data. In [4], Boldyreva et al. proposed a user revocation scheme for IBE and KP-ABE in which user collusion attacks are well addressed. The proposed scheme is built on top of the construction of Fuzzy IBE [19] and the binary tree data structure. More specifically, it introduces a time attribute and use it in the encryption of each message. The root node of each user's access tree is an AND gate with one child being the time attribute and the other being the root node of ordinary access structure. When a user is to be revoked, the system administrator generates key updates on the time attribute using the binary tree, each leaf node of which is associated to one user. Since new messages will be encrypted with the updated time attribute, users didn't receive the key updates will not be able to decrypt. In this scheme, the complexity of encryption and decryption is comparable to that of current KP-ABE [12]. The complexity of user revocation in terms of message size and computation overhead is $O(rlog(\frac{n}{r}))$ when $1 < r \leq n/2$, or $O(n-r)$ when $n/2 < r < n$, where r is the total number of revoked users and n is the total number of users. It should be noted that, we can also use this revocable KP-ABE in our scheme for achieving fine-grained data access control. One significant advantage of using the revocable KP-ABE is its enhanced security against user collusion. However, the complexity of user revocation is linear to the number of revoked users which could raise concerns in large scale systems when that number is approaching $n/2$. In our proposed user revocation solution, the system designer is free to choose a broadcast scheme. For example, he/she can use [46] which has the constant ciphertext size if communication overhead is of the most importance. However, security level is reduced in this solution. We treat the above issue as a trade-off between efficiency and security, and leave the choice to

the system deployer.

7.2.5 Discussions

1. Change of Sensor Attributes

Conventionally the set of attributes of each sensor node does not change throughout the node's lifetime, or we can make this assumption as it is enough for many application scenarios. Nevertheless, there are still some cases in which the attributes of sensor nodes would change. For example, in some dynamic environments such as battlefields, the location of a portion of sensor nodes might be adjusted frequently. In this case, it is desirable to change the location attributes for the involved sensor nodes while not affecting the others. To achieve this goal, we just need to load the involved sensor nodes with the new attribute lists as well as the corresponding public key components. This can be easily realized in our proposed scheme as long as the involved sensor nodes are convinced of the authenticity of the update, which can be realized without any difficulty by attaching the network controller's signature to the update.

2. Support for Concealed Data Aggregation

In-network aggregation of data has been put forward as an important paradigm for wireless sensor networks which enables in-network consolidation of redundant data and thus saves energy [87]. In practical settings, it is often desired to provide in-network data aggregation while guaranteeing data confidentiality for privacy concerns. The concept of Concealed Data Aggregation (CDA) [88] was proposed to address this issue. With CDA, the intermediate nodes are able to aggregate data by performing the aggregation operations on incoming ciphertexts without knowing the data encryption keys nor the plaintext. To realize CDA, sensing nodes should encrypt data using certain encryption transformation, a.k.a. *privacy homomorphism*

(PH). A survey on existing PH schemes, including symmetric and asymmetric ones, can be found in [89]. We stress that our proposed scheme can seamlessly integrate existing symmetric PH schemes and thus realize CDA. To justify this, we take [90] as an example and show how that PH scheme is integrated with our proposed scheme. At a high level the PH scheme in [90] has the property as follows. Given two messages m_1 and m_2, and their respective encryption keys k_1 and k_2, if $c_1 = Enc(m_1, k_1)$ and $c_2 = Enc(m_2, k_2)$, then $c_1 + c_2 = Enc(m_1 + m_2, k_1 + k_2)$ and $Dec(c_1 + c_2, k_1 + k_2) = m_1 + m_2$. This property still holds for the case of more than two messages and can be used to compute statistical values, e.g., mean, variance and standard deviation, of sensed data. Intuitively, we can integrate this PH scheme into our proposed scheme in the following way: First, let each sensing node encrypt the sensed data with its data encryption key k_i and encrypt k_i (actually its seed) under its attributes. This process is basically the same as that of our proposed scheme. Then, upon data query every sensing node sends both the ciphertext of the sensed data and that of k_i to its upstream aggregating node. The intermediate aggregating nodes, after having collected all the downstream data, do the aggregation operations on the ciphertexts of data while keeping ciphertexts of the data encryption keys intact. Subsequently, they transmit the aggregated ciphertext of data along with the ciphertexts of data encryption keys to their respective upstream aggregating nodes. The above process is recursively executed until it reaches the user. The user, on receiving the ciphertexts, first recovers the data encryption keys if his access structure satisfies with the sets of attributes of all the sensing nodes. Then he does the same aggregation operations over the recovered data encryption keys as all the aggregating nodes did to compose a "aggregated" data encryption key of the final data ciphertext and decrypt the aggregated value of data. The drawback of this intuitive solution is that the ciphertexts of the data encryption keys could

be a heavy communication overhead. This is because the size of such a ciphertext grows linear with the number of attributes of the sensing node and each intermediate aggregating node should forward these ciphertexts of all its downstream sensing nodes. To alleviate this overhead, we can enforce our access control strategy only on few designated upstream aggregating nodes. In this way the downstream sensing nodes just need to encrypt sensed data with their data encryption keys. These designated upstream aggregating nodes fulfill our access control strategy by encrypting the "aggregated" data encryption keys under certain set of attributes. One issue underlying this method is that the aggregating nodes should distribute data encryption keys to all its downstream sensing nodes, which can be resolved using existing key distribution methods [91].

7.3 Scheme Evaluation

This section evaluates our proposed scheme in terms of security and performance aspects.

