Cryptographic Enforcement of Attribute-based Authentication

Huihui Yang

Cryptographic Enforcement of Attribute-based Authentication

Doctoral Dissertation for the Degree *Philosophiae Doctor (PhD)* in
Information and Communication Technology

University of Agder
Faculty of Engineering and Science
2016

Doctoral Dissertations at the University of Agder 133
ISBN: 978-82-7117-826-0
ISSN: 1504-9272

Printed in the Printing Office, University of Agder
Kristiansand

Preface and Acknowledgments

This dissertation is the research result during my PhD study, from August 2012 to January 2016 in University of Agder, Norway. The experience and training during my PhD allow me to learn research and academic paper writing skills and finally grow as a research scientist.

First and foremost I would like to express my special appreciation to my main supervisor Dr. Vladimir A. Oleshchuk, who has been a tremendous mentor for me. As a professor with broad knowledge in information security, he has guided me into my current research area and given me enough space and time to find where my real interest lies. He is always patient, kind and willing to discuss with me about my ideas. I appreciate all his contributions of time, professional suggestions and proof-readings for papers and this thesis. At the same time, I want to thank my second supervisor, Dr. Geir M. Køien. He has a special way of inspiring his PhD students by asking lots of questions, which makes us reflect more on details and impresses me with his serious attitude towards research. He is also a nice person and always willing to help.

I would like to acknowledge all supports received from the administration and all faculty from ICT departments, especially Tonje Sti and Emma Elizabeth Horneman. Tonje helped me to settle down when I just arrived in Norway and she is a person that I can turn for help even for trivial things in life. As a PhD program coordinator, Emma has helped me so much in these three years. In addition, I would like to express my appreciations for all the help and kindness that I have received.

I would also like to thank all PhD fellows in ICT and other departments I know. I have spent so much happy time with them and we have encouraged each other through difficulties in PhD research and life. They have made my life more colorful and have brought me so many laughters. Many thanks to my friend Alex P. Hage for all his help and the good, happy time we have spent together, to my best old friend Jing Xu for our long years' friendship, her understandings and support for everything, to the couple of Xuan Zhang and Lei Jiao for their warmth like a family, to Indika A. M. Balapuwaduge, Martin W. Gerdes and all my other friends.

The last but not the least, my most sincerely gratitudes go to my family for their love, support and encouragement in my lifetime.

Huihui Yang
January 2016
Grimstad, Norway

Summary

This dissertation investigates on the cryptographic enforcement about attribute-based authentication (ABA) schemes. ABA is an approach to authenticate users via attributes, which are properties of users to be authenticated, environment conditions such as time and locations. By using attributes in place of users' identity information, ABA can provide anonymous authentication, or more specifically, ABA enables to keep users anonymous from their authenticators. In addition, the property of least information leakage provides better protection for users' privacy compared with public key based authentication approaches. These properties make it possible to apply ABA schemes in privacy preserving scenarios, for instance, cloud-based applications.

The most important security requirements of ABA schemes consist of anonymity, traceability, unforgeability, unlinkability and collision resistance. In this dissertation, we combine these security requirements with other properties such as hierarchy to divide ABA schemes into different categories, based on which we use examples to demonstrate how to construct these schemes cryptographically. The main contributions of this dissertation include the following aspects:

• We categorize ABA schemes into different types and describe their structures as well as workflows, such that readers can gain a big picture and a clear view of different ABA schemes and their relations. This categorization serves as a guideline how to design and construct ABA schemes.

• We provide two examples to demonstrate how to construct ciphertext-policy attribute-based authentication (CP-ABA) schemes via two different approaches. Different from key-policy attribute-based authentication (KP-ABA) schemes, attribute keys generated in CP-ABA schemes are comparatively independent of relations among attributes. Thus compared with KP-ABA, CP-ABA extends the flexibility and usage scope of ABA schemes.

• We extend the core ABA schemes to hierarchical ABA (HABA) schemes by adding the property of hierarchy. Then we propose two different types of hierarchical structures, i.e., user related hierarchical ABA (U-HABA) and attribute related hierarchical ABA (A-HABA). According to these two hierarchical structures, an example is provided for each type to show how to use cryptographic primitives to build HABA schemes.

All ABA schemes discussed above and proposed in this dissertation can be implemented to assist users to achieve anonymous authentication from their authenticators. Therefore, these schemes can offer more opportunities to protect users' privacy, for example, in attribute-based access control (ABAC) and cloud-based services.

Contents

List of Figures

List of Tables

Abbreviations

ABA	Attribute-based authentication
ABAC	Attribute-based access control
ABE	Attribute-based encryption
ABID	Attribute-based identification
ABS	Attribute-based signature
AC	Access control
ACM	Access control mechanism
A-HABA	Attribute related hierarchical attribute-based authentication
CPA	Chosen plaintext attack
CP-ABA	Ciphertext-policy attribute-based authentication
CP-ABE	Ciphertext-policy attribute-based encryption
DAC	Discretionary access control
DeLP	Decision linear Diffie-Hellman problem
DH	Diffie-Hellman
DLE	Decision linear Diffie-Hellman based encryption
DLP	Discrete logarithm problem
HABA	Hierarchical attribute-based authentication
HABAC	Hierarchical attriubte-based access control
HABE	Hierarchical attribute-based encryption
HASBE	Hierarchical attribute-set-based-encryption
HGABAC	Hierarchical group and attribute-based access control
HIBE	Hierarchical identity-based encryption
IBE	Identity-based encryption
IND-CPA	In-distinguishable chosen plaintext attack
KP-ABA	Key-policy attribute-based authentication
KP-ABE	Key-policy attribute-based encryption
PDP	Policy decision point

PEP	Policy enforcement point
PKI	Public key infrastructure
MAC	Mandatory access control
q-SDH	q-Strong Diffie-Hellman problem
RBAC	Role-based access control
U-HABA	User related hierarchical attribute-based authentication

Chapter 1

Introduction

In this chapter, we first discuss the motivations why we conduct the research on the cryptographic enforcement on attribute-based authentication (ABA) in Section 1.1, which is followed by some background knowledge in Section 1.2. Then we express the problem statements and our main contributions in Section 1.3. Section 1.4 is a brief introduction about how this dissertation is organized and the research topic in each chapter.

1.1 Motivations

Currently there are a lot of cloud-based services, such as cloud storage [1, 2], cloud computing [3, 4] and data outsourcing [5]. Consider Dropbox [6, 7], one of the most popular cloud storage applications. By using Dropbox, users can create a special folder on their computers and save files in this folder. Later, users can access them elsewhere from mobiles, iPad or other computers. In addition, users can share files saved in Dropbox folders with other people. Cloud users can also use cloud resources to perform complicated calculations. When utilizing cloud facilities, on the one side, cloud users want to keep their data safe and secret from other Internet users and even the service providers. On the other side, users may also need to gain more fine-grained control over their data, for example, how their data can be accessed.

Many cloud services have an increasing demand and emphasis on protecting both the privacy and security of users and their data, but how to achieve it? One possible solution is attribute-based access control (ABAC) [8]. Access control (AC) [9, 10] is an approach to control the authorization of users' or subjects' access

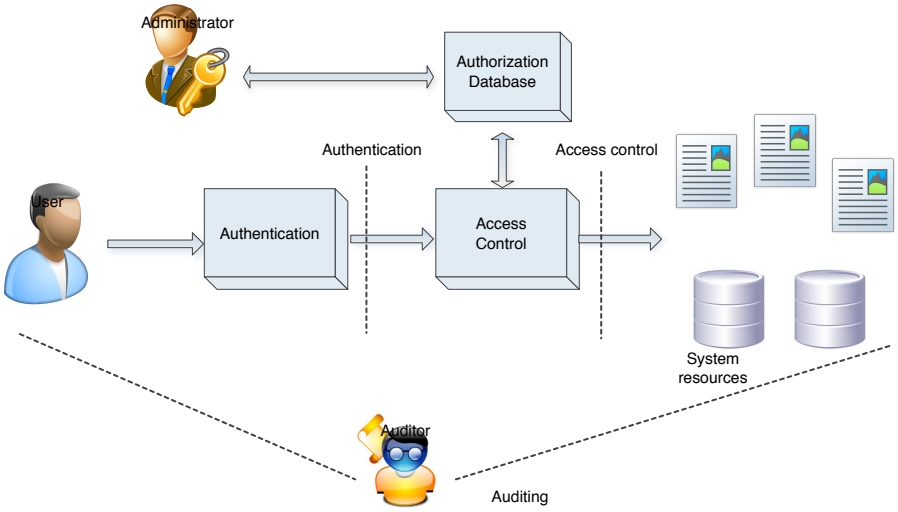


Figure 1.1: Access control mechanism

to resources and aims at protecting the security of the resources or objects. The definition of AC from [8] is as follows. "Access control mechanism (ACM): The logical component that serves to receive the access request from the subject, to decide, and to enforce the access decision." We can illustrate ACM as described in Fig. 1.1. The administrator manages AC policies in the system and decides whether users should be granted with specified accessing rights. When a user or subject wants to access system resources, she sends an access request and then the administrator checks whether the user is allowed to access the resources. If the user's request does not conflict with AC policies, then the user will be allowed to carry out the access request. Otherwise, user's request will be denied. Only if a user's access request is allowed, the user may have the chance to access resources. However, the enforcement of the access request also depends on factors that are comparatively independently of the access policies, for example, system time and locations.

ABAC is an ACM based on attributes, which requests properties of subjects and objects. A typical ABAC system can be illustrated in Fig. 1.2. When a subject or a user wants to access an object or resources, it sends an access request to the system together with its attributes to the system, where attributes are properties of the subject, such as addresses, ages, names and so on. After receiving the request and attributes from the subject, the policy decision point (PDP) combines the subject's attributes, the object's attributes and environmental conditions, and then checks whether it satisfies AC policies. If so, the subject's access request will be

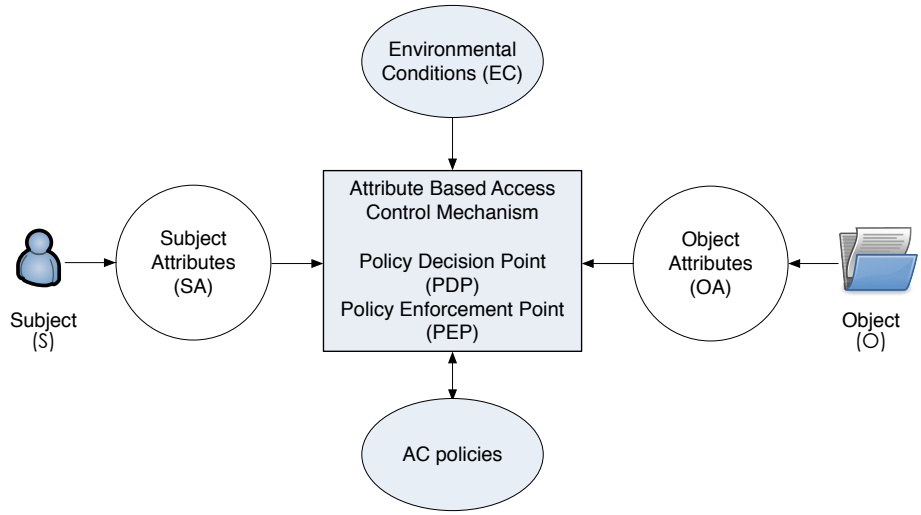


Figure 1.2: Attribute-based access control mechanism

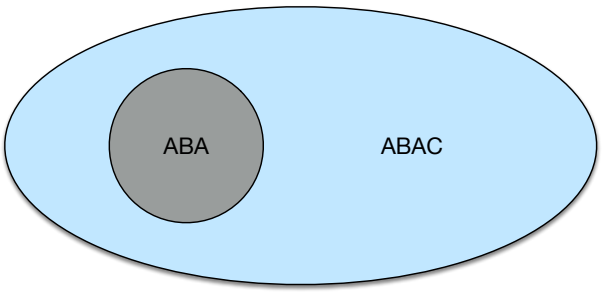


Figure 1.3: Relations between ABA and ABAC

allowed and it will be enforced by the policy enforcement point (PEP). During the process described above, PDP's decision making part can be considered as a part of authentication, while the AC policy enforcing part by PEP be can considered as authorization. As illustrated in Fig. 1.3, ABA is only part of ABAC and the authentication result of ABA is an important factor to decide whether a user's access request can be enforced or not. In this dissertation, our main focus is ABA rather than the whole scope of ABAC.

As far as we know, research on ABA mainly includes the general structure how an ABA system can be built [11], the languages how to express AC policies [12] and cryptographic enforcement [13]. For most current approaches to build ABA schemes, AC polices are expressed as attribute requirements, which are combined with group signatures to construct ABA schemes [13]. Since there have already

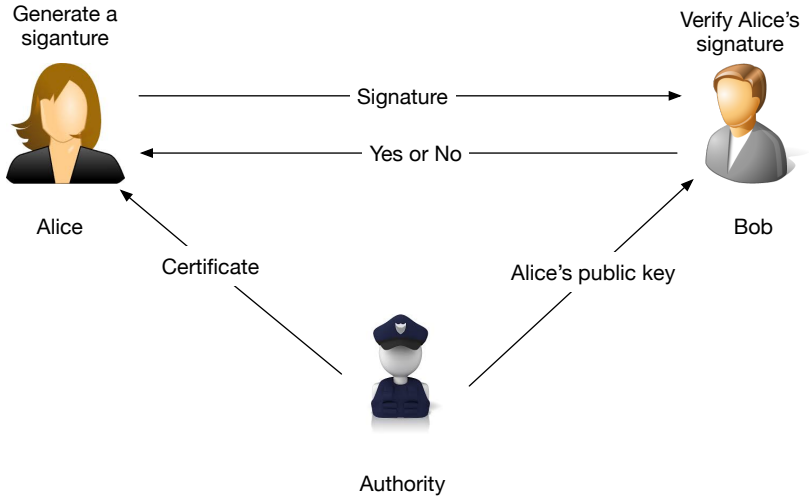
been lots of research on the cryptographic construction of attribute-based signatures (ABS) [14, 15] and attribute-based encryption (ABE) [16, 17], it must be a good choice to utilize these results to construct ABA schemes. Besides, there is also a drawback on the group signature based ABA schemes. The way how to represent attribute requirement is not flexible enough. If we need to change the attribute requirements, the whole system should be rebuilt which will surely consume extra system resources. To solve the above two issues, research in this dissertation is conducted.

1.2 Background Knowledge

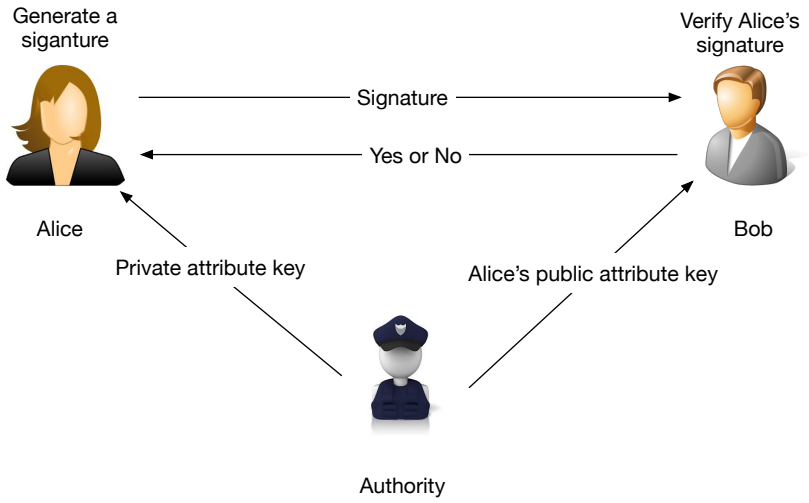
One of the well known authentication approaches is based on the public key infrastructure (PKI) [18, 19]. Assume Alice is to be authenticated by Bob. For PKI-based authentication, as illustrated in Fig. 1.4a , Alice first obtains a pair of public and private keys from the authority. Then Alice uses her private key to generate a signature and sends it to Bob. After Bob receives the signature from Alice, he retrieves Alice's pubic key from the authority and uses it to check whether Alice's signature is valid or not. Alice will be successfully authenticated only if her signature is valid.