7.3.1 Security Analysis

We evaluate the security of our work by analyzing the its fulfillment of the security requirements described in section 7.1.

Fine-grained Data Access Control: To provide fine-grained data access control, the proposed scheme should provide a strategy that is able to define and enforce complex access policies for sensor data of various types or security levels. In FDAC, the access structure embedded in each user's secret key is able to represent complicated predicates such as disjunctive normal form (DNF), conjunctive normal form (CNF),

and threshold gates. The combination of these predicates are able to represent sophisticated access structures. In fact, our scheme is able to support non-monotonic (general) access structures if we define the NOT of each attribute as a separate attribute, which in turn will double the number of attributes in our system. To enforce our access control strategy, we encrypt the master key of the key chain in each stage under a set of attributes. Without the master key, the adversary is not able to derive the data encryption keys due to the one-wayness of the key chain, which can be guaranteed by choosing a secure one-way hash function such as SHA-1. In our basic scheme, the master key is actually encrypted under the standard key-policy attribute-based encryption (KP-ABE) scheme [12] which is provably secure. Our advanced scheme, to achieve efficient user revocation, makes some enhancement to the standard KP-ABE when encrypting the master key. The enhanced KP-ABE is provably secure under the Decisional Bilinear Diffie-Hellman (DBDH) assumption. (A formal security proof is available in our journal paper [119]). This turns out that the adversary is not able to decrypt the master key unless he owns the intended access structure. Therefore, our proposed scheme is able to control the disclosure of sensor data so that only authorized users are able to access.

Collusion Resilience: To compromise sensor data, the main task of the colluding users is to decrypt the master key of the target data if the one-wayness of the underlying one-way has function, e.g., SHA-1, is guaranteed. Since the master key is encrypted under our enhanced scheme, we have to prove that it is collusion-resistant. Intuitively, we can sketch the collusion-resistance of our enhanced scheme as follows. Recall that the master key is encrypted in the form of $Ke(g,g)^{ys}$. The user has to cancel $e(g,g)^{ys}$ to recover K. To compose $e(g,g)^{ys}$, the only way is to execute the following: $e(g^{\frac{y-r}{\beta}}, g^{\beta s}) = e(g,g)^{ys}/e(g,g)^{rs}$. To extract $e(g,g)^{ys}$, the user should

compute $e(g,g)^{rs}$. Actually, for each user, r is randomly and independently selected from $\mathbb{Z}_p$. The secret key from one unauthorized user does not give the other user any help in terms of computing $e(g,g)^{rs}$. Actually, in our security proof the security definition (cf. our journal paper []) implies collusion resistance. As our scheme is provably secure under this security definition, collusion resistance is also guaranteed.

Sensor Compromise Resistance: To meet this security requirement, we should achieve two security goals: (1) compromising the sensor node does not disclose the sensor data generated before the sensor is compromised, and (2) compromising one sensor node does not give the adversary any advantage to obtain data generated by other sensor nodes. We can show the fulfillment of our scheme with respect to these two security goals as follows: (1) In our scheme, each sensor node just keeps the current data encryption key in the memory, while erasing all the previously used keys. Because of the one-wayness of the key chain, the compromiser is not able to derive the previously keys from the current key. (2) is easy to prove since each sensor node encrypts data independently.

Backward Secrecy: As is described in the previous section, our advanced scheme is able to update the master key y for legitimate users while excluding those to be revoked. Since the new sensor data will be encrypted under the latest master key, the revoked users are not able to decrypt. One problem in our scheme is, the user revocation instruction will not take effort until a new stage starts. Such a delay occurs because it would take one stage for each sensor node to encrypt the master key under the attributes. This delay may differ for different systems. For example, if a system defines 30 phases as a stage and each phase lasts 1 second, the delay will be at the most half a minute. Generally, if a system has a stage with less phases

and each phase takes less time, e.g., each sensor node is assigned a smaller number of attributes or has a more powerful computational capability, the delay can be shorter. In practical applications, we leave this delay as a system parameter, and the system designer can adjust this parameter by changing the number of attributes assigned to each sensor node or using a different type of sensor nodes.

In addition to the security goals listed above, there are also some other security requirements such as data integrity and authenticity, which are desired by conventional WSN applications. In fact, security requirements such as message integrity can be easily supported in our scheme with minor modifications using existing off-the-shelf techniques. A challenging orthogonal issue would be data authenticity which requires sensing nodes to provide proofs of data authenticity to users. Some current work such as [92, 93] has provided salient solutions to this problem. As the main focuses of this work is fine-grained data access control, we do not explicitly address all those security problems.

7.3.2 Performance Evaluation

This section evaluates the performance of our proposed scheme in terms of computation and communication overheads. In our scheme, sensor data are generated and encrypted by sensor nodes, and retrieved and decrypted by users. As sensor nodes are usually resource constrained, they may not be able to execute expensive cryptographic primitives efficiently and thus become the bottleneck of the scheme. For this reason, our evaluation focuses on the performance of sensor nodes. In the following section, we first discuss the numeric results in terms of computation and communication overheads for sensor nodes. Then, we present our implementation results on real sensors.

1. Numeric Result

In our proposed scheme, each sensor node is responsible for the following operations in each stage: (1) generate the master key and encrypt it using our enhanced KP-ABE, (2) derive the data encryption keys based on the master key, and (3) encrypt sensor data using the data encryption keys. These operations are further distributed to each phase. Specifically, if we choose elliptic curves as the underlying bilinear group, in each phase the sensor node needs to execute at the most one scalar multiplication on elliptic curves, one one-way hash, and one symmetric key data encryption. Table 7.1 lists all these operations.