Different from PKI-based authentication mechanism, ABA is based on multiple factors, such as subjects' attributes and objects' attributes. Fig. 1.4b and Fig. 1.5 explain the main idea and the workflow of ABA respectively. The way how ABA proceeds can be summarized into the following steps:

1. A user obtains authorized attribute keys from the authority.
2. The user sends an authentication request to the authenticator.
3. After receiving the authentication request from the user, the authenticator responds with the attribute requirements.
4. If the user owns the required attributes, he or she generates a signature with the required attribute keys and sends the signature to the authenticator.
5. To verify the signature, the authenticator retrieves some information from the authority first, such as the user's attribute public keys, revocation information and so on. Next the authenticator checks whether the signature is valid or not.
6. The authenticator responses the user with the verification result, which is normally yes or no.



(a) PKI-based authentication



(b) ABA

Figure 1.4: Comparison between PKI-based authentication and ABA

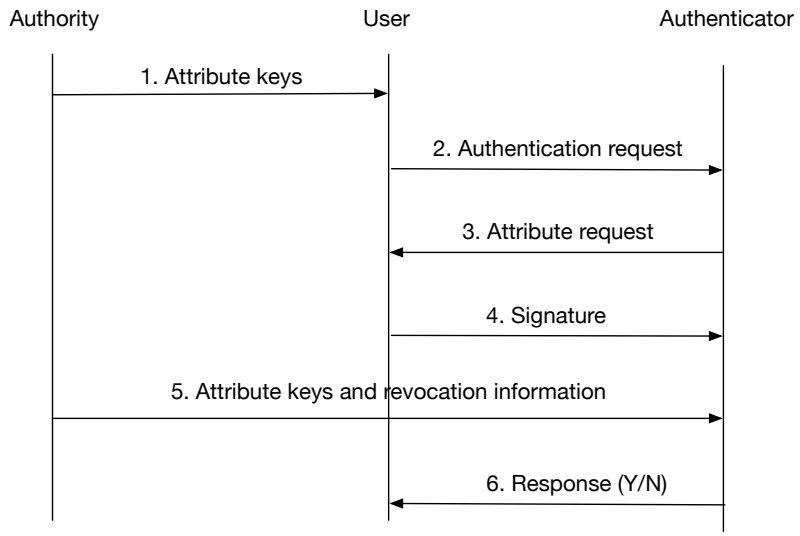


Figure 1.5: Message flow of attribute-based authentication

Compared with the PKI-based authentication, ABA has three main differences. First of all, Alice generates a signature based on specific attribute requirements. Secondly, instead of using Alice's private key which represents her identity, Alice uses a set of attribute keys to generate a signature, where these attributes are the required ones from Bob. Finally, different from the one-to-one relation between Alice's private key and public key in the PKI-based authentication, the relation between Alice's public attribute keys and private attribute keys may be one-to-one or one-to-multiple.

1.3 Problem Statement and Main Contributions

This dissertation focuses on the cryptographic enforcement of ABA schemes [13, 20], and the main topics can be summarized as follows:

1. When and under which situations do we need ABA?

2. How do ABA schemes work and what properties do they have?

3. How to construct ABA schemes cryptographically?

4. When do we need hierarchical attribute-based authentication (HABA) schemes and how to construct them?

5. When do we need traceable ABA schemes, how to construct untraceable ABA schemes based on traceable ones?

The main contributions consists of the following.

1. We categorize ABA schemes into different types according to two criteria in Chapter 3, which are their security requirements and the way how to generate attribute keys respectively.

2. We demonstrate how to construct ciphertext-policy attribute-based authentication (CP-ABA) schemes by two different approaches by using different cryptographic primitives in Chapter 4.

3. We provide two examples to explain how to implement user- and attribute-related hierarchy in ABA schemes in Chapter 5.

4. We research on untraceable and traceable ABA schemes as well as their relations in Chapter 6.

5. We survey on ABA related research results, discuss open problems and propose possible solutions in Chapter 7.

1.4 Organization of the Dissertation

The rest of this dissertation is organized as follows:

In Chapter 2, we elaborate on possible usage scenarios and applications.

In Chapter 3, we categorize ABA schemes according to two different criteria, based on which we describe the workflow of ABA schemes and their security and privacy properties.

In Chapter 4, we use two examples to demonstrate how to construct CP-ABA schemes cryptographically. These two examples are based on the work in [20] and [21] respectively.

In Chapter 5, we present how to apply user- and attribute-related hierarchy in ABA schemes and provide two examples to demonstrate how to construct them. This chapter is an extension of the work in [22].

In Chapter 6, we use two examples to study on the relations between traceable and untraceable ABA schemes. These two examples are based on the two schemes proposed in Chapter 4.

In Chapter 7, we first conclude our work and main contributions in this dissertation in Section 7.1. Then in Section 7.2, we study on several open problems related to ABA schemes, including attribute tree building, general frameworks to construct ABA schemes, security models, hierarchy, traceability and revocation. Next we propose possible solutions to these open problems. The main results in this chapter are based on paper [23].

Chapter 2

Usage Scenarios and Application Requirements

In this Chapter, we first describe several usage scenarios and applications where ABA schemes can be applied. Then we summarize under which circumstances different ABA schemes can be used.

2.1 Usage Scenarios

To better understand ABA schemes, we start with three different usage scenarios to describe properties and advantages of ABA schemes.The first scenario is data sharing among multiple organizations and we show how complex can the situation be without applying ABA. The second scenario is a case study related to data outsourcing and we express the reason why ABA schemes with specific security requirements are needed. The third scenario is in e-Health. In this scenario, we explain the general structure of e-Health applications and how ABA can be applied to solve privacy issues in an e-Health system.

2.1.1 Usage Scenario 1: Multi-organization Cooperation

Let us consider three companies C_1, C_2 and C_3 (Fig. 2.1) that have to cooperate and share some data with each other. Suppose the companies have their own separate data manage systems. We assume that as long as a user belongs to any of these three companies, the user should be allowed to access the shared data. Consider the case when an employee E_1 belonging to C_1 who wants to access the shared data

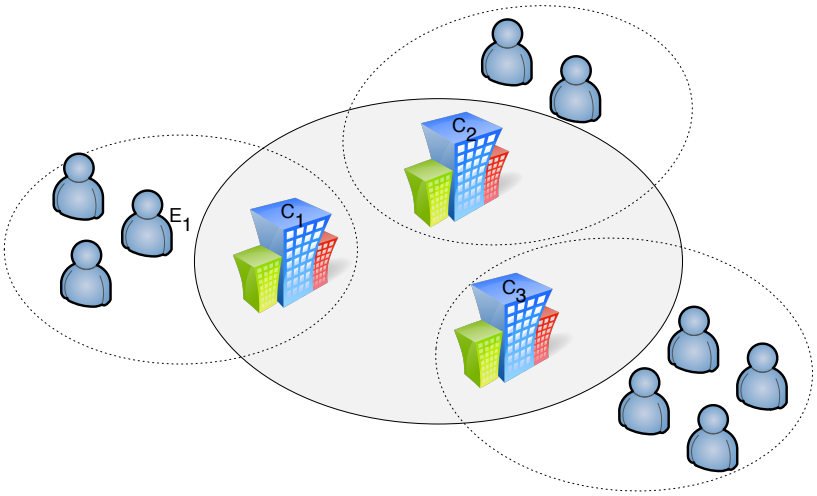


Figure 2.1: Usage scenario 1: multi-organization cooperation

from companies C_2 and C_3. This issue can be solved by a mechanism based on role-based access control (RBAC) or ABA. In the following, we will describe both approaches and compare them.

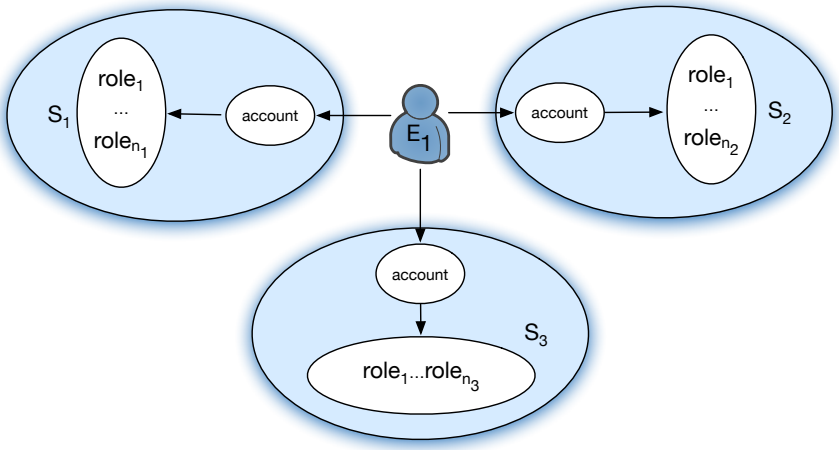
Assume that the RBAC systems of companies C_1, C_2 and C_3 are denoted by S_1, S_2 and S_3 respectively. The mechanism to solve the multi-cooperation issue for usage scenario 1 can be described in Fig. 2.2a. Since E_1 is an employee of company C_1, E_1 must have already had an account in S_1. After E_1 logs in S_1, it will be assigned with a role such that it can access data in S_1. To access the shared data owned by companies C_2 and C_3, E_1 should first have an account in systems S_2 and S_3 respectively, and each account should be associated with several roles based on which E_1 can access data in systems S_2 and S_3. Suppose there are n_i $(1 \leq i \leq 3)$ roles associated with E_1's account in system S_i, and then there are three accounts and $(n_1+n_2+n_3)$ roles created for E_1. Assume there are N_i $(1 \leq i \leq 3)$ employees in company C_i, each employee has one account in a system and each account is associated with an average of n roles. Then each system has to create $(N_1+N_2+N_3)$ accounts and there will be $n(N_1+N_2+N_3)$ roles in each system and $3n(N_1+N_2+N_3)$ accounts all together. When the number of cooperation companies increase, the number of accounts and roles will increase much more. In addition, the synchronization of employee's data in all cooperated companies' system may cause problems. For example, when an employee newly joins in a company, transfers internally or leaves the company, not only the employee's company has to update his

information in the system, all the other cooperated companies also have to update his information. Otherwise, a newly joined or internally transferred employee may not be able to access the shared data correctly and a former employee still have access to the shared data.

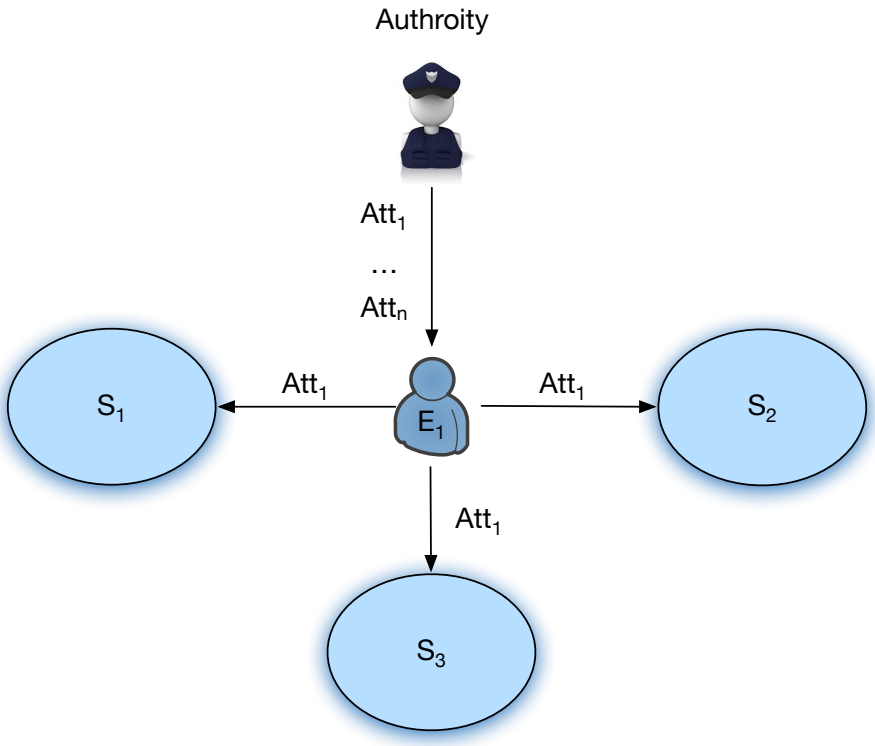
In the usage scenario described above, it actually should be enough for E_1 to show an evidence that it is an employee of company C_1, which is also the main idea behind ABA. The way how to resolve this multi-organization cooperation issue via ABA based system can be described in Fig. 2.2b . Assume there is an authority which is fully trusted by companies C_1, C_2 and C_3, and being an employee of company C_1 is considered as an attribute denoted by Att_1. First of all, evidence that E_1 is an employee of company C_1 should be shown to the authority, which is normally realized by company C_1 handing in all employee's information to the authority. Secondly, the authority authorizes E_1 with all the attributes E_1 possesses, which is denoted by $\{Att_1, \cdots, Att_n\}$. When E_1 wants to access the shared data owned by companies C_2 and C_3, it shows the the authorized attribute Att_1 to C_2 and C_3. Since both C_2 and C_3 trust the authority, once they have checked that Att_1 is authorized by the trusted authority, they believe that E_1 is an employee of C_1. E_1 will be authenticated and this authentication result will be used to judge whether E_1 can be authorized with the required access privileges or not. Compared with the mechanism based on RBAC, this approach is not only more simple and effective, but also can overcome the synchronization issue.

2.1.2 Usage Scenario 2: Data Outsourcing

A scenario of data outsourcing is described in Fig. 2.3. There is a top authority, which can only authorize new domain authorities. A domain authority can authorize both new domain authorities and data users. The top authority has the highest level of trust and the lowest level authorities have the lowest level of trust. In general, the lower the authorities are, the less trustful they are. In the system described in Fig. 2.3, data owners outsource their data in the cloud. At the same time, they want to have some control of the way how their data are accessed. Therefore, data owners define access requirements and the cloud server implements them. In addition, data users may want to access the data anonymously to the data owner. A possible solution can be based on ABAC. First of all, data users register themselves to authorities and gain authorized credentials of the attributes they possess. When data users request to access the outsourced data, they generate a signature based



(a) Usage scenario 1: RBAC-based mechanism



(b) Usage scenario 1: ABA-based mechanism

Figure 2.2: RBAC and ABA based mechanisms for usage scenario 1

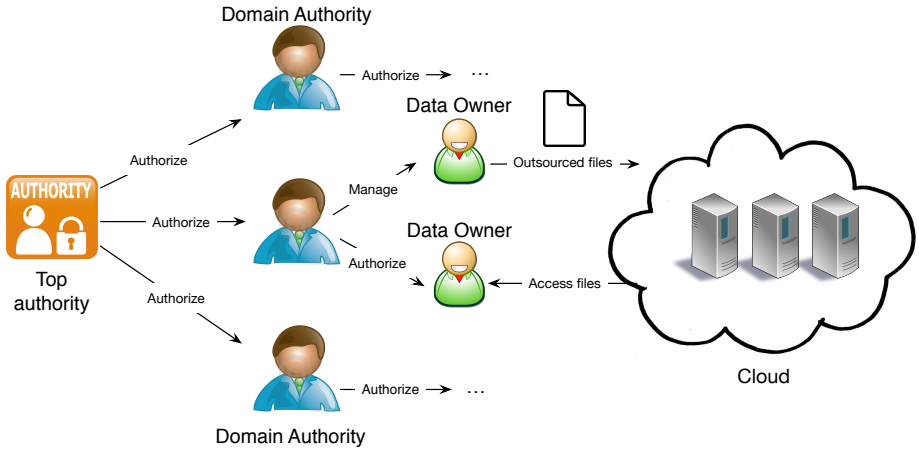


Figure 2.3: Usage scenario 2: data outsourcing

on their credentials according to associated AC rules. If the signature is valid, data users are authenticated and will be authorized with requested access rights. The process described is a typical user scenario of basic ABA schemes.

As shown in Fig. 2.3, authorities are organized in a hierarchical structure according to their trust levels, and therefore credentials authorized by different leveled authorities should also have different trust levels. In this way, the data owner can even require that the credentials should be generated by which specific authority or specific trust level. If this property is supported by ABA schemes, HABA schemes can be obtained and data owners can have more flexibility in defining authentication requirements and policies.

2.1.3 Usage Scenario 3: e-Health

Fig. 2.4 is a simplified illustration of an e-Health system with main system entities. Patients use medical devices to record their physical conditions and send their medical data to the cloud. Nurses and doctors can access patients' data in the cloud, but they can only access the data related to their work rather than all patients' records. Service provider manages cloud servers and the system but should not be allowed to access medical records stored in the cloud. Administration employees can only access data related to administration and management. Normal users are only allowed to perform some basic operations, for example, reviewing hospital regulations and opening hours. In addition, we have the following assumptions:

1. Patients are allowed to be registered in different hospitals.

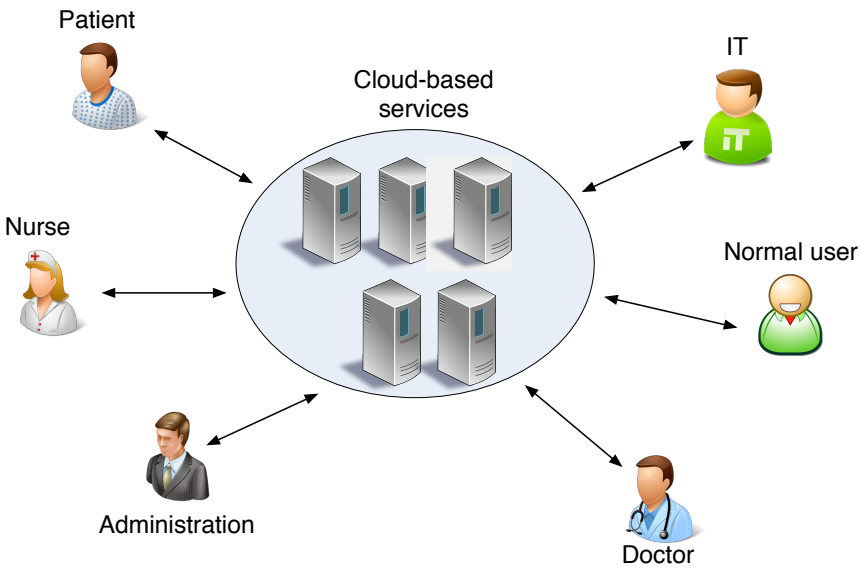


Figure 2.4: Usage scenario 3: e-Health

2. Different hospitals share part of their medical records in the cloud with each other, for instance, the medical records of patients who are treated by co-operation hospitals.

3. Doctors may be responsible for medical cases in different hospitals at the same time. For instance, a patient P_1 has newly transferred from hospital H_1 to hospital H_2. Doctor D_1 from hospital H_1 was in charge of P_1' case. Now doctor D_2 from hospital H_2 takes the charge instead of doctor D_1, but D_1 should still be allowed to access the necessary part of P_1's medical records to insist doctor D_2.

4. For complex medical cases, it should be very helpful to collect opinions and suggestions from other doctors. However, these doctors may want to give advice anonymously. Meanwhile, it should be able for the system to trace these doctors' identities when necessary.

From the above descriptions, we can see that the e-Health system can be considered as another case of multi-organization cooperation. However, according to assumption 4 listed above, there is an additional requirement compared with the first usage scenario, i.e., the system needs to achieve anonymity and traceability at the same time. Therefore, there should be an authority in the system that can open signatures and recognize signers' identities. Meanwhile, it should also be impossible

for normal users to gain any identifying information even given enough signatures. An e-health system described above can be solved by traceable ABA schemes.

2.2 Applications of Different ABA Schemes

In this section, we study categorizations of ABA schemes and which types of applications they are suitable for. We divide them into the following four categories:

1. Basic ABA schemes: a basic ABA scheme is as described in usage scenario 1 in Subsection 2.1.1. In a basic ABA scheme, users are supposed to be authorized with keys of attributes that they possess. When a user wants to be authenticated, he sends a request to the authenticator. The authenticator replies with attribute requirements. If the user owns the required attributes, he uses his attribute keys to generate a signature and sends it to the authenticator. If the signature is valid, the user is authenticated and otherwise rejected. The main goal of basic ABA schemes is to achieve anonymous authentication. They can provide least information leakage for users as well, which means that users only need to provide exactly the required "attributes" rather than a whole package of information. In addition, ABA schemes are also suitable for multi-organization cooperation and similar applications.

2. Attribute-related hierarchical attribute-based authentication (A-HABA) schemes: As stated in usage scenario 2 in Subsection 2.1.2, attribute authorities can be organized in a hierarchical structure based on their trust levels. Besides, they can also be organized according to their privilege levels based on specific needs of the system. In such systems, data owners or the storage systems have requirements on the trust or privilege level of attribute credentials.

3. User-related hierarchical attribute-based authentication (U-HABA) schemes: Except for attribute related hierarchy, ABA systems can also be built based on user related hierarchy. In such an ABA scheme, attribute keys are authorized by the same authority, while users are managed by different domain authorities of different trust levels. In this way, data owners can specify which groups can access their data in addition to attribute related requirements. This property enables more fine-grained and more flexible definition for authentication requirements. Different from using "group" as an attribute, this type of

domain management has several advantages, one of which is that the domain authority can gain more control over group users' identity information. Instead of using an opener that is responsible to trace all users' identities in the system, the group can have its own opener which can trace users' identities in the same group.

4. Traceable and untraceable ABA schemes: The ABA scheme described for e-Health in usage scenario 3 in Subsection 2.1.3 is traceable. Therefore, it achieves both anonymous authentication and traceability at the same time. In some ABA systems, signers' identities may need to be revealed under some special circumstances such that identities can serve as legal evidence, so traceable ABA schemes are preferred for these cases. If there is only a need for anonymous authentication, untraceable ABA schemes can be a good choice.

Among all the properties described above, only anonymity is mandatory for ABA schemes, meaning that given a signature, it is impossible for normal users to trace the signers' identity. As for hierarchy and traceability, they can be selectively implemented and combined, depending on requirements of specific applications and systems. Based on the above categorizations, we will express the cryptographic constructions of basic ABA schemes, user and attribute-related HABA schemes and untraceable ABA schemes in details in Chapters 4, 5 and 6 respectively.

Chapter 3

Categorization of ABA Schemes

In this chapter, we divide ABA schemes into different types according to two different criteria. The first criterion is properties of ABA schemes, such as anonymity, dynamic, traceability and hierarchy. In this Chapter, we will use these properties to divide ABA schemes into different categories, such as core, traceable, dynamic and HABA schemes. Meanwhile, we will describe the structure and workflow of each type of ABA schemes. The second criterion is the relation between attribute keys and attribute requirements, and we will use this criterion to divide ABA schemes into KP-ABA and CP-ABA schemes. This chapter will end with a summary of the security requirements of ABA schemes.

3.1 Categorization of ABA Schemes: Criterion 1

Most ABA schemes [20] use different authorities which are in charge of different tasks, but some ABA schemes [24] do not separate authorities by their responsibilities or functions. For better understanding, we assume that all ABA schemes in this dissertation consists of different authorities and each authority has only one main function. In this following, we divide ABA schemes into five types and describe their workflow, including core ABA schemes, traceable ABA schemes [13, 20], dynamic ABA schemes [13], user- and attribute-related HABA schemes. As far as we know, there has been no work in literature that deals with such detailed categorization for ABA schemes before.

3.1.1 Core ABA Schemes

As shown in Fig. 3.1, core ABA schemes consist of five entities, three of which are authorities, including the central authority, the attribute authority and the revocation authority. The central authority is considered as the top authority and it is responsible for system parameter generation and user registration. Given an attribute set, the attribute authority generates public and private attribute key pairs for all attributes in the given attribute set. The revocation authority manages revocation information related to users and attributes. The workflow of the core ABA schemes includes two phases: 1) system setup and 2) signature generation and verification. Its workflow proceeds as follows.

1. **System setup** This phase includes five steps from 1-a to 1-d in Fig. 3.1. The central authority generates system parameters in Step 1-a and sends them to the attribute authority in Step 1-b. Users register themselves and get their user keys from the central authority in Step 1-c. Next, the attribute authority generates public and private key pairs for all attributes in the system and deliver related private attribute keys to users in Step 1-d. In Step 1-e, the revocation authority retrieves revoked information from the central authority and the attribute authority, where the revoked information can related to revoked users or revoked attributes depending on different revocation mechanisms.

2. **Signature generation and verification** In this phase, the verifier chooses the required attributes and sends the requirement to the signer. If the signer owns the required attributes, he uses related attribute keys to generate a signature and sends it to the verifier in Step 2-a. After receiving the signature, the verifier first retrieves related revocation information from the revocation authority in Step 2-b and then gets necessary public parameters such as public attribute keys from the attribute authority in Step 1-e. Next the verifier checks whether the signature received in Step 1-a is valid as well as whether the signer or the attribute keys used are revoked. If neither the user nor the attribute keys used are revoked and the signature is valid at the same time, the signer will be authenticated successfully. Otherwise, the authentication fails. The verifier sends the verification result to the signer in Step 2-c.

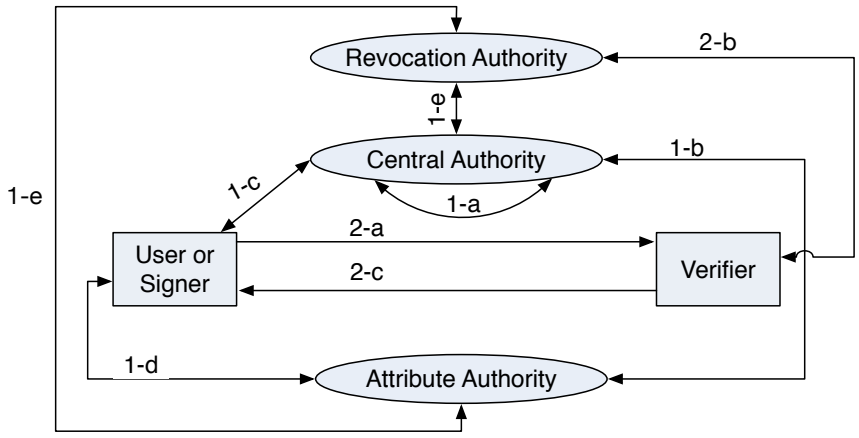


Figure 3.1: Workflow of core ABA schemes without traceability

3.1.2 Traceable ABA Schemes

Fig. 3.2 describes the general workflow of traceable ABA schemes. Compared with core ABA schemes, there is one more authority, called "opener", in traceable ABA schemes. In Step 1-f, the central authority communicates with the opener, sending the system tracing key and users' registration keys to the opener. The system tracing key is usually generated by the central authority and will be used by the opener to trace signers' identity later. Users' registration key may be users' keys or part of users' keys and they will be considered as users' identifying information.

In Step 2-d, the verifier sends a request to the opener requiring that the signer's identity needs to be revealed. This situation normally happens when there is a dispute and the signer's identity can be used as legal evidence. Identity revealing can only be performed when it is authorized by legal authorities. Intuitively, there are two requirements for identity tracing. First of all, given a valid signature, the opener should be able to "open" the signature and find the identity information. Secondly, the found identity information should belong to the real signer rather than a forged user.

3.1.3 Dynamic ABA Schemes

When users register themselves in the core ABA schemes in Step 1-c (Fig. 3.1), their user keys are generated by the central authority. As described in Subsection 3.1.1, the central authority is assumed as the top authority and should have the highest trust level. However, if the central authority is compromised, all user keys

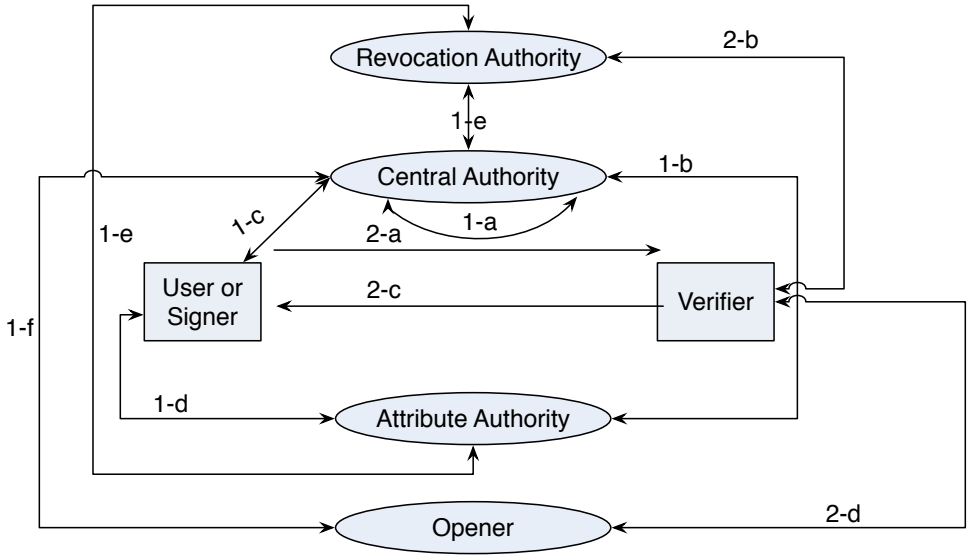
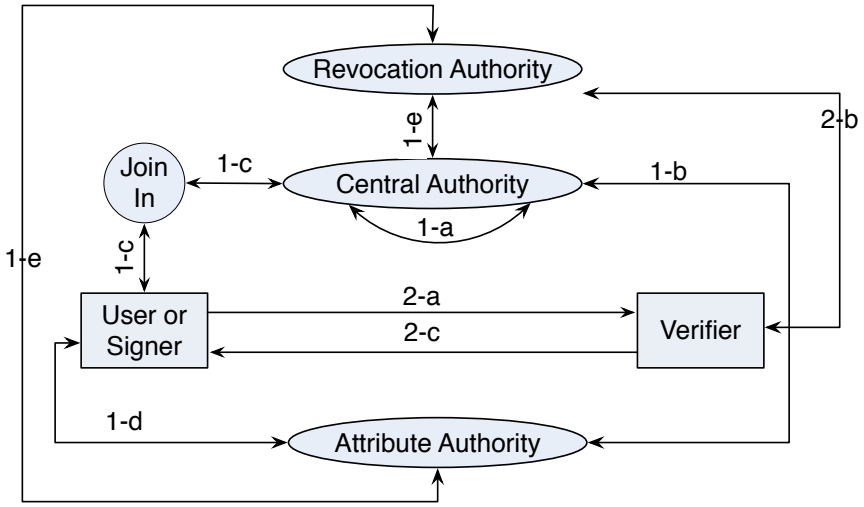


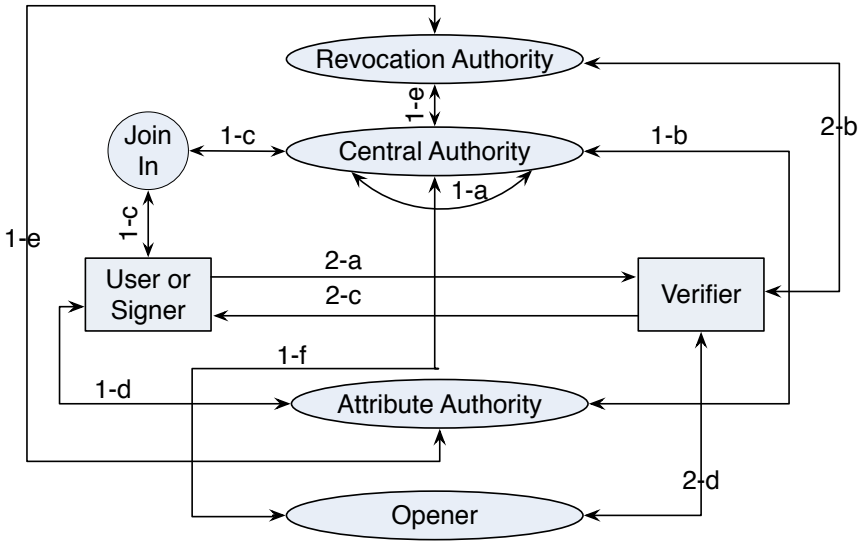
Figure 3.2: Workflow of traceable ABA schemes

will be revealed to adversaries and this can be solved by "dynamic" ABA schemes described in Fig. 3.3. In dynamic ABA schemes, a "join in" protocol [13] is applied to strengthen the security and privacy of users' keys. During this "join in" protocol, users interact with the central authority and both contribute in users' key generation by providing their own secrets. Two types of dynamic ABA schemes are shown in Fig. 3.3, without (Fig. 3.3a) and with (Fig. 3.3b) traceability respectively. Similar to the core ABA scheme shown in Fig. 3.1, there is no opener in Fig. 3.3a, an authority that can get the signer's identity information. However, the dynamic ABA scheme described in Fig. 3.3b is traceable, which means signers' identity information can be revealed with the help of an opener. Meanwhile, compared with Fig. 3.3a, there are two more steps in Fig. 3.3b. In step "1-f", the opener receives a tracing key from the central authority. In step "2-d", the opener sends back the signer's identity information to the verifier after it receives a valid request from a user requesting to open a signature.