Table 7.1: Computation load on each sensor node

	Scalar Mul	Hash	Data Encryption
Each Stage	$\|\mathcal{I}_i\|$ +1	n	n
Each Phase	1 or 0	1	1

On each data retrieval request, the sensor node responds with $\langle E^v, \{D\}_{K_t} \rangle$ for sensor data of phase t in stage v, where E^v contains $|\mathcal{I}_i| + 1$ group elements on $\mathbb{G}_1$ and one on $\mathbb{G}_T$, and $\{D\}_{K_t}$ is the data payload. On user revocation, $\mathcal{T}$ only needs to broadcast one group element of $\mathbb{G}_T$ to all the sensor nodes. The communication overload for each sensor node is shown in Table 7.2.

Table 7.2: Communication load

Data Retrieval	User Revocation
($\|\mathcal{I}_i\|$ +1) $\mathbb{G}_1$ + 1$\mathbb{G}_T$ + data payload	1$\mathbb{G}_T$

2. Implementation

In our implementation, we choose Tmote Sky and iMote2 as the target platforms. We use SHA-1 as the one-way hash function and AES (supported by CC2420 Radio module of the motes) as the the data encryption algorithm. Our implementation shows that it takes about $0.06ms$ for SHA-1 to execute one hash operation and

$0.4ms$ for AES to encrypt 64 bytes data. Our implementation also shows that one scalar multiplication takes several seconds in the worst case. The scalar multiplication operation is thus the bottleneck of the sensor performance. To optimize this operation, one key issue is to find appropriate parameters for the elliptic curve.

In past years, many works have efficiently implemented Elliptic Curve Cryptography (ECC) on various sensor platforms. In these works, elliptic curves are ususally chosen according to standards such as NIST [94] and SECG [95], which enable most of the optimization methods. Although these elliptic curves serve perfectly for security schemes such as ECDH, ECDSA, et al, they are not pairing-friendly, i.e., they can not be used as bilinear groups. In FDAC, however, the elliptic curve is required to be pairing-friendly. Most pairing-friendly elliptic curves studied by current work fall into two categories, namely Supersingular (SS) curves and MNT curves. In the case of SS curves, the two elliptic groups $\mathbb{G}_1$ and $\mathbb{G}_2$ (cf. Section 2.2) could be the same. For MNT curves, $\mathbb{G}_1$ and $\mathbb{G}_2$ are different. To choose an appropriate elliptic curve, several factors should be taken into account as follows. Let l be the group size of the elliptic curve and k be its embedding degree. To achieve a comparable security strength of 1024-bit RSA, we should have lk to be larger than 1024, or at least close to 1024. Given the security level, a higher k results in a shorter group size. Therefore, choosing a high embedding degree for the elliptic curve in our scheme may result in not only a short ciphertext, but also an efficient scalar multiplications on each sensor. However, the embedding degree k of the elliptic curve can not be arbitrarily large. Choosing an appropriate embedding degree for the elliptic curve is actually another research area. According to the benchmark of Pairing-Based Crypto (PBC) library [96], elliptic curves with $l = 512$ and $k = 2$ results in the fastest bilinear pairing as compared to those with $k > 2$ for SS curves. The case is on the opposite for MNT curves. According to our testing of the PBC library on

Linux platform with an Intel Pentium D 3.0GHz CPU, SS curves with $l = 512$ and $k = 2$ (type a curves in PBC) take about $6ms$ to execute a pairing, while MNT curves with $l = 159$ and $k = 6$(type d curves in PBC) take about $14ms$ (Actually, on the user side of our solution decryption time is linear to the number of pairings). Although both results are acceptable to users, MNT curves imply a much shorter ciphertext as well as key size to sensor nodes. More importantly, scalar multiplication over 512-bit curves may not be supported by low-end sensor nodes such as Tmote Sky because it consumes too much RAM. For these reasons, we believe MNT curves with high embedding degrees are suitable for our proposed scheme.

In our implementation, the elliptic curve is a MNT curve over F_q with embedding degree of 6, where q is a 159-bit prime number. The curve has the form $y^2 = x^3 + ax + b$. Our implementation is based on the TinyECC library [97] with curve specific optimization disabled since the group size q is not a Mersenne prime. Our result shows that iMote2 consumes about $35ms$ to execute a scalar multiplication when working at 416MHz, $69ms$ at 208MHz, and $139ms$ at 104MHz. Tmote Sky consumes $4.1s$. For the 512-bit SS curve, iMote2 consumes $170ms$ at 416MHz, $341ms$ at 208MHz, and $682ms$ at 104MHz. Tmote Sky does not have enough RAM to support 512-bit SS curve. Tab.3 to Tab.5 summarize the above implementation results.

Table 7.3: One-phase computation load on iMote2 (MNT curves)

SHA-1	AES	One Scalar Multiplication		
		104MHz	208MHz	416MHz
0.06ms	0.4ms	139ms	69ms	35ms

Table 7.4: One-phase computation load on iMote2 (SS curves)

SHA-1	AES	One Scalar Multiplication		
		104MHz	208MHz	416MHz
0.06ms	0.4ms	682ms	341ms	170ms

Table 7.5: One-phase computation load on Tmote Sky

SHA-1	AES	One Scalar Multiplication	
		MNT Curves	SS Curves
0.07ms	0.4ms	4.1s	N/A

We can estimate energy consumption of one phase using the equation $W = U \times I \times t$, where U is the voltage, I is the current draw, and t is the execution time for one phase. According to the date sheet of each platform [98, 99], the current draw for TMote Sky is $1.8mA$ and the voltage can be chosen as $3v$. The iMote2 data sheet [98] just gives the current draw for running at 104MHz with radio on, which is $66mA$. To be conservative, we use this value in our computation. The voltage of iMote2 can be chosen as $0.95v$. Based on the execution time we measured, the energy consumption on the iMote2 platform (running at 104MHz) for one phase is $8.74mJ$ in case of MNT curves and $42.79mJ$ in case of SS curves. The energy consumption on the TMote Sky platform for one phase is $24.68mJ$ (for MNT curves only).