There are two main advantages for both types of dynamic ABA schemes. Firstly, users can join the system at any point of time. Secondly, since users' secret key generation partly depends on users' secret which is unknown to the central authority, compromising the central authority will not cause the leakage of users' keys. Therefore it can strengthen the security and stability of an ABA scheme. Compared with the core ABA schemes described in Subsection 3.1.1, the ABA schemes in Fig. 3.3 is more "dynamic" because users can join in the system anytime, which is the



(a) Dynamic ABA schemes without traceability



(b) Dynamic ABA schemes with traceability

Figure 3.3: Workflow of dynamic ABA Schemes

criteria in [13] to divide ABA schemes into "static" and "dynamic". In this thesis, however, we will use the term "dynamic" in a different context to indicate whether attribute trees [25] are changeable or not, and this approach will be used to construct CP-ABA schemes. Therefore, when we use the term "dynamic ABA schemes" in this dissertation, it means those with the structure illustrated in Fig. 3.3 unless we clearly mention "dynamic attribute trees".

3.1.4 U-HABA Schemes

In both core and traceable ABA schemes, all users are authorized by the central authority. For some applications, however, it could be beneficial to utilize user-related hierarchical structure to some application domains. First of all, authorities may have different trust and privilege levels. Secondly, users should only be allowed to access data in a specific domain rather than the whole system. To satisfy these two requirements, a hierarchical structure of authorities and users illustrated in Fig. 3.4 can be used.

As shown in Fig. 3.4, a U-HABA scheme has only one top authority and the remaining authorities are domain authorities. Assume there are N levels of domain authorities and the following assumptions apply to the scheme.

1. The level 0 authority has the highest privilege and authorities of level N have the lowest privilege.

2. Authorities of level i ($1 \leq i \leq N$) can only be authorized by authorities of level $i-1$.

3. The top authority can only authorize domain authorities rather than users, while all domain authorities are able to authorize both users and domain authorities of a lower trust level.

4. Users can be authorized by different domain authorities and they can only access data in these authorized domains instead of the whole system.

Based on the above assumptions, we can see that the main difference between the U-HABA schemes and the core ABA schemes is that user keys are generated by their domain authorities rather than the central authority in U-HABA schemes. Therefore, when a verifier wants to authenticate a user from a specific domain, he or she needs to specify the domain in addition to the required attributes. Except for

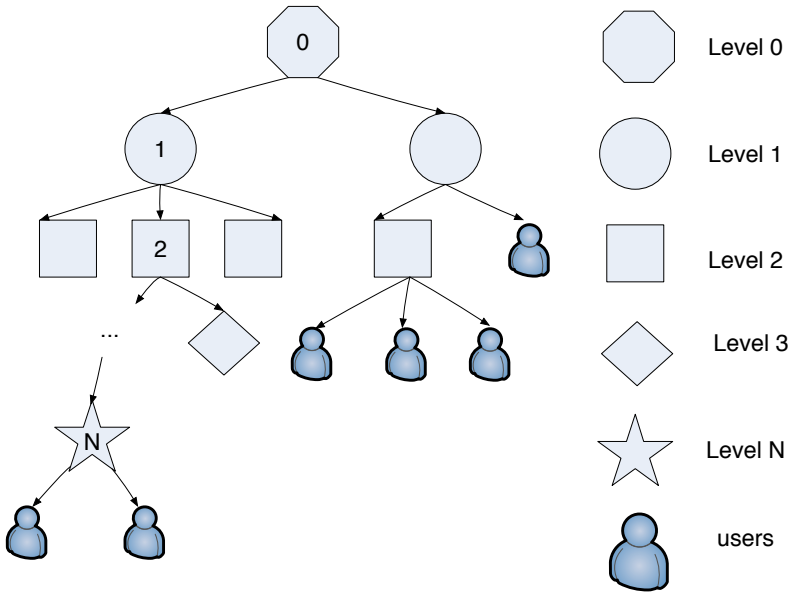


Figure 3.4: Structure of authorities and users in U-HABA schemes

the difference described above, the workflow of U-HABA schemes is the same as of the core ABA schemes.

3.1.5 A-HABA Schemes

Instead of demanding a hierarchical structure of authorities and users as in U-HABA schemes, some applications may require that attribute keys should be combined with different trust levels. For instance, data are classified by trust levels in some systems and users can access data as follows. For data of low trust levels, as long as users have the required attributes, they are allowed to access the data. However, for highly secret data, users should prove that their attribute keys are authorized by attribute authorities of a high trust level.

To meet the requirements of applications such as in the situation described above, an approach illustrated in Fig. 3.5 can be used. Assume that there are $N+1$ levels of attribute authorities. Difference in this case from the hierarchical authorities in Fig. 3.4 is that the attribute authorities are responsible for attribute keys generations instead of users' keys and users keys are generated by a central authority. For this approach (Fig. 3.5), the following assumptions have to be satisfied.

1. The level 0 attribute authority has the highest trust level and the level N attribute authorities have the lowest trust level.

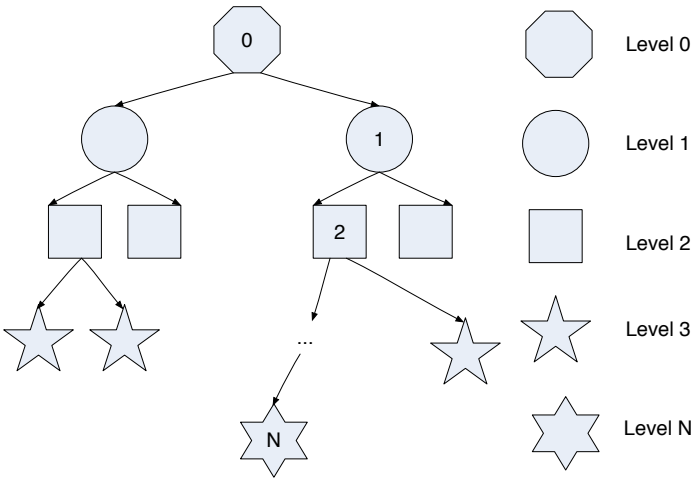


Figure 3.5: Structure of A-HABA schemes

2. Attribute authorities of trust level i ($1 \leq i \leq N$) can only be authorized by authorities of trust level $i-1$.

3. The top authority can only authorize domain attribute authorities rather than attribute keys, while all domain authorities are able to authorize both attribute keys and domain authorities of a lower trust levels.

4. Domain attribute authority of trust level i ($1 \leq i \leq N$) can only authorize attribute keys of the same or lower trust levels.

5. The trust levels of attribute keys are partly determined by the trust levels of their authorizing domain attribute authorities. The lower trust levels of the attribute authorities are, the lower trust levels the authorized attribute keys may be.

6. Users that hold attribute keys of trust level i ($1 \leq i \leq N$) can only access data that requires attributes keys of the same or a lower trust level.

In such ABA schemes, users register themselves in the central authority as in the core ABA schemes and users' keys are also generated by the central authority, while users' attribute keys are generated by domain attribute authorities. Except for these differences, the workflow and system structure are the same as in the core ABA schemes.

3.2 Categorization of ABA schemes: Criterion 2

In Section 3.1, we divided ABA schemes into five types according to the properties of traceability, dynamic and hierarchy. In this section, we will use another criterion to categorize ABA schemes, which is the relation between attribute keys and attribute requirements. To our best knowledge, there are no publications that clearly categorizes ABA schemes by this criterion.

This categorization originates from attribute-based encryption (ABE) schemes [26–32]. ABE schemes aim at encrypting plaintext based on a set of attributes and only users who possess the required attributes are able to decrypt the ciphertext. According to the way how plaintext is encrypted, ABE schemes can be divided into two types, i.e., key-policy attribute-based encryption (KP-ABE) [33] and ciphertext-policy attribute-based encryption (CP-ABE) [34]. For KP-ABE schemes [35–38], attribute keys are generated based on a specific attribute predicate. More specifically, a set of attributes with fixed relations is chosen first, where relations among these attributes usually represent policies which are represented by attribute trees [25] or access trees [39]. Then attribute keys are generated based on this set of attributes together with their relations. Therefore, ciphertext generated by this set of attribute keys can only be decrypted by users who own attributes with the same relations. For CP-ABE schemes [40–43], the generation of attribute keys are comparatively independent of specific policies. Plaintext is encrypted by a set of attributes, the relation between which does not have to be fixed. Therefore, as long as users possess the attributes used during encryption, they can decrypt the ciphertext. Compared with KP-ABE schemes, CP-ABE schemes are more flexible.

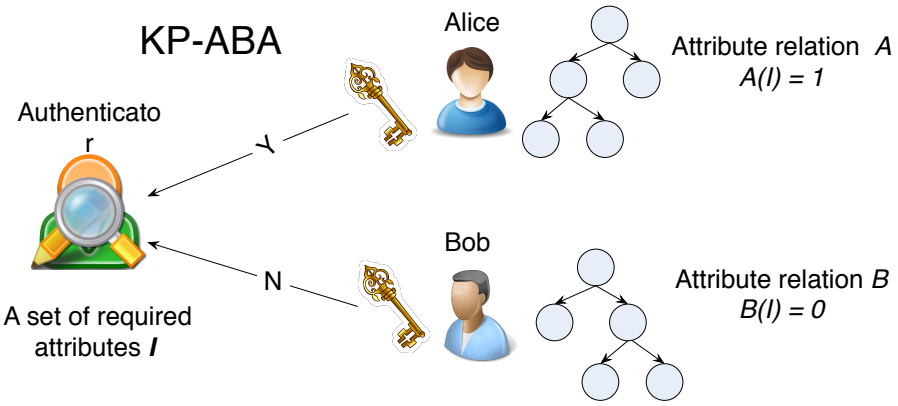
ABA schemes have some overlaps with ABE schemes but differ mainly in two aspects. First of all, the goals to be achieved are different. ABE schemes aim at the secrecy of data and the anonymity of attributes, more precisely, the encryption and decryption of data. Secondly, its security requirement is that an adversary cannot forge a valid ciphertext [44]. Differently, ABA schemes focus on user authentication, to confirm that the one to be authenticated possess a set of required attributes. The security requirement "anonymity" of ABA schemes emphasizes more on hiding the identity information of the one to be authenticated from the authenticator.

Despite of the two main differences described above, ABE and ABA schemes share a lot in common in their cryptographic construction, for example, how to generate attribute keys based on a set of attributes. Even though there is no ciphertext, encryption or decryption in ABA schemes, we still borrow the abbreviations

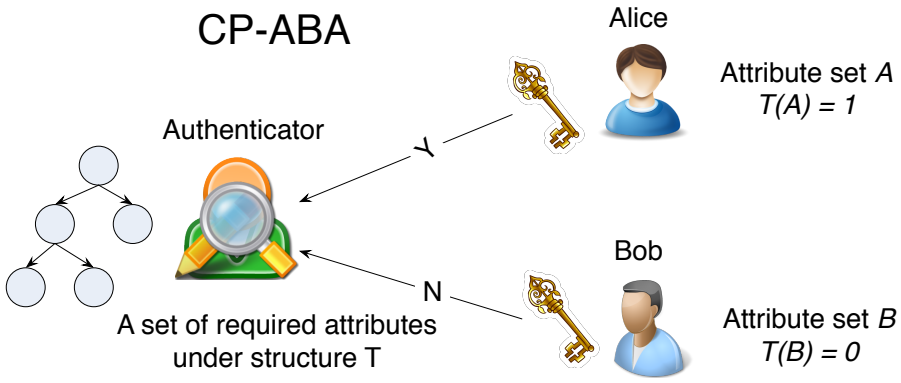
"KP" and "CP" from ABE schemes to divide ABA schemes into two categories in this dissertation, named as KP-ABA and CP-ABA respectively. Attribute keys of KP-ABA schemes are generated based on a set of attributes with fixed relations. Only users owning the required attributes with this fixed relations can be authenticated. For example, all ABA schemes proposed in [13] are KP-ABA based. On the contrary, the generation of attribute keys in CP-ABA schemes are independent from their relations. Given an attribute set, related attribute keys in CP-ABA schemes can be generated without pre-defining relations of the attributes, which will be defined by an authenticator later during authentication. These relations among attributes are usually represented by attribute trees [39]. As long as users own the attributes required in the attribute tree, they can be authenticated. All ABA schemes proposed in this dissertation are CP-ABA based, so we will not mention it when we introduce ABA schemes in the following chapters.

We will use Fig. 3.6 as an example to explain the main differences between KP-ABA and CP-ABA schemes. Suppose there are two users named as Alice and Bob, and both of them want to be authenticated by the authenticator. As illustrated in Fig. 3.6a, the authenticator requires that users own a set of attributes (denoted by I) satisfying relations represented by access structure A. Both Alice and Bob own the attribute set I. The difference is that Alice's attribute keys are generated based on access structure A but Bob's attribute keys are generated based on access structure B. We use $A(I) = 1$ and $B(I) = 0$ to denote whether the access structure based on attribute set I satisfying the requirement of the authenticator or not. Since Alice possesses attribute keys generated according to access structure A, she will be successfully authenticated. On the contrary, even though Bob owns the required attribute set I, his attribute keys are generated according to access structure B, so unfortunately Bob will be authenticated eventually.

From Fig. 3.6b, we can see that different from KP-ABA schemes, the access structure T is bound to the required attribute set rather than users' attribute keys in CP-ABA schemes. In this case, the authenticator requires that users should hold a set of attributes that satisfy the access structure T. Alice owns attribute set A and Bob owns attribute set B, and we use $T(A) = 1$ and $T(B) = 0$ to express that attribute set A satisfies the access structure T while B does not. Therefore, Alice will pass the authenticator's authentication but the poor Bob will fail the authentication again.