7.4 Summary

In this chapter, we discussed a novel yet important issue of fine-grained data access control for distributed storage in WSNs. To address the problem, we proposed a scheme in which each sensor node is assigned a set of attributes, and each user is assigned an access structure which designates the access capability of the user. The sensor data is encrypted under the attributes such that only the users with the intended access structure are able to decrypt. As the access structure is extremely expressive, we are able to control data access precisely, and thus achieve fine-grained access control. Moreover, our proposed scheme is able to provide security assurance such as resilience to user colluding and sensor compromising attacks as well as user

revocability. We also showed that the proposed scheme is able to support attribute change of sensor nodes and seamlessly integrate existing PH schemes to realize concealed data aggregation. Our experiment shows that the system complexity in our proposed scheme is reasonable in practical scenarios, especially for high-end sensor nodes.

Chapter 8

Conclusion and Future Work

8.1 Conclusion

In this dissertation, we addressed an important issue of secure data sharing on untrusted storage. We investigated the challenges pertained to this problem and proposed to exploit a novel PKC Attribute-Based Encryption (ABE) to provide cryptographically enforced data access control. With ABE, we are able to enjoy fine-grained access control. However, there are still several open security issues in state-of-the-art constructions of ABE. Toward providing a full-fledged cryptographic basis for secure data sharing on untrusted storage, we proposed three security-enhancing solutions for ABE: The first enhancement we made is to provide efficient user revocation in ABE. In this work, we particularly considered practical application scenarios in which semi-trustable proxy servers are available. With this assumption we uniquely combined the proxy re-encryption technique with ABE and enabled the authority to delegate most laborious tasks to proxy servers. Such a enhancement places minimal load on authority when revoking users. Our proposed scheme is provably secure against chosen ciphertext attacks. In our second enhance-

ment to ABE, we addressed key abuse attacks and proposed an abuse free KP-ABE (AFKP-ABE) scheme. (An abuse free CP-ABE (AFCP-ABE) scheme can also be built in the same way.) To defend against the key abuse attacks, we introduce hidden attributes in the system with which the tracing algorithm can identify any single pirate or partial colluding users. Our design enables black boxing tracing and does not require the well-formness of the user secret key. The complexity of AFKP-ABE in terms of ciphertext size and user secret keys size is just $\mathcal{O}(logN)$, where N is the total number of users. Our scheme is provably secure under DBDH assumption and D-Linear assumption. Our third enhancement is to provide better privacy preservation for ABE in terms of access policy information protection. In particular, we focused on CP-ABE and proposed two privacy-preserving CP-ABE schemes, in which data access structures are well protected from both the untrusted servers and all the users even under powerful attacks, e.g., colluding attacks. Numerical and experimental results show that our scheme is suitable for large-scale applications since its overhead is just linear to the number of attributes rather than the number of users.

With ABE and our enhancement schemes, we presented our solutions for secure data sharing for two specific application scenarios – Cloud Computing and Wireless Sensor Networks. For Cloud Computing, we proposed a scheme that provides fine-grained data access control for data owners in large-scale data centers outsourced to cloud. In doing so, we combined our enhanced ABE schemes with techniques such as dummy attribute and lazy re-encryption, and made it possible for both the data owner and users to delegate most computation-intensive operations to powerful cloud servers. Notably, our scheme can support resource-restrained cloud users such as mobile phones. Formal security proofs show that our proposed scheme is secure under standard cryptographic models. For Wireless Sensor Networks, we

proposed a fine-grained data access control scheme for distributed storage in WSNs. In our proposed scheme, each sensor node is assigned a set of attributes, and each user is assigned an access structure which designates the access capability of the user. Sensitive sensor data is encrypted with the attributes using ABE public keys and just allows users with the intended access structure to decrypt. In this way, confidentiality of sensitive data is protected even if the sensor node is compromised since the adversary can not obtain the data decryption key via reading the sensor memory. As user access structure can be extremely expressive, fine-grained access control is supported. In our proposed scheme, we revised a current ABE scheme to minimize the computation load on sensor nodes in case of user revocation. We also showed that our proposed scheme is able to support attribute change of sensor nodes and seamlessly integrate existing PH schemes to realize concealed data aggregation. Our experiment shows that the system load is affordable to contemporary sensor nodes.

8.2 Future Work

We identify three directions for future work for secure data sharing on untrusted storage as follows.

Decentralized Access Control In this dissertation, there is one cryptosystem in each data application and the data owner acts as the only authority in every cryptosystem. Users should possess a separate set of secret keys for each cryptosystem. In large-scale systems, it is desirable to provide decentralized access control in the sense that on one hand we enable users to access multiple cryptosystems using one set of secret keys, and on the other hand we allow the existence of multiple authorities in an application as well as encryption of data using public keys assigned

by multiple authorities. The concept of decentralized ABE [106] provides the cryptographic basis for this solution. However, existing schemes for decentralized ABE have various limitations in terms of the expressiveness of the access policy and etc. It is necessary to conduct further research to enhance decentralized ABE and hence provide decentralized data access control for untrusted storage.