(a) KP-ABA schemes



(b) CP-ABA schemes

Figure 3.6: Two categories of ABA schemes

3.3 Security Requirements of ABA Schemes

Authors of [13, 20] have summarized the security requirements of ABA schemes. Since the security requirements in [13] are more detailed and systematic compared with those in [20], we will utilize them and briefly introduce the summarized properties in the following, including no previous knowledge, unforgeability, unlinkability, traceability, anonymous identities, anonymous attributes, coalition resistance and separability.

- No previous knowledge: We assume that the signer and the verifier do not know each other before, so the authentication cannot depend on any previous knowledge.

- Unforgeability: It is hard for an adversary to forge a valid signature or the proof of possessing attributes.

- Unlinkability: Given two signatures, it should be hard to tell whether they are generated by the same signer or not.

- Traceability: Given a valid signature, there should be an authority to "open" the signature and reveal the identity of the signer.

- Anonymous identities: It should be impossible for a normal user to obtain any identifying information of the signer.

- Anonymous attributes: It requires least attribute disclosure for an ABA scheme.

- Coalition resistance: When a single signer does not possess all the attributes required by the verifier, it should be impossible for a couple of users to cooperate and generate a valid signature.

- Separability: Tasks should be separated to different authorities, and each authority should be able to perform its own task independently except for gaining some necessary inputs from other authorities.

If we categorize the above properties more precisely, no previous knowledge assumptions, anonymous attributes and separability can be considered as scheme properties while the rest are security requirements. The property of "no previous knowledge assumptions" is an assumption rather than a security requirement that

should be proved formally during scheme design. The property of anonymous attributes is a general requirement for ABA schemes and it is very difficult to express it formally and precisely. As for "separability", when we design the system structures of ABA schemes, we divide the tasks and assign them to different authorities as described in Fig. 3.1. Therefore, it is straightforward to judge whether an ABA scheme satisfies separability or not. As a result, when we design new ABA schemes, we will only formally define and prove the five security requirements, which are anonymity, traceability, unforgeability, unlinkability and coalition resistance.

Chapter 4

Construction of ABA Schemes

In this chapter we explain how to cryptographically construct ABA schemes. We start with a survey on recent work on the cryptographic construction of ABA schemes, including attribute tree building and algorithms for ABA scheme setup. In Section 4.1, we describe how to build attribute trees [45], which are logical statements with tree structures and are used to express attribute requirements. There are two approaches to build attribute trees. The most commonly used approach is to build attribute trees from top to down [13, 46], but there are still some research results on generating attribute trees from down to top [20, 25].

Based on the research results from [20] and [22], we will study two cases about how to construct CP-ABA schemes by applying two different approaches to build attribute trees in Sections 4.4 and 4.5 respectively. In the first case, a big central attribute tree is built first based on all attributes in the system. When a verifier selects part of the system attributes as the attribute requirement, he can obtain the attribute subtree he needs by simplifying the big central attribute tree. In this way, system parameter regeneration can be avoided for new attribute tree generation such as in KP-ABA schemes [13]. In the second case, a top-to-down attribute tree generation approach is utilized, where the generation of new attribute trees will not cause the regeneration of system parameters. Meanwhile, the computation and storage cost for a big central attribute tree can also be reduced. For each case study, we will start with the description of the adversary models and formal definitions of security requirements. Then we explain the algorithms of ABA system setup as well as the process of signature generation, verification and possibly opening. Next we analyze the correctness of the proposed scheme and provide formal proofs of corresponding security requirements.

4.1 Related Work

There has already been some work on ABA scheme construction. As far as we know, [13] is the most systematic research result about ABA schemes. In Khader's PhD thesis [13], he built several static ABA schemes based on group signatures and developed a general approach how to construct static ABA schemes. However, when it came to dynamic ABA schemes, he only demonstrated how to construct them by a few examples without providing a general approach. All the ABA schemes presented in [13] are traceable KP-ABA schemes. All signers' attribute keys are generated based on signers' identifying information and an attribute tree. The approach causes the drawback that the attribute requirement should be fixed. Once the attribute requirement is changed, all signers' attribute keys must be regenerated. The ABA scheme proposed in [46] is an attribute-based authentication key exchange protocol, of which the main goal is to gain a shared secret key at the end of the protocol. Therefore, its security requirement is to prevent adversaries from learning any information about the secret key, such that it differs from protocols that mainly focus on authentication. Authors in [47] have proposed an attribute-based signatures (ABS) scheme for online social networks but its main purpose is for authenticating. The ABA scheme proposed in [48] aims at saving system resources for multi-networks' users wireless roaming, and the scheme can resist the man-in-the-middle attacks, replay attacks and collusions attacks. However, neither of the ABA schemes in [47, 48] is traceable. Besides, there is no proof in [47, 48] showing that the signatures used to authenticate the signer cannot be forged by an adversary, meaning that it cannot be confirmed whether a signature is generated by the signer or an adversary. The ABA scheme proposed in [20] is traceable and it builds attribute trees from down to top instead of the most commonly used top-to-down approach [13], so that attribute trees are changeable. Its flexibility is at the cost of a big central attribute tree, from which different attribute subtrees can be extracted.

As for other attribute-based approaches, such as ABAC [8, 49–53], ABS [14, 15, 54] and ABE [16, 55], their cryptographic constructions share a lot in common with ABA schemes and the way how they build attribute trees are similar with differences in some aspects. ABAC includes both authentication and authorization, and thus it covers a wider and sometimes a different scope [56] compared with ABA schemes. ABE is a type of public key encryption and it was firstly proposed by Amit Sahai and Brent Waters [17]. In ABE schemes, both users' secret keys and ciphertext

are dependent on users' attributes. Only those who possess the decryption keys which are related to the attributes used during encryption can decrypt the ciphertext. The main goal of ABE is message encryption and decryption, with the security requirement that an adversary cannot forge valid ciphertext [44]. ABS schemes [14] are schemes under which a signer attests a message using a set of attributes he possesses from the authority. Given a predicate or requirement, only those who satisfy the predicate can generate a valid signature based on some set of attributes they possess from the authority. Meanwhile, users cannot collude to pool their attributes together to generate a valid signature. Given a signature, others cannot gain any knowledge about the identifying information of the signer. Even though signers' identifying information should be kept secret, the privacy of ABS schemes is achieved by using the attribute set to generate signatures rather than signers' identifying information as in ABA schemes. Moreover, traceability is required in neither ABE nor ABS schemes.

4.2 Attribute Trees

Attribute trees originate from access trees [39], where they are used to represent logical access control requirements and usually built from top to down. Another approach to construct attribute trees is from down to top [20, 25]. In this approach, a big central attribute tree is built first and attribute subtrees can be obtained by simplifying this big attribute tree, so it allows dynamic attribute tree construction. However, the generation and storage costs for such a big attribute tree can be a waste of resources unless there is a need to change attribute trees frequently. In Subsections 4.2.1 and 4.2.2, we will use an example to describe in details how to construct attribute trees in both top-to-down and down-to-top approaches mentioned earlier.

4.2.1 Top-to-down Attribute Trees

An attribute tree is a tree structure describing the logic relations among required attributes. In this subsection, we will use Fig. 4.1 as an example to explain attribute trees and how to build them. Assume the related attribute set of attribute tree in Fig. 4.1 is $\{att_1, \cdots, att_5\}$. We index all leaf nodes and interior nodes from 1 to 9 as shown in Fig. 4.1. Leaf nodes are indexed from 1 to 5 and interior nodes indexed from 6 to 9. In an attribute tree, leaves represent attributes and interior nodes are

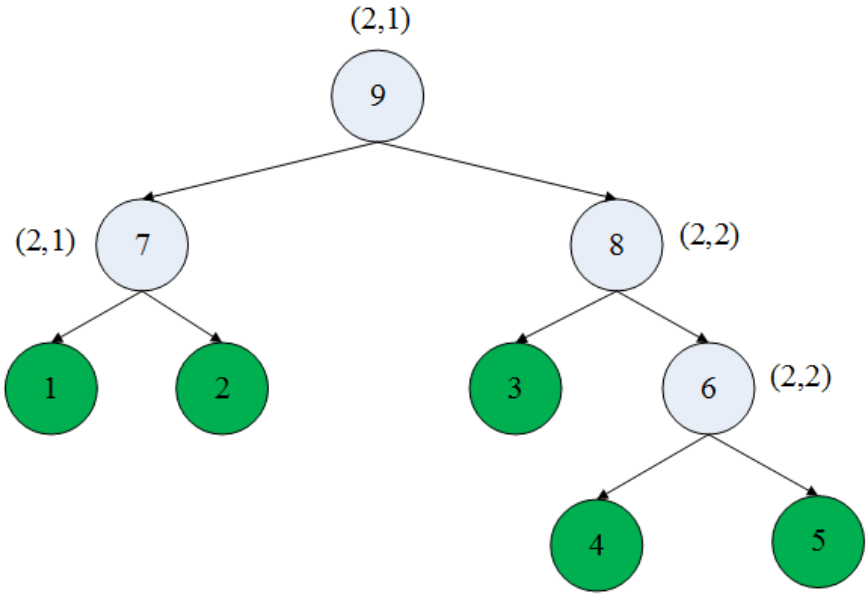


Figure 4.1: Attribute tree: an example

threshold gates. Leaf nodes indexed from 1 to 5 correspond to attributes att_1 to att_5 respectively. For an interior node x, let l_x and k_x be the numbers of x's children and the threshold respectively. If k_x is less than l_x, the interior node represents logical "OR". If k_x equals to l_x, the interior node represents logical "AND". For example, node 7 has two children and its threshold is one, so it represents logical "OR". The same for node 9. As for nodes 6 and 8, either the number of their children or the threshold is two, so both represent logical "AND".

Suppose Γ is an attribute tree with rt as its root and Ψ as its attribute set. We use $ind(Node) \in \mathbb{Z}_p^*$ to represent the index of node $Node$ and assume q_{Node} is a polynomial bound to node $Node$ with degree $k_{Node} - 1$. To construct Γ, we start from the root by selecting polynomials over $\mathbb{Z}_p^*$, of which degrees equal to $k_{Node} - 1$ for all nodes. Let $\alpha \in \mathbb{Z}_p^*$ be randomly selected and we assign $q_{rt}(0) = \alpha$, and then we randomly select a polynomial $q_{rt}(x)$ satisfying $q_{rt}(0) = \alpha$ for root rt. The value of the remaining nodes are computed by $q_{Node}(0) = q_{par(Node)}(ind(Node))$, where $par(Node)$ is the parent of $Node$. Then a polynomial $q_{Node}(x)$ is chosen such that the equation $q_{Node}(0) = q_{par}(ind(Node))$ holds. For a leaf node, there is no related polynomial but only a constant value.

In the following, we use a specific example to demonstrate the generation of an attribute tree. As explained above, the relations of all leaf (green) nodes in Fig. 4.1 can be represented by equation $(att_1 \vee att_2) \vee (att_3 \wedge (att_4 \wedge att_5))$. Assume $\alpha = 15$.

We start from the root, i.e., node 9 and the polynomials are set as following:

$$\begin{aligned} q_9(0) &= 15 & q_9(x) &= 15 \\ q_7(0) = q_9(7) &= 15 & q_7(x) &= 15 \\ q_8(0) = q_9(8) &= 15 & q_8(x) &= x+15 \end{aligned}$$

$$\begin{aligned} q_1(0) = q_7(1) &= 15 \\ q_2(0) = q_7(2) &= 15 \\ q_3(0) = q_8(3) &= 18 \\ q_6(0) = q_8(6) &= 21 & q_6(x) &= 2x+21 \\ q_4(0) = q_6(4) &= 29 \\ q_5(0) = q_6(5) &= 31. \end{aligned}$$

As mentioned before, there is no polynomial rather than a constant value assigned to a leaf node. Leaves 1, 2, 3, 4, 5 are assigned with values 15, 15, 18, 29 and 31 respectively.

4.2.2 Down-to-top Attribute Trees

There are two phases in down-to-top attribute tree building approach. First of all, a big central attribute tree is built based on all attributes in the system. Secondly, given an attribute subset, the big central attribute tree is simplified and what remains after the simplification is the required attribute subtree. In the rest of this dissertation, we use *Create_CTree* and *Simplify_CTree* to denote the algorithms of "creating a central attribute tree" and "simplifying a central attribute tree" respectively.

Before describing the down-to-top construction of attribute trees, we first explain Lagrange interpolating polynomials [57–59] because they will be used to compute polynomials that are bound to interior nodes.

Definition 1. ***Lagrange Interpolating Polynomial*** *[58] The Lagrange interpolating polynomial is the polynomial $P(x)$ of degree $\leq (n-1)$ that passes through the n points $(x_1, y_1 = f(x_1))$, $(x_2, y_2 = f(x_2))$, $\cdots$, $(x_n, y_n = f(x_n))$, and is given by $P(x) = \Sigma_{j=1}^{n} P_j(x)$, where*

$$P_j(x) = y_j \Pi_{k=1, k\neq j}^{n} \frac{x - x_k}{x_j - x_k}.$$

Written explicitly,

$$P(x) = \frac{(x-x_2)(x-x_3)\cdots(x-x_n)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_n)}y_1 + \frac{(x-x_1)(x-x_3)\cdots(x-x_n)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_n)}y_2 + \cdots + \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_n-x_1)(x_n-x_2)\cdots(x_n-x_{n-1})}y_n.$$

Suppose the system attribute set is Ψ and the corresponding attribute tree is Γ with *rt* as its root. Next, we explain the algorithms that will be used to construct attribute trees from down to top. For an interior node $Node$ of Γ, let l_{Node} and k_{Node} be the numbers of its children and threshold respectively. The algorithm $Create\_CTree$ proceeds as follows:

1. Add dummy nodes: For an interior node $Node$, if $k_{Node} < l_{Node}$, add $l_{Node} - k_{Node}$ dummy nodes as $Node$'s children such that the interior node $Node$ represents logic "AND" now. Repeat this dummy node adding process for all "XOR" interior nodes until they all become "AND" nodes.

2. Index nodes: Index all leaf nodes (attribute leaf nodes and dummy leaf nodes) and interior nodes.

3. Assign leaf nodes: Assign a secret value to each leaf node.

4. Compute polynomials: Each interior node $Node$ should be assigned with a polynomial denoted by $q_{Node}(x)$, the degree of which should be $d_{Node} = k_{Node} - 1$. We start the polynomial computing process from the parents of leaf nodes. Since each leaf child of node $Node$ is assigned with a secret value, then the polynomial $q_{Node}(x)$ can be calculated by Lagrange interpolating polynomial [58]. Then assign $q_{Node}(0)$ to $Node$. The values assigned to dummy nodes are public.

5. Repeat 4 until every interior node is assigned with a polynomial.

Next we explain how to obtain an attribute subtree by simplifying the central attribute tree Γ^{Ext}. Assume $\Psi_i \subset \Psi$ is an attribute subset and its related attribute subtree is Γ_i. The algorithm $Simplify\_CTree$ proceeds as follows:

1. Let $\overline{\Psi}_i = \Psi - \Psi_i$. Delete all leaf nodes that corresponds to attributes in $\overline{\Psi}_i$.

2. Delete an interior node $Node$ together with its descendants if the number of $Node$'s children is less than its threshold. The remaining tree is the required attribute tree Γ_i.