Operation on Encrypted Data When encryption provides data confidentiality, it also greatly limits the flexibility of data operation. To address this issue, we need to combine ABE with cryptographic primitives such as searchable encryption [120–124], private information retrieval [125] and homomorphic encryption [126] to enable computations on encrypted data without decrypting. Moreover, as limitations in terms of data operations supported and efficiency still exist in these cryptographic primitives, another interesting future work would be taking into account information theoretic techniques from the areas such as database privacy.

Combining with Secure Computation In this dissertation, we occasionally assumed the servers to be honest-but-curious. In practical systems, it would be beneficial to remove this assumption to provide a stronger level of security protection. In order for doing so, one interesting future work would be integrating techniques from trusted computing [127] into the data access control mechanism.

Bibliography

[1] J. Anderson. Computer Security Technology Planning Study. Air Force Electronic Systems Division, Report ESD-TR-73-51, 1972. http://seclab.cs.ucdavis.edu/projects/history/.

[2] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *Proc. of SP'07*, Washington, DC, USA, 2007.

[3] M. Blaze, G. Bleumer, and M. Strauss. Divertible Protocols and Atomic Proxy Cryptography. In *Proc. of EUROCRYPT '98*, Espoo, Finland, 1998.

[4] A. Boldyreva, V. Goyal, and V. Kumar. Identity-based Encryption with Efficient Revocation. In *Proc. of CCS'08*, Alexandria, Virginia, USA, 2008.

[5] D. Boneh and M. Franklin. Identity-Based Encryption from The Weil Pairing. In *Proc. of CRYPTO'01*, Santa Barbara, California, USA, 2001.

[6] S. Yu, K. Ren, W. Lou, and J. Li. Defending Against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems. In *Proc. of Securecomm'09*, Athens, Greece, 2009.

[7] R. Canetti, S. Halevi, and J. Katz. Chosen Ciphertext Security from Identity Based Encryption. In *Proc. of EUROCRYPT'04*, Interlaken, Switzerland, 2004.

[8] R. Canetti and S. Hohenberger. Chosen-Ciphertext Secure Proxy Re-Encryption. In *Proc. of CCS'07*, New York, NY, USA, 2007.

[9] L. Cheung and C. Newport. Provably Secure Ciphertext Policy ABE. In *Proc. of CCS'07*, New York, NY, USA, 2007.

[10] R. H. Deng, J. Weng, S. Liu, and K. Chen. Chosen-Ciphertext Secure Proxy Re-encryption without Pairings. In *Proc. of CANS'08*, Berlin, Heidelberg, 2008.

[11] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Over-encryption: Management of Access Control Evolution on Outsourced Data. In *Proc. of VLDB'07*, Vienna, Austria, 2007.

[12] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-Based Encryption for Fine-grained Access Control of Encrypted Data. In *Proc. of CCS'06*, Alexandria, Virginia, USA, 2006.

[13] S. Yu, K. Ren, and W. Lou. Attribute-Based On-Demand Multicast Group Setup with Membership Anonymity. In *Proc. of SecureComm'08*, Istanbul, Turkey, 2008.

[14] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable Secure File Sharing on Untrusted Storage. In *Proc. of FAST'03*, Berkeley, California, USA, 2003.

[15] J. Li, K. Ren, B. Zhu, and Z. Wan. Privacy-Aware Attribute-Based Encryption with User Accountability. In *Proc. of ISC'09*, Pisa, Italy, 2009.

[16] X. Liang, Z. Cao, H. Lin, and J. Shao. Attribute Based Proxy Re-encryption with Delegating Capabilities. In *Proc. of ASIACCS'09*, Sydney, Australia, 2009.

[17] S. Yu, K. Ren, and W. Lou. Attribute-Based Content Distribution with Hidden Policy. In *Proc. of NPSEC'08*, Orlando, Florida, USA, 2008.

[18] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Atrribute-Based Systems. In *Proc. of CCS'06*, New York, NY, USA, 2006.

[19] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Proc. of EUROCRYPT'05*, Aarhus, Denmark, 2005.

[20] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," University of California, Berkeley, Tech. Rep. USB-EECS-2009-28, Feb 2009.

[21] Amazon Web Services (AWS), Online at http://aws. amazon.com.

[22] Google App Engine, Online at http://code.google.com/appengine/.

[23] Microsoft Azure, http://www.microsoft.com/azure/.

[24] 104th United States Congress, "Health Insurance Portability and Accountability Act of 1996 (HIPPA)," Online at http://aspe.hhs.gov/admnsimp/pl104191.htm, 1996, last access: July 16, 2009.

[25] H. Harney, A. Colgrove, and P. D. McDaniel, "Principles of policy in secure groups," in *Proc. of NDSS'01*, 2001.

[26] P. D. McDaniel and A. Prakash, "Methods and limitations of security policy reconciliation," in *Proc. of SP'02*, 2002.

[27] T. Yu and M. Winslett, "A unified scheme for resource protection in automated trust negotiation," in *Proc. of SP'03*, 2003.

[28] J. Li, N. Li, and W. H. Winsborough, "Automated trust negotiation using cryptographic credentials," in *Proc. of CCS'05*, 2005.

[29] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "Sirius: Securing remote untrusted storage," in *Proc. of NDSS'03*, 2003.

[30] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," in *Proc. of NDSS'05*, 2005.

[31] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. of ESORICS '09*, 2009.

[32] L. Youseff, M. Butrico, and D. D. Silva, "Toward a unified ontology of cloud computing," in *Proc. of GCE'08*, 2008.