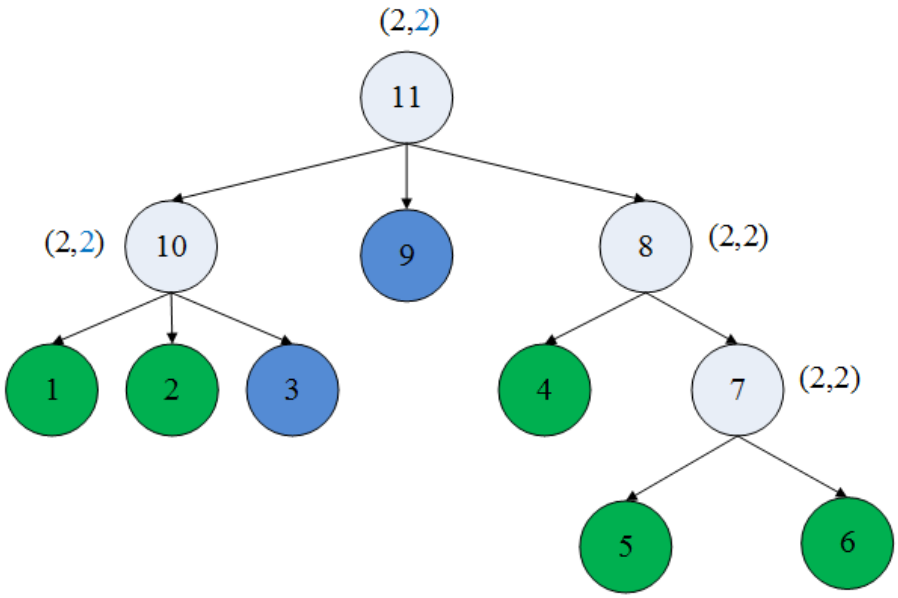


Figure 4.2: The central attribute tree

Assume the system attribute set is $\Psi = \{att_1, \cdots, att_5\}$ and the related attribute tree Γ is as shown in Fig. 4.1. Now we use the example from Subsection 4.2.1 to describe how to implement the algorithms *Create_CTree* and *Simplify_CTree*. Since the thresholds of nodes 7 and 9 is 1, which is less than 2, i.e., the number of their children, we need to add a dummy node for each of these nodes. The extended attribute tree is shown in Fig. 4.2. If we compare Fig. 4.2 with Fig. 4.1, we can see that there are three differences between these two attribute trees:

1. We add a dummy (blue) node for both nodes 10 and 11 in Fig. 4.2, and these two dummy nodes are indexed as nodes 3 and 9 respectively.

2. Since we have added dummy nodes, we index all nodes in Fig. 4.1 with different numbers and the results are illustrated as in Fig. 4.2.

3. The thresholds of nodes 10 and 12 are both changed from 1 to 2. They are high lightened in blue.

In the following, we demonstrate how to assign values to all nodes and polynomials to all interior nodes of the central attribute tree Γ^{Ext}.

1. Assign secret values to leaf nodes: Each attribute (green) node should be assigned with a random secret value. Let s_i be secret values, $i \in \{1,2,4,5,6\}$, where $s_1 = 15$, $s_2 = 17$, $s_4 = 23$, $s_5 = 52$ and $s_6 = 56$.

2. Compute polynomials: For node 10, the related polynomial $q_{10}(x)$ can be considered as a line passing two points $(1,15)$ and $(2,17)$. Therefore, according to Lagrange's theorem, $q_{10}(x)$ can be computed as $q_{10}(x) = 2x+13$. Similarly, we have $q_7(x) = 4x+32$, $q_8(x) = 3x+11$ and $q_{11}(x) = x+3$.

3. Assign values to dummy nodes: Assume the values related to dummy nodes 3 and 9 are denoted by d_3 and d_9. Since $q_{11}(x) = x+3$, we can compute d_9 and d_9 by $d_9 = q_{11}(9) = 12$ and $d_3 = q_{10}(3) = 19$.

Based on this central attribute tree, we can obtain attribute subtrees according to different attribute subsets. Assume we have attribute subset $\Psi_i = \{att_1, att_3\}$ and the related attribute subtree is denoted by Γ_i. The algorithm $Simplify\_CTree$ proceeds as follows. First, we delete all leaf nodes that relates to attributes in attribute subset $\overline{\Psi}_i = \{att_2, att_4, att_5\}$, which are related to nodes indexed with 2, 5 and 6 respectively. The result of this step is shown as Fig. 4.3a. Since node 7 has no child now, we delete it and the result is shown in Fig. 4.3b. From Fig. 4.3b, we can see that the threshold is 2 but it has only one child now. Therefore, we delete node 8 and the remaining part is the required attribute subtree Γ_i, which is illustrated in Fig. 4.3c.

4.3 Definitions

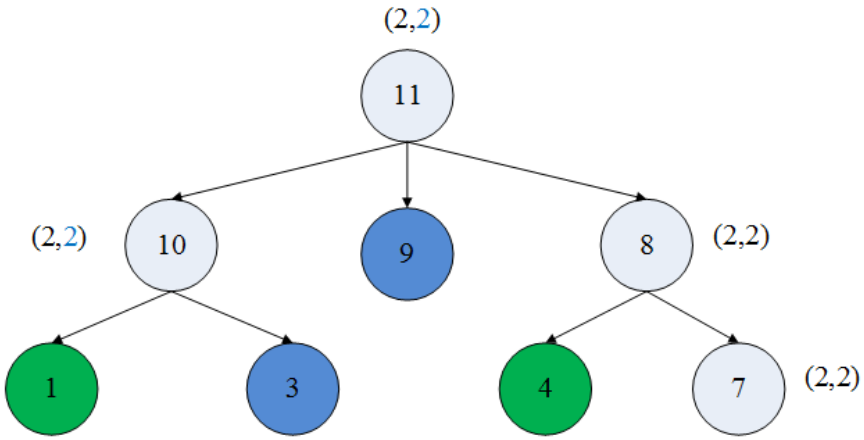
The security of all cryptographic schemes is based on certain hard mathematical problems or mathematical assumptions. The same holds for ABA schemes. Before formally describing the first scheme, we first explain some notions and preliminaries as follows.

Definition 2. ***(Bilinear Maps)*** *[60–62] Let G_1, G_2 and G_3 be cyclic groups of prime order p, with $g_1 \in G_1$ and $g_2 \in G_2$ as the generators. e is an efficient bilinear map if the following two properties hold.*

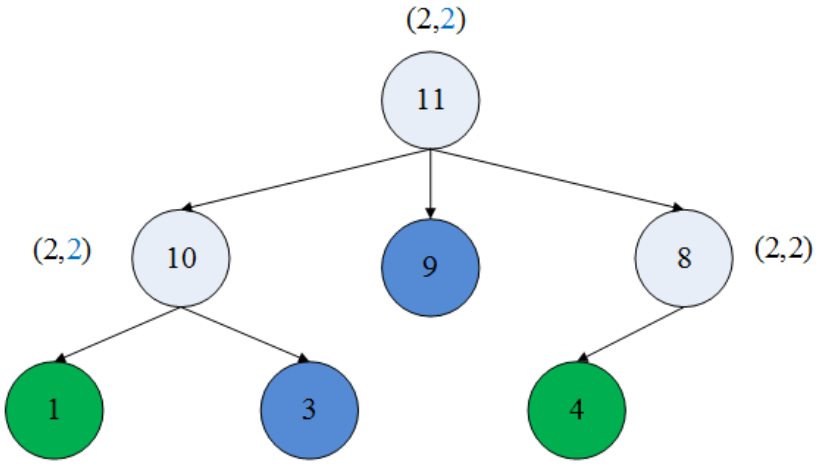
1. *Bi-linearity: equation $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds for any $a, b \in \mathbb{Z}_p^*$.*

2. *Non-degenerate: $e(g_1, g_2) \neq 1_{G_3}$, where 1_{G_3} is the unit of G_3.*

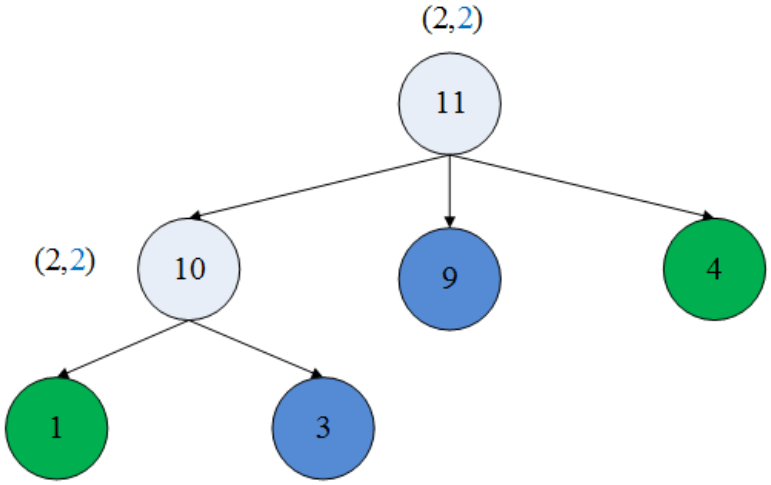
Definition 3. ***(Negligible Functions)*** *[63, 64] Assume $\mathbb{Z}_{\geq 0}$ and $\mathbb{R}_{\geq 0}$ are the sets of integers and real numbers equal or larger than 0 respectively. A function $f : \mathbb{Z}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ is negligible if $\forall c > 0$, there exists a $\lambda_0 > 0$ such that for $\forall \lambda > \lambda_0$, equation $f(\lambda) < 1/\lambda^c$ holds.*



(a) Simplifying a central attribute tree: a



(b) Simplifying central attribute tree: b



(c) Simplifying central attribute tree: c

Figure 4.3: Simplifying a central attribute tree

Definition 4. ***(Discrete Logarithm Problem (DLP))*** *[65, 66] Let G be a cyclic group of prime order p with $g \in G$ as its generator. Given $h = g^k$, computing $k = log_g h$ is considered as a DLP.*

Definition 5. ***(q-Strong Diffie-Hellman (q-SDH) Problem)*** *[67, 68] Let G be a (multiplicative) cyclic group of prime order p with $g \in G$ as its generator. Given input $(g, g^x, g^{x^2}, \cdots, g^{x^q}) \in G$, outputting a pair $(x_i, g^{1/(x+x_i)})$ where $x_i \in \mathbb{Z}_p^*$ is considered as a q-SDH problem.*

Definition 6. *([67])* ***(Decision Linear Diffie-Hellman based Encryption (DLE) in*** *G) Assume G is a multiplicative cyclic group of order p. In a DLE scheme, a user's public key is $u, v, h \in G$ and its private key is $\varepsilon_1, \varepsilon_2 \in \mathbb{Z}_q^*$, satisfying $u^{\varepsilon_1} = v^{\varepsilon_2} = h$. To encrypt a message M, the user randomly chooses $\alpha, \beta \in \mathbb{Z}_q^*$ and computes the encryption message as a triple $< C_1, C_2, C_3 >$, where $C_1 = u^\alpha$, $C_2 = v^\beta$ and $C_3 = Mh^{\alpha+\beta}$. The decrypted message is calculated by $C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$.*

Definition 7. ***(Decision Linear Diffie-Hellman Problem (DeLP))*** *[20] Let G be a cyclic group of prime order p, with $u, v, h \in G$ as its generators. Given $u^a, v^b, h^c \in G$ $(a, b, c \in \mathbb{Z}_p^*)$ as the input, deciding whether $a + b = c$ or not is considered as a decision DeLP.*

4.4 ABA Scheme with Dynamic Attribute Trees

In this section, we provide an example that demonstrates how to construct an ABA scheme with dynamic attribute trees by using the down-to-top attribute tree building approach. This scheme is called Scheme 1. Scheme 1 is a traceable CP-ABA scheme. Our main focus is to use a specific example to provide more details on how to cryptographically construct an ABA scheme as illustrated in Fig. 3.2.

4.4.1 Security Requirements

In Section 3.3, we discussed eight properties of ABA schemes, including no previous knowledge assumptions, unforgeability, unlinkability, traceability, anonymous identities, anonymous attributes, coalition resistance and separability. However, in this thesis, we will only focus on the five security requirements listed as below.

Anonymity To achieve basic anonymity, identities of signers should be protected. Furthermore, even signers' attributes should be protected, and signers only

have to prove that they possess the required attributes. This property is the main security requirement of an ABA scheme and is mandatory.

Unforgeability A signer's signature should not be able to be forged by an outsider that does not belong to the system. In some systems, a signature is even required to be unforgeable for authorities in the system. However, in a system where authorities generate all keys and secrets, the authorities obviously can forge all signatures. Therefore, "unforgeable" is defined differently in different ABA schemes. However, all ABA systems should provide at least the basic level of "unforgeability", i.e, for the outsiders.

Unlinkability Given two signatures, if it should be impossible to decide whether they are generated by the same signer, the ABA scheme is unlinkable. If a system does not satisfy it, given enough signatures, there may exist a possibility to reveal the signer's identity.

Coalition resistance A signer can only generate the required signature if he or she has all the required attributes. It should be impossible for different users to pool their attributes together to generate a valid signature together. If a system satisfies this security requirement, it is coalition resistant.

Traceability Given a valid signature, if the opener can successfully trace the signer's identity and the signer is a valid user in the ABA system, the ABA scheme is traceable. It is a useful security requirement for some applications, such as obtaining evidence for legitimate issues and so on.

Based on the above descriptions, we will give formal definitions when analyzing the security requirements of each scheme. Descriptions of these five security requirements apply for all ABA schemes proposed in this dissertation unless the scheme does not satisfy the security requirements. Therefore, we will not repeatedly express them later.

4.4.2 Building Blocks: Algorithms

To describe the cryptographic construction of Scheme 1, we need the following algorithms serving as building blocks. More details about the meanings of notation are given in Table. 4.1.

- **Scheme1.spg($\mathbf{k_0}$)**$\rightarrow \{\mathbf{MPK}, \mathbf{MSK}\}$: This algorithm provides system parameter generation. It takes system security parameter k_0 as its input and outputs the system public and private key sets MPK and MSK respectively.

• **Scheme1.ukg**($\mathbf{MPK}, \mathbf{MSK}, \mathbf{U_i}$)$\rightarrow$ $\mathbf{bsk_i}$: This algorithm provides user key generation. It takes the system public key set MPK, private key set MSK and the identity of U_i as input and outputs U_i's private key base $bsk_i =< A_i, x_i >$, where x_i is a secret based on which U_i's secret key A_i is generated.

• **Scheme1.akg**($\mathbf{MPK}, \Psi$)$\rightarrow$ $\{\mathbf{APK}_\Psi, \mathbf{ASK}_\Psi\}$: This algorithm provides attribute key generation. It takes the system public key set MPK and the attribute set Ψ as input to generate the public attribute key set APK_Ψ and the private attribute key set ASK_Ψ.

• **Scheme1.uakg**($\mathbf{MPK}$, $\mathbf{A_i}$, Ψ_i)$\rightarrow$ $\mathbf{USK_i}$: This algorithm provides user attribute key generation. It takes the system public key set MPK, U_i's private key A_i and attribute subset Ψ_i as input to generate U_i's private attribute key set USK_i.

• **Scheme1.atg**($\mathbf{MPK}$, Ψ, $\mathbf{ASK}_\Psi$, $\mathbf{APK}_\Psi$,$\mathbf{ASK}_\Psi$)$\rightarrow$ $\{\Gamma, \mathbf{PTree}_\Psi\}$: This algorithm provides attribute tree generation. It takes the system public key set MPK, attribute set Ψ, system public and private attribute key sets APK_Ψ and ASK_Ψ as input and generates the attribute tree Γ and related parameters $PTree_\Psi$.

• **Scheme1.ars**()$\rightarrow$ Ψ': This algorithm provides attribute requirement selection. The verifier selects the required attribute set Ψ' based on which the signer will generate a signature later.

• **Scheme1.sg**($\mathbf{MPK}, \mathbf{MSK}$, Γ, Ψ', $\mathbf{USK_i}$, $\mathbf{M}$)$\rightarrow$ δ: This algorithm provides signature generation. It takes the system public and private key sets MPK and MSK, attribute tree Γ, the selected attribute set Ψ', U_i's private key set USK_i and the message M as input to generate the signature δ.

• **Scheme1.sv**($\mathbf{M}, \delta$, Ψ')$\rightarrow$ $\mathbf{b}$: This algorithm provides signature verification. It takes message M with its signature δ and the selected attribute set Ψ' as input to check whether the signature is valid or not. The checking result is denoted by $b \in \{0,1\}$, where 0 and 1 mean YES and NO respectively.