[33] D. Sheridan, "The optimality of a fast CNF conversion and its use with SAT," in *Proc. of SAT'04*, 2004.

[34] D. Naor, M. Naor, and J. B. Lotspiech, "Revocation and tracing schemes for stateless receivers," in *Proc. of CRYPTO'01*, 2001.

[35] M. Atallah, K. Frikken, and M. Blanton, "Dynamic and efficient key management for access hierarchies," in *Proc. of CCS'05*, 2005.

[36] B. Chor, A. Fiat, and M. Naor, "Tracing traitors," in *CRYPTO'94*. London, UK: Springer-Verlag, 1994, pp. 257–270.

[37] D. Boneh and M. K. Franklin, "An efficient public key traitor tracing scheme," in *CRYPTO'99*. London, UK: Springer-Verlag, 1999, pp. 338–353.

[38] A. Kiayias and M. Yung, "Traitor tracing with constant transmission rate," in *EUROCRYPT'02*. London, UK: Springer-Verlag, 2002, pp. 450–465.

[39] D. Boneh, A. Sahai, and BrentWaters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT'06*. London, UK: Springer-Verlag, 2006.

[40] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *ACNS'08*. LNCS 5037, 2008, pp. 111–129.

[41] A. Kapadia, P. Tsang, and S. Smith, "Attribute-based publishing with hidden credentials and hidden policies," in *NDSS'07*. LNCS 5037, 2007, pp. 179–192.

[42] J. Katz, A. Sahai, and B.Waters, "Predicate encryption supporting disjunctions, polynomial equations, and inner products," in *Eurocrypt'08*. LNCS 4965, 2008, pp. 146–162.

[43] J. Li, K. Ren, and K. Kim, "A2be: Accountable attribute-based encryption for abuse free access control," Cryptology ePrint Archive, Report 2009/118, 2009, http://eprint.iacr.org.

[44] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *CRYPTO'04*. LNCS 3152, 2004, pp. 41–55.

[45] A. Barth, D. Boneh, and B. Waters. Privacy in encrypted content distribution using private broadcast encryption. In *Financial Cryptography '06*, pages 52–64, 2006.

[46] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. *CRYPTO '05*, pages 258–275, 2005.

[47] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure lnternet multicast using boolean function minimization techniques. *INFOCOM '99*, pages 689–698 vol. 2, 1999.

[48] L. Cheung, J. A. Cooley, R. Khazan, and C. Newport. Collusion-resistant group key management using attribute-based encryption. Cryptology ePrint Archive, Report 2007/161, 2007.

[49] C.D.M. Cordeiro, H. Gossain, and D.P. Agrawal. Multicast over wireless mobile ad hoc networks: Present and future directions. *IEEE Network*, 17:52–59, 2003.

[50] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO '93*, pages 480–491, 1994.

[51] C. Grosch. Framework for anonymity in ip-multicast environments. *GLOBECOM '00.*, pages 365–369 vol. 1, 2000.

[52] M. Keller, T. Kerins, and W. Marnane. FPGA implementation of a GF(2^{4m}) multiplier for use in pairing based cryptosystems. *Field Programmable Logic and Applications*, pages 594–597, 2005.

[53] NIST. Secure hash standard. *Federal Information Processing Standard, FIPS-180-1*, April,1995.

[54] A. Pfitzmann and M. K. Anonymity, unobservability, and pseudeonymity − a proposal for terminology. In *International workshop on Designing privacy enhancing technologies*, pages 1–9, 2001.

[55] V. Upkar. Multicast over wireless networks. *Commun. ACM*, 45(12):31–37, 2002.

[56] M. Waldvogel, G. Caronni, D. Sun, N. Weiler, and B. Plattner. The versakey framework: versatile group key management. *Selected Areas in Communications, IEEE Journal on*, 17(9):1614–1631, Sep. 1999.

[57] N. Weiler. Secure anonymous group infrastructure for common and future internet applications. *ACSAC 2001. Proceedings 17th Annual*, pages 401–410, 2001.

[58] S. Yu, K. Ren, and W. Lou, "FDAC: Toward fine-grained distributed data access control in wireless sensor networks," in *IEEE INFOCOM'09*, Brazil, Apr. 2009.

[59] I. F. Akyildiz and I. H. Kasimoglu, "Wireless sensor and actor networks: Research challenges," *Ad Hoc Networks Journal (Elsevier)*, vol. 2, no. 4, pp. 351–367, Oct. 2004.

[60] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *IEEE Commun. Mag.*, vol. 40, no. 8, pp. 102–116, Aug. 2002.

[61] C.-Y. Chong and S. P. Kumar, "Sensor networks: Evolution, opportunities, and challenges," *Proc. of IEEE*, Aug 2003.

[62] A. Cerpa, J. Elson, D. Estrin, L. Girod, M. Hamilton, and J. Zhao, "Habitat monitoring: Application driver for wireless communications technology," in *ACM SIGCOMM Workshop Data Comm. Latin America and the Caribbean*, Costa Rica, Apr. 2001.

[63] D. Estrin, D. Culler, and K. Pister, "Connecting the physical world with pervasive networks," *IEEE Percasive Computing*, Jan-Mar 2002.

[64] B. Thuraisingham, "Secure sensor information management and mining," *Signal Processing Magazine, IEEE*, vol. 21, no. 3, pp. 14–19, May 2004.

[65] A. Banerjee, A. Mitra, W. Najjar, D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "Rise co-s : High performance sensor storage and co-processing architecture," in *SECON'05*, Santa Clara, California, 2005.

[66] A. Mitra, A. Banerjee, W. Najjar, D. Zeinalipour-Yazti, V. Kalogeraki, and D. Gunopulos, "High-performance low power sensor platforms featuring gigabyte scale storage," in *MobiQuitous'05*, San Diego, CA, 2005.

[67] G. Mathur, P. Desnoyers, D. Ganesan, and P. Shenoy, "Capsule: An energy-optimized object storage system for memory-constrained sensor devices," in *ACM Sensys*, November 2006.

[68] D. Zeinalipour-Yazti, V. Kalogeraki, D. Gunopulos, A. Mitra, A. Banerjee, and W. Najjar, "Towards in-situ data storage in sensor databases," in *PCI'05, LNCS 3746*, Volos, Greece, 2005, pp. 36–46.

[69] S. Ratnasamy, B. Karp, L. Yin, F. Yu, D. Estrin, R. Govindan, and S. Shenker, "GHT: A geographic hash table for data-centric storage," in *WSNA'02*, Sep. 2002.

[70] J. Newsome and D. Song, "GEM:graph embedding for routing and data-centric storage in sensor networkswithout geographic information," in *SenSys'03*, Nov. 2003.

[71] J. Girao, D.Westhoff, E. Mykletun, and T. Araki, "Tinypeds: Tiny persistent encrypted data storage in asynchronous wireless sensor networks," *Ad Hoc Networks, Elsevier*, vol. 5, no. 7, p. 1073C1089, 2007.

[72] R. D. Pietro, L. V. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Catch me (if you can): Data survival in unattended sensor networks," in *IEEE PerCom'08*, Mar. 2008.

[73] Q. Wang, K. Ren, W. Lou, and Y. Zhang, "Dependable and secure sensor data storage with dynamic integrity assurance," in *IEEE INFOCOM'09*, Brazil, Apr. 2009.

[74] S. Zhu, S. Xu, S. Setia, and S. Jajodia, "LHAP: A lightweight hop-by-hop authentication protocol for ad-hoc networks," in *ICDCSW'03*, Providence, Rhode Island, USA, May 2003.

[75] F. Ye, H. Luo, S. Lu, and L. Zhang, "Stastical en-route filtering of injected false data in sensor networks," in *IEEE INFOCOM'04*, Hong Kong, China, Mar. 2004.

[76] H. Chan, A. Perrig, and D. Song, "Random key predistribution schemes for sensor networks," in *IEEE S& P'03*, Oakland, CA, May 2003.

[77] M. Shao, S. Zhu, W. Zhang, and G. Cao, "pDCS: Security and Privacy Support for Data-Centric Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 8, pp. 1023–1038, 2009.

[78] R. D. Pietro, L. Mancini, C. Soriente, A. Spognardi, and G. Tsudik, "Playing hide-and-seek with a focused mobile adversary in unattended wireless sensor networks," *Ad Hoc Networks*, vol. 7, no. 8, pp. 1463–1475, 2009.

[79] W. Zhang, H. Song, S. Zhu, and G. Cao, "Least Privilege and Privilege Deprivation: Towards Tolerating Mobile Sink Compromises in Wireless Sensor Networks," *ACM Transactions on Sensor Networks*, vol. 4-4, Nov. 2008.

[80] H. Wang and Q. Li, "Distributed User Access Control in Sensor Networks," in *IEEE DCOSS*, Jun. 2006.

[81] H. Wang, B. Sheng, C. C. Tan, and Q. Li, "Comparing Symmetric-key and Public-key based Schemes in Sensor Networks: A Case Study for User Access Control," in *IEEE ICDCS*, Jun. 2008.

[82] C. C. Tan, H. Wang, S. Zhong, and Q. Li, "Body Sensor Network Security: An Identity-Based Cryptography Approach," in *ACM WiSec*, Mar.-Apr. 2008.

[83] N. Subramanian, C. Yang, and W. Zhang, "Securing distributed data storage and retrieval in sensor networks," in *Pervasive and Mobile Computing Journal (Special Issue for PerCom 2007)*, November 2007.

[84] M. Albrecht, C. Gentry, S. Halevi, and J. Katz, "Attacking cryptographic schemes based on "perturbation polynomials"," in *Cryptology ePrint Archive Report 2009/098*, 2009.

[85] B. Karp and H. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *Mobicom'00*, Aug 2000.

[86] M. Goodrich, J. Sun, and R. Tamassia, "Efficient tree-based revocation in groups of low-state devices," in *Advances in Crytology CRYPTO'04*, 2004.

[87] B. Krishnamachari, D. Estrin, and S. Wicker, "The impact of data aggregation in wireless sensor networks," in *In International Workshop on Distributed Event-Based Sytems*, Austria, Jul. 2002.

[88] J. Girao, D. Westhoff, and M. Schneider, "CDA: concealed data aggregation for reverse multicast traffic in wireless sensor networks," in *IEEE ICC'05*, Korea, May. 2005.

[89] S. Peter, D. Westhoff, and C. Castelluccia, "A survey on the encryption of convergecast-traffic with in-network processing," *IEEE Transactions on Dependable and Secure Computing*, 2009.

[90] C. Castelluccia, E. Mykletun, and G. Tsudik, "Efficient aggregation of encrypted data in wireless sensor networks," in *IEEE MobiQuitous'05*, Jul. 2005.

[91] S. A. Camtepe and B. Yener, Key distribution mechanisms for wireless sensor networks: a survey, Rensselaer Polytechnic Inst., Comput. Sci. Dept., Troy, NY, Tech. Rep. TR-05-07, 2005.