• **Scheme1.so**($\mathbf{M}, \delta$, Ψ', $\mathbf{tk}$)$\rightarrow$ $\mathbf{U_i}$: This algorithm provides signature opening. It takes the message M with its signature δ, the selected attribute set Ψ' and the opener's tracing key tk as the input to trace the identity information of the signer U_i, which is U_i's index i here.

4.4.3 Scheme 1 Construction

The design of Scheme 1 is presented below with related notation explained in Table 4.1.

System setup includes two phases: 1) system setup and 2) signature generation, verification and opening. System setup is the preparation phase and it includes system parameter generation, user key generation, attribute key generation, user attribute key generation and attribute tree generation.

1. **System parameter generation (Scheme1.spg)** This algorithm is performed by the central authority. Assume k_0 is the system security parameter. G_1, G_2 and G_3 are three multiplicative groups of prime order p and $e : G_1 \times G_2 \to G_3$ is a bilinear map. φ is a computable isomorphism between G_1 and G_2. $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a hash function. Randomly select a generator $g_2 \in G_2$ and $\gamma, \xi_1, \xi_2, \in \mathbb{Z}_p^*$. Let $g_1 = \varphi(g_2)$, $u = h^{\xi_1}$ and $v = h^{\xi_2}$. Then the system private parameter set is $MSK =< \gamma, tk >$, where $tk =< \xi_1, \xi_2 >$ is the tracking key. The system public parameter set is $MPK =< G_1, G_2, G_3, e, H, g_1, g_2, h, u, v, w >$, where $w = g_2^{\gamma}$.

2. **User key generation (Scheme1.ukg)** This algorithm is performed by the central authority. Take γ as input and generate a private key base $bsk_i =< A_i, x_i >$ for each user U_i, where $x_i \in \mathbb{Z}_p^*$ is randomly selected by the central authority and $A_i = g_1^{1/(\gamma + x_i)}$ is the output of a q-SDH pair $(g_1, g_2, g_2^{r_0}, g_2^{r_0^2}, \cdots, g_2^{r_0^q})$. A_i should be registered in the opener's database for the purpose of tracing if necessary.

3. **Attribute key generation (Scheme1.akg)** This algorithm is run by the attribute authority. Assume $\Psi = \{att_1, \cdots, att_{N_a}\}$ is the system attribute set and $N_A = |\Psi|$ is the number of the attributes. To generate a system attribute key, the attribute authority randomly selects a number $t_j \in \mathbb{Z}_p^*$ for each attribute att_j first, where $1 \leq j \leq N_a$. Then the private and public attribute key pair for att_j is denoted by $ask_j = t_j$ and $apk_j = g_2^{t_j}$ respectively. Let $h_j = h^{t_j}$ $(1 \leq j \leq N_a)$. The public key set related to Ψ is denoted by $APK_\Psi = \{h_1, \cdots, h_{N_a}, apk_1, \cdots, apk_{N_a}\}$ and the private key set related to Ψ is denoted by $ASK_\Psi =< t_1, \cdots, t_{N_a} >$.

4. **User attribute key generation (Scheme1.uakg)** This algorithm is run by the attribute authority. Assume that user U_i is valid and not included in the

Table 4.1: Notation for Scheme 1

Notation	Description
k_0	The system security parameter.
$\mathbb{Z}_p^*$	The set of integers modulo p, where p is a large prime number.
G_i	(Multiplicative) cyclic group of prime order p, where $1 \leq i \leq 3$.
e	$e : G_1 \times G_2 \rightarrow G_3$ is a bilinear map.
φ	A computable isomorphism between G_1 and G_2.
H	$H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function [69].
U_i	User i in the system.
Ψ	System attribute set.
Ψ_i	$\Psi_i \subset \Psi$, attribute subset owned by U_i.
Ψ'	$\Psi' \subset \Psi_i$, attribute subset owned by U_i and used to generate the signature.
N_a	$N_a = \lvert\Psi\rvert$, the number of attributes in set Ψ.
N_i	$N_i = \lvert\Psi_i\rvert$, the number of attributes in set Ψ_i.
N'_a	$N'_a = \lvert\Psi'\rvert$, the number of attributes in set Ψ'.
MPK	The system public parameter set.
MSK	The system private parameter set.
bsk_i	The private key base of U_i and $bsk_i = \{A_i, x_i\}$.
apk_j	Public attribute key related to attribute att_j, where $1 \leq j \leq N_a$.
ask_j	Private attribute key related to attribute att_j, where $1 \leq j \leq N_a$.
$T_{i,j}$	U_i's private attribute key related to attribute att_j.
APK_Ψ	Public attribute key set related to attribute set Ψ.
ASK_Ψ	Private attribute key set related to attribute set Ψ.
USK_i	Private key set of U_i.
Γ	Attribute tree generated based on attribute set Ψ.
Γ^{Ext}	The central attribute tree extended from Γ.
Γ'	Attribute tree generated based on attribute set Ψ'.
N_d	The number of dummy nodes in attribute tree Γ^{Ext}.
rt	Root of attribute tree Γ.
rt'	Root of attribute tree Γ'.
$\mathbb{L}^{Att}$	Leaf node set related to attribute set Ψ.
$\mathbb{L}'^{Att}$	Leaf node set related to attribute set Ψ'.
$\mathbb{L}^{Dum}$	Dummy node set of attribute tree Γ.
$\mathbb{L}'^{Dum}$	Dummy node set of attribute tree Γ'.
$\mathbb{L}$	Leaf node set of attribute tree Γ, where $\mathbb{L} = \mathbb{L}^{Att} + \mathbb{L}^{Dum}$.
$\mathbb{L}'$	Leaf node set of attribute tree Γ', where $\mathbb{L}' = \mathbb{L}'^{Att} + \mathbb{L}'^{Dum}$.
V	The verifier.
M	The message based on which U_i generates the signature.
δ	Signature generated by user U_i based on attribute subset Ψ' and message M.

revocation list. Let $\Psi_i \subset \Psi$ be the attribute set owned by U_i and $N_i = |\Psi|$ be the number of attributes in Ψ_i. When U_i wants to obtain the attribute key related to attribute att_j ($1 \leq j \leq N_i$), the attribute authority interacts with U_i to calculate $T_{i,j} = A_i^{t_j}$ and sends it to U_i. Then U_i's private key set is $USK_i =< A_i, x_i, T_{i,1}, \cdots, T_{i,N_i} >$.

5. **Attribute tree generation (Scheme1.atg)** This algorithm is carried out by the attribute authority. Assume Ψ is the system attribute set. Then a central attribute tree Γ based on Ψ can be created by applying the algorithm $Create\_CTree$ described in Subsection 4.2.2. Let rt be the root of attribute tree Γ, $\mathbb{L}^{Att}$ ($|\mathbb{L}^{Att}| = |\Psi| = N_a$) be the attribute leaf node set and $\mathbb{L}^{Dum}$ ($N_d = |\mathbb{L}^{Dum}|$) be the dummy leaf node set. Compared with the algorithm $Create\_CTree$, the only difference is the leaf node value assigning process. Instead of using random secret numbers, the attribute authority assigns t_j to the related leaf node. The numbers assigned to dummy nodes are computed the same as described in Subsection 4.2.2 and denoted by d_j. Then the attribute authority calculates $g_2^{t_j}$ based on all t_j ($1 \leq j \leq N_a$), and $g_2^{d_k}$ based on all d_k ($1 \leq k \leq N_d$). The public parameter set related to the central attribute tree Γ is denoted by $PTree_\Gamma =< g_2^{d_1}, \cdots, g_2^{d_{N_D}} >$.

Signature generation and verification After the system setup phase, signers and verifiers can generate and verify signatures now. Suppose M is the message to be signed. The algorithm of signature generation and verification proceed as follows:

1. **Verifier: attribute requirement selection (Scheme1.ars)** Let V be the verifier. V selects the attribute requirements and sends the selection result to U_i.

2. **Signer: signature generation (Scheme1.sg)** After receiving the attribute requirements from V, U_i first checks whether he or she possesses the required attributes or not. If so, U_i decides the attribute set to use and continue. Otherwise, the algorithm aborts. Assume the attribute set to be used is $\Psi' \subset \Psi_i$ and $N'_a = |\Psi'|$. Based on Ψ', U_i runs algorithm $Simplify\_CTree$ as described in Subsection 4.2.2 and obtains the attribute subtree Γ'. Assume the attribute and dummy leaf sets of attribute subset Γ' are $\mathbb{L}^{'Att}$ and $\mathbb{L}^{'Dum}$ respectively and $\mathbb{L}' = \mathbb{L}^{'Att} + \mathbb{L}^{'Dum}$. Then U_i computes $\triangle_j$, $\triangle_{d_j}$ and its root value rt' as described in Subsection 4.2.2. Next U_i calculates $g_d = \prod_{d_k \in \mathbb{L}^{'Dum}} (g_2^{d_k})^{\triangle_{d_k}}$. Then

U_i randomly selects $\zeta, \alpha, \beta, \varepsilon, r_\zeta, r_\alpha, r_\beta, r_\varepsilon, r_x, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p^*$ and performs the following calculations

$$
\begin{aligned}
&C_1 = u^\zeta, C_2 = v^\beta, C_3 = A_i h^{\zeta+\beta}, C_4 = A_i w^\varepsilon, CT_j = T_{i,j} h_j^\alpha,\\
&\delta_1 = x_i\zeta, \delta_2 = x_i\beta, R_1 = u^{r_\zeta}, R_2 = v^{r_\beta}, R_4 = C_1^{r_x} u^{-r_{\delta_1}},\\
&R_5 = C_2^{r_x} v^{-r_{\delta_2}}, R_3 = e(C_3, g_2)^{r_x} e(h,w)^{-r_\zeta - r_\beta} e(h, g_2)^{-r_{\delta_1} - r_{\delta_2}},\\
&R_{Att} = \frac{e(\prod_{att_j \in \Psi'} h_j^{\triangle_j}, g_2)^{r_\alpha}}{e(w, rt'/g_d)^{r_\varepsilon}},\\
&c = H(M, C_1, C_2, C_3, C_4, R_1, R_2, R_3, R_4, R_5, R_{Att}) \in \mathbb{Z}_p^*,\\
&s_\zeta = r_\zeta + c\zeta, s_\beta = r_\beta + c\beta, s_\alpha = r_\alpha + c\alpha, s_\varepsilon = r_\varepsilon + c\varepsilon, s_x = r_x + cx_i,\\
&s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_2} = r_{\delta_2} + c\delta_2.
\end{aligned}
$$

Then the signature is $\delta = < M, C_1, C_2, C_3, C_4, c, CT_1, \cdots, CT_{|N_a'|}, s_\zeta, s_\beta, s_\alpha, s_\varepsilon, s_x, s_{\delta_1}, s_{\delta_2}, \Psi' >$.

3. **Verifier: signature verification (Scheme1.sv)** After receiving the signature δ from U_i, the first step for V is to check whether U_i and Ψ' are revoked or not. If not, V continues and otherwise it rejects the signature. Based on Ψ', V runs algorithm *Simplify_CTree* described in Subsection 4.2.2 to get the simplified tree Γ'. Then V computes the root value rt', $\triangle_j$, $\triangle_{d_k}$ and $g_d = \prod_{d_k \in \mathbb{L}'^{Dum}} (g_2^{d_k})^{\triangle_{d_k}}$. Next it calculates

$$
\begin{aligned}
&R_1' = u^{s_\zeta} C_1^{-c}, R_2' = v^{s_\beta} C_2^{-c}, R_4' = u^{-s_{\delta_1}} C_1^{s_x}, R_5' = v^{-s_{\delta_2}} C_2^{s_x},\\
&R_3' = e(C_3, g_2)^{s_x} e(h,w)^{-s_\zeta - s_\beta} e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \left(\frac{e(C_3, w)}{e(g_1, g_2)}\right)^c,\\
&R_{Att}' = \frac{e(\prod_{att_j \in \Psi'} h_j^{\triangle_j}, g_2)^{s_\alpha}}{e(w, rt'/g_d)^{s_\varepsilon}} \left(\frac{e(C_4, rt'/g_d)}{e(\prod_{att_j \in \Psi'} CT_j^{\triangle_j}, g_2)}\right)^c.
\end{aligned}
$$

Finally, V checks whether $c = H(M, C_1, C_2, C_3, C_4, R_1', R_2', R_3', R_4', R_5', R_{Att}')$ holds. If so, V accepts the signature and U_i is authenticated. Otherwise V rejects the signature.

Signature opening (Scheme1.so) Before opening the signature, the opener needs to check whether the signature is valid or not. If it is valid, it computes $A_i = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$, where A_i is considered as U_i's identity information.

4.4.4 Correctness Analysis

In this subsection, we analyze the correctness of Scheme 1.

Theorem 1. ***(Correctness)*** *The construction of Scheme 1 proposed in Subsection 4.4.3 is correct, which means:*

1. *Tuple* $< R_1, R_2, R_3, R_4, R_5, R_{Att} >$ *equals to* $< R'_1, R'_2, R'_3, R'_4, R'_5, R'_{Att} >$.
2. $A_i = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$ *holds.*

Proof. 1) As described in the construction algorithm in Subsection 4.4.3, the verifier computes the tuple $< R'_1, R'_2, R'_3, R'_4, R'_5, R'_{Att} >$ as follows.

$$
\begin{aligned}
R'_1 &= u^{s_\zeta} C_1^{-c} = u^{r_\zeta + c\zeta} (u^\zeta)^{-c} = u^{r_\zeta} = R_1 \\
R'_2 &= v^{s_\beta} C_2^{-c} = v^{r_\beta + c\beta} (v^\beta)^{-c} = v^{r_\beta} = R_2 \\
R'_3 &= e(C_3, g_2)^{s_x} e(h, w)^{-s_\alpha - s_\beta} e(h, g_2)^{-s_{\delta_1} - s_{\delta_2}} \left(\frac{e(C_3, w)}{e(g_1, g_2)}\right)^c \\
&= e(C_3, g_2)^{r_x + cx_i} e(h, w)^{-(r_\alpha + c\alpha) - (r_\beta + c\beta)} e(h, g_2)^{-(r_{\delta_1} + c\delta_1) - (r_{\delta_2} + c\delta_2)} \left(\frac{e(C_3, w)}{e(g_1, g_2)}\right)^c \\
&= R_3 \left(e(C_3, g_2)^{x_i} e(h, w)^{-(\alpha+\beta)} e(h, g_2)^{-(\delta_1 + \delta_2)} \frac{e(C_3, w)}{e(g_1, g_2)} \right)^c \\
&= R_3 \left(e(C_3 h^{-(\alpha+\beta)}, w g_2^{x_i}) e(C_3, w)^{-1} \frac{e(C_3, w)}{e(g_1, g_2)} \right)^c \\
&= R_3 \left(e(A_i, w g_2^{x_i}) (g_1, g_2)^{-1} \right)^c = R_3 \\
R'_4 &= u^{-s_{\delta_1}} C_1^{s_x} = u^{-r_{\delta_1} - c\delta_1} (u^\zeta)^{r_x + cx_i} = u^{-r_{\delta_1} - c\zeta x_i} (u^\zeta)^{r_x + cx_i} = u^{-r_{\delta_1}} u^{\zeta r_x} \\
&= C_1^{r_x} u^{-r_{\delta_1}} = R_4 \\
R'_5 &= v^{-s_{\delta_2}} C_2^{s_x} = v^{-r_{\delta_2} - c\delta_2} (v^\beta)^{r_x + cx_i} = v^{-r_{\delta_2} - c\beta x_i} (v^\beta)^{r_x + cx_i} = v^{-r_{\delta_2}} v^{\beta r_x} \\
&= C_2^{r_x} v^{-r_{\delta_2}} = R_5
\end{aligned}
$$