[92] K. Ren, W. Lou, and Y. Zhang, "LEDS: Providing location-aware end-to-end data security in wireless sensor networks," in *IEEE INFOCOM'06*, Barcelona, Spain, Apr. 2006, pp. 1–12.

[93] ——, "Multi-user broadcast authentication in wireless sensor networks," in *IEEE SECON'07*, Jun. 2007.

[94] National Institute of Standards and Technology. Recommended elliptic curves for federal government use, August 1999.

[95] Certicom Research. Standards for efficient cryptography C SEC 2: Recommended elliptic curve domain parameters. http://www.secg.org/collateral/sec2_final.pdf, September 2000.

[96] PBC Library. http://crypto.stanford.edu/pbc/times.html.

[97] TinyECC Library. http://discovery.csc.ncsu.edu/software/TinyECC/index.html.

[98] Imote2: High-performance wireless sensor network node. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/Imote2_Datasheet.pdf.

[99] TelosB mote platform. http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/TelosB_Datasheet.pdf.

[100] ACL. http://en.wikipedia.org/wiki/Access_control_list

[101] H. M. Levy, "Capability-Based Computer Systems", Digital Equipment Corporation 1984. ISBN 0-932376-22-3.

[102] NIST. "Role Based Access Control (RBAC) and Role Based Security". http://csrc.nist.gov/groups/SNS/rbac/

[103] R. Ostrovsky, A. Sahai, and B. Waters. "Attribute-based encryption with non-monotonic access structures". In *Proc. of CCS'06*, New York, NY, 2007.

[104] M. Chase. "Multi-authority attribute based encryption". In *Proc. of TCC'07*, Amsterdam, Netherlands, 2007.

[105] H. Lin, Z. Cao, X. Liang and J. Shao. "Secure threshold multi authority attribute based encryption without a central authority", In *Proc. of INDOCRYPT'08* , Kharagpur, India, 2008.

[106] A. Lewko and B. Waters, "Decentralizing Attribute-Based Encryption", http://eprint.iacr.org/2010/351.

[107] B. Waters, "Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization", http://eprint.iacr.org/2008/290.

[108] V. Goyal, A. Jain, O. Pandey and A. Sahai, "Bounded Ciphertext-Policy Attribute based Encryption", In *Proc. of ICALP'08*, Reykjavik, Iceland, 2008

[109] M. J. Hinek, S. Jiang, R. Safavi-Naini, and S. F. Shahandashti, "Attribute-Based Encryption with Key Cloning Protection", http://eprint.iacr.org/2008/478

[110] Jin Li, Qian Wang, Cong Wang, and Kui Ren, "Enhancing Attribute-based Encryption with Attribute Hierarchy," In *Proc. of ChinaCom'09*, Xi'an, China, 2009.

[111] A. Kapadia , P. Tsang, and S. W. Smith, "Attribute-Based Publishing with Hidden Credentials and Hidden Policies," In *Proc. of NDSS'07*, San Diego, CA, 2007.

[112] L. Ballard, M. Green, B. Medeiros, F. Monrose. "Correlation-Resistant Storage via Keyword-Searchable Encryption". http://eprint.iacr.org/2005/417

[113] A. Miyaji, M. Nakabayashi, and S. Takano. "New explicit conditions of elliptic curve traces for FR-reduction". IEICE Trans. Fundamentals, E84-A(5):1234C43, May 2001.

[114] D. Boneh, B. Lynn, and H. Shacham. "Short signatures from the Weil pairing". In *Proc. of Asiacrypt'01*, Gold Coast, Australia, 2001.

[115] D. R. Stinson. "Cryptography: Theory and Practice", CRC Press, 2002

[116] D. Boneh, "The Decision DiffieCHellman Problem", In *Proc. of ANTS'98*, Portland, Oregon, 1998.

[117] W. Diffie and M. Hellman. "New directions in cryptography". *IEEE Transactions on Information Theory*, IT No.2(6):644C654, November 1976.

[118] D. Boneh, C. Gentry, B. Lynn and H. Shacham, "Aggregate and Verifiably Encrypted Signatures from Bilinear Maps", In *Proc. of Eurocrypt'03*, Warsaw, Poland, 2003.

[119] S. Yu, K. Ren, and W. Lou, "FDAC: Toward Fine-Grained Distributed Data Access Control in Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, 16 Jun. 2010. http://doi.ieeecomputersociety.org/10.1109/TPDS.2010.130

[120] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. "Public Key Encryption with Keyword Search". In *Proc. of EUROCRYPT'04* , 2004.

[121] D. Boneh and B. Waters. "Conjunctive, Subset, and Range Queries on Encrypted Data". In *Proc. of TCC'07*, Amsterdam, Netherlands, 2007.

[122] E. Shen, E. Shi, and B. Waters. "Predicate Privacy in Encryption Systems". In *Proc. of TCC'09*, San Francisco, CA, 2009.

[123] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig. "Multi-Dimensional Range Query over Encrypted Data". In *Proc. of SP'07*, Oakland, CA, 2007.

[124] D. Song, D. Wagner, and A. Perrig. "Practical Techniques for Searches on Encrypted Data". In *Proc. of SP'00*, Oakland, CA, 2000.

[125] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. "Private Information Retrieval". J. ACM, 45, 6 (1998), 965-981.

[126] C. Gentry. "Fully Homomorphic Encryption Using Ideal Lattices". In *Proc. of STOC'09*, Bethesda, Maryland, 2009.

[127] Trusted Computing Group. http://www.trustedcomputinggroup.org/