In the following, we prove that

$$
R'_{Att} = R_{Att} \left(\frac{e(A_i, rt'/g_d)}{e(\prod_{att_j \in \Psi'} A_i^{t_j \triangle_j}, g_2)} \right)^c .
$$

For a leaf node x, let $ind(x)$ be its index and $In_{sib(x)}$ be the index set of all x's siblings including x. We define $L_x = \prod_{l \in In_{sib(x)}, l \neq ind(x)} \frac{-l}{ind(x) - l}$, $Path_x = \{x, x_1, \cdots, x_n\}$ and $\triangle_x = \prod_{z \in \{Path_x - rt\}} L_z$, where $x_1 = par(x)$, $x_{i+1} = par(x_i)$ $(1 \leq i \leq n-1)$ and $x_n =$

rt. Then for attribute tree Γ, equation $\sum_{j\in L^{Att_i}} \triangle_j t_j + \sum_{j\in L^{Dum_i}} \triangle_{d_j} d_j = q_{rt}(0)$ holds. Therefore, we know that equations $\sum_{j\in \mathbb{L}'^{Att}} \triangle_j t_j + \sum_{j\in \mathbb{L}'^{Dum}} \triangle_{d_j} d_j = rt'$ and $g_d = \prod_{d_j\in \mathbb{L}'^{Dum}} (g_2^{d_j})^{\triangle_{d_j}}$ hold, so we have

$$
\begin{aligned}
& e(\prod_{j\in \mathbb{L}'^{Att}} A_i^{t_j\triangle_j}, g_2) e(\prod_{j\in \mathbb{L}'^{Dum}} A_i^{d_j\triangle_j}, g_2) = e(A_i^{rt'}, g_2) \\
\Longleftrightarrow\ & e(\prod_{j\in \mathbb{L}'^{Att}} A_i^{t_j\triangle_j}, g_2) e(A_i, g_d) = e(A_i, rt') \\
\Longleftrightarrow\ & e(\prod_{j\in \mathbb{L}'^{Att}} A_i^{t_j\triangle_j}, g_2) = e(A_i, rt'/g_d).
\end{aligned}
$$

Based on the above equation, we have

$$
\begin{aligned}
R'_{Att} &= \frac{e(\prod_{att_j\in\Psi'} h_j^{\triangle_j}, g_2)^{s\alpha}}{e(w, rt'/g_d)^{s\varepsilon}} \left(\frac{e(C_4, rt'/g_d)}{e(\prod_{att_j\in\Psi'} CT_j^{\triangle_j}, g_2)} \right)^c \\
&= \frac{e(\prod_{att_j\in\Psi'} h_j^{\triangle_j}, g_2)^{r\alpha}}{e(w, rt'/g_d)^{r\varepsilon}} \frac{e(\prod_{att_j\in\Psi'} h_j^{\triangle_j}, g_2)^{c\alpha}}{e(w, rt'/g_d)^{c\varepsilon}} \left(\frac{e(C_4, rt'/g_d)}{e(\prod_{att_j\in\Psi'} CT_j^{\triangle_j}, g_2)} \right)^c \\
&= R_{Att} \left(\frac{e(\prod_{att_j\in\Psi'} h_j^{\triangle_j}, g_2)^{\alpha}}{e(w, rt'/g_d)^{\varepsilon}} \frac{e(C_4, rt'/g_d)}{e(\prod_{att_j\in\Psi'} CT_j^{\triangle_j}, g_2)} \right)^c \\
&= R_{Att} \left(\frac{e(\prod_{att_j\in\Psi'} h_j^{\alpha\triangle_j}, g_2)}{e(w^{\varepsilon}, rt'/g_d)} \frac{e(w^{\varepsilon}, rt'/g_d)}{e(\prod_{att_j\in\Psi'} h_j^{\alpha\triangle_j}, g_2)} \right)^c \left(\frac{e(A_i, rt'/g_d)}{e(\prod_{att_j\in\Psi'} T_{i,j}^{\triangle_j}, g_2)} \right)^c \\
&= R_{Att} \left(\frac{e(A_i, rt'/g_d)}{e(\prod_{att_j\in\Psi'} T_{i,j}^{\triangle_j}, g_2)} \right)^c = R_{Att} \left(\frac{e(A_i, rt'/g_d)}{e(\prod_{att_j\in\Psi'} A_i^{t_j\triangle_j}, g_2)} \right)^c .
\end{aligned}
$$

Since $e(\prod_{j\in\mathbb{L}^{Att_i}} A_i^{t_j\triangle_j}, g_2) = e(A_i, rt'/g_d)$, equation $R_{Att} = R'_{Att}$ holds. Therefore, the tuple $< R_1, R_2, R_3, R_4, R_5, R_{Att} >$ equals to $< R'_1, R'_2, R'_3, R'_4, R'_5, R'_{Att} >$.

2) $C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2}) = A_i h^{\zeta+\beta} / ((u^{\zeta})^{\varepsilon_1} (v^{\beta})^{\varepsilon_2}) = A_i$.

From the above proof, we can see that Theorem 1 holds and thus the algorithm presented in Subsection 4.4.3 is correct. □

4.4.5 Adversary Model

Before describing and proving the security requirements of Scheme 1, we need to introduce the adversary model. In this dissertation, we use random oracle [70–73] to express the security requirements of all proposed ABA schemes formally. We will use **Scheme1** and .**Oracle** as the prefix and the suffix respectively to represent the names of oracles.

• **Scheme1.Syspara.Oracle**: It represents system parameter oracle and allows an adversary to query all parameters generated by algorithm Scheme1.spg described in Subsection 4.4.2.

• **Scheme1.Userkey.Oracle**: It represents the user key oracle and allows an adversary to query the secret key base bsk_i of a user U_i by sending U_i's index i.

• **Scheme1.Attkey.Oracle**: It represents the attribute key oracle. An adversary can send an attribute $att_j \in \Psi$ $(1 \leq j \leq N_a)$, to query the system private attribute key ask_j related to att_j. There is no need to query about apk_j because it is public.

• **Scheme1.UserAttkey.Oracle**: It represents the user attribute key oracle. An adversary can send U_i's index i and an attribute $att_j \in \Psi$ $(1 \leq j \leq N_a)$, to query about U_i's attribute key $T_{i,j}$ related to attribute att_j. If U_i does not own the attribute att_j, the oracle will reply with NULL. Otherwise, the oracle replies with $T_{i,j}$.

• **Scheme1.Signature.Oracle**: It represents the signature oracle and allows an adversary to obtain a user's signature. The adversary sends message M, U_i's index i and the required attribute set Ψ' to the signature oracle and will be replied with a valid signature δ based on $\{M, i, \Psi'\}$.

• **Scheme1.Opening.Oracle**: It represents the signature opening oracle and allows an adversary to obtain the signer's index i by sending $\{M, \delta, \Psi'\}$ to the opening oracle.

In the following, we will use these oracles to describe the adversary's abilities and define the security requirements of anonymity and traceability. To define anonymity, we first introduce a game named **Scheme1.Game.Anonymity** as follows.

Scheme1.Game.Anonymity Assume A is an adversary with polynomially bounded computational abilities [74]. This game consists of the following steps:

1. **System setup**: The system setup should be completed in this step by running the algorithms Scheme1.spg, Scheme1.ukg, Scheme1.akg, Scheme1.uakg, Scheme1.atg and Scheme1.ars. The adversary can obtain all public parameters.

2. **Phase 1**: In this phase, the adversary A is allowed to query all oracles described above but is forbidden from querying Scheme1.Syspara.Oracle about the tracing key tk. Otherwise, A can use tk to open signatures in the challenge phase, which will make this game meaningless.

3. **Challenge**: The adversary A selects two indexes i_0 and i_1, a message M together with a selected attribute set Ψ', and then sends them to the system. The system replies with δ_b which is the signature generated based on $\{i_b, M, \Psi'\}$, $b \in \{0,1\}$. One thing to notice is that this challenge should not have been queried before.

4. **Phase 2**: The adversary A performs the same as described in phase 1 and A is not allowed to query about the challenge described above.

5. **Guess**: The adversary A outputs its guess $b' \in \{0,1\}$. If b equals to b', A wins the game. Otherwise, A fails.

Definition 8. ***(Anonymity of Scheme 1)*** *Assume A is an adversary with polynomially bounded computational abilities. If the probability that A wins the game* ***Scheme1.Game.Anonymity*** *is negligible, then Scheme 1 provides anonymity.*

Scheme1.Game.Traceability Assume A is an adversary with polynomially bounded computational abilities. This game consists of the following steps:

1. **System setup**: The system setup should be completed in this step by running the algorithms Scheme1.spg, Scheme1.ukg, Scheme1.akg, Scheme1.uakg, Scheme1.atg and Scheme1.ars. The adversary can obtain all public parameters, the tracing key tk, users' private key base list, the system private attribute key set ASK and users' private attribute keys. This adversary A is very powerful. A can trace signers' identity information as long as it is given valid signatures.

2. **Phase 1**: In this phase, the adversary A is able to query all oracles described above.

3. **Challenge**: The adversary decides to challenge now. It chooses the required attribute set Ψ', message M to be singed and U_i's index i. Here U_i is the user the adversary wants to impersonate.

4. **Phase 2** The adversary A repeats phase 1, but A cannot query about the challenge described above.

5. **Output**: The adversary A forges a valid signature δ and sends it to the opener. The opener opens the signature δ and gets the signer's index i'. If $U_{i'}$ is not registered in the system, the opener outputs 1 and A wins the game. Otherwise the opener outputs 0 and A fails the game.

Definition 9. ***(Traceability of Scheme 1)*** *Assume A is an adversary with polynomially bounded computational abilities. If the probability that A wins the game* ***Scheme1.Game.Traceability*** *is negligible, then Scheme 1 provides traceability.*

4.4.6 Security Analysis

In this subsection, we prove that Scheme 1 satisfies the five security requirements described in Subsection 4.4.1, i.e., anonymity, unforgeability, unlinkability, coalition resistance and traceability. Before the proof, we first introduce some assumptions on which we base our proof.

Assumption 1. *([67, 68])* ***(q-SDH Assumption)*** *Assume G_1 and G_2 are (multiplicative) cyclic groups with prime order p and $|G_1| = |G_2|$. For an algorithm A, if $|Pr[A(g_1, g_2, g_2^{\gamma}, g_2^{\gamma^2}, \cdots, g_2^{\gamma^q}) = (g_1^{1/\gamma+x}, x)] - 1/|G_1|| \leq \varepsilon$ holds, we say that A has a negligible advantage to solve q-SDH in (G_1, G_2) and then we can assume q-SDH is hard.*

Assumption 2. ***(DeLP Assumption)*** *[20] Assume G is a (multiplicative) cyclic group with prime order p. For an algorithm A, if $|Pr[A(u^{\alpha}, v^{\beta}, h^{\alpha+\beta}) = (u^{\alpha}, v^{\beta}, h^{c})] - 1/|G|| \leq \varepsilon$ holds, we say that A has a negligible advantage to solve DeLP in G and then we can assume DeLP is hard.*

Assumption 3. ***(DLE Assumption)*** *([67]) If DeLP holds, we say that DLE is semantically secure against chosen plaintext attack (CPA) or in-distinguishable chosen plaintext attack (IND-CPA) secure.*

Assumption 4. ***(Forking Lemma)*** *([13]) Given only public data as input, if an adversary A with polynomially bounded computation ability can find a valid signature* $(M, \delta_0, c, \delta_1)$ *with non-negligible probability, then there exists a replay with a different oracle, which can output a new valid signature* $(M, \delta_0, c', \delta_1')$ *with non-negligible probability where* $c \neq c'$.

Theorem 2. ***(Anonymity)*** *Scheme 1 is fully anonymous if DLE is IND-CPA secure under the same attribute set. More specifically, given* $A_{i(0)}$ *and* $A_{i(1)}$ *and the corresponding signature* δ_1 *and* δ_2*, where* $\delta_b =< M, C_1, C_2, C_{3(b)}, C_4, c, CT_1, \cdots, CT_{|N_A'|}, s_\zeta, s_\beta, s_\alpha, s_\varepsilon, s_x, s_{\delta_1}, s_{\delta_2}, \Psi' >$, $b \in \{0,1\}$. *Given a random toss* $b \in \{0,1\}$*, if the probability that an adversary A with polynomially bounded computation ability has non-negligible advantage to guess the correct b, we say that Scheme 1 is not fully anonymous. Otherwise, Scheme 1 is fully anonymous.*

Proof. Suppose that adversary A can break the anonymity of Scheme 1. Then it means A has a non-negligible advantage to guess the correct b in the above statement. More precisely, given $A_{i(0)}$ and $A_{i(1)}$, the adversary A has a non-negligible advantage to distinguish the tuple $< C_1, C_2, C_{3(0)} >$ from $< C_1, C_2, C_{3(1)} >$, where $C_1 = u^\zeta$, $C_2 = v^\beta$, $C_{3(0)} = A_{i(0)} h^{\zeta+\beta}$ and $C_{3(1)} = A_{i(1)} h^{\zeta+\beta}$. From Definition 6, we know that $< C_1, C_2, C_{3(b} >$ is a DLE tuple. If A has the ability to distinguish $< C_1, C_2, C_{3(0)} >$ from $< C_1, C_2, C_{3(1)} >$, it means A can break DLE problem, and it contradicts with Assumption 3. Thus it is impossible for A to distinguish δ_0 from δ_1 with a non-negligible probability, and Scheme 1 is fully anonymous. □

Theorem 3. ***(Traceability)*** *Scheme 1 is fully traceable if q-SDH is hard in* G_1 *and* G_2*. More specifically, if an adversary A with polynomially bounded computation ability can find a valid signature* $\delta =< M, C_1, C_2, C_3, C_4, c, CT_1, \cdots, CT_{|N_A'|}, s_\zeta, s_\beta, s_\alpha, s_\varepsilon, s_x, s_{\delta_1}, s_{\delta_2}, \Psi' >$*, then it can find a find a SDH pair and thus break the q-SDH problem.*

Proof. The proof is based on Forking Lemma. Suppose adversary A can forge a valid signature $\delta =< M, \delta_0, c, \delta_1, \delta_2 >$, where $\delta_0 = \{C_1, C_2, C_3, C_4\}$, $c = H(M, C_1, C_2, C_3, C_4, R_1, R_2, R_3, R_4, R_5, R_{Att})$ as computed during signature generation procedure described in Subsection 4.4.3, $\delta_1 = \{s_\zeta, s_\beta, s_\alpha, s_\varepsilon, s_x, s_{\delta_1}, s_{\delta_2}, \Psi'\}$, and $\delta_2 = \{CT_1, \cdots, CT_{N_A'}\}$. According to Assumption 4, we can extract a tuple $< \delta_0, c', \delta_1', \delta_2 >$ from $\delta =< \delta_0, c, \delta_1, \delta_2 >$, where $c' \neq c$ and $\delta_1' \neq \delta_2'$. Then based on $< \delta_0, c', \delta_1', \delta_2 >$ and $c' \neq c$ and $\delta_1' \neq \delta_1$, we can create a new SDH tuple denoted by $< A_i', x' >$, which is

presented as **Theorem 5.3.5** in details in [13] and we will not repeat it here. If adversary A can create a q-SDH pair without the knowledge of γ, it can break q-SDH problem. It contradicts to Assumption 1 and thus Theorem 3 holds. □

Among the above security requirements, unforgeability and coalition resistance can be derived from traceability, and unlinkability can be derived from anonymity. The general idea is as follows. Suppose S is a system that provides both anonymity and traceability. If S is not secure against unforgeability, it means that an adversary A can forge a valid signature δ on behalf of a valid user U_i. Similarly, if S is not coalition resistant, an adversary A can corrupt a few users and then use their private keys to generate a valid signature δ. In both cases, when δ is handed over to an opener, the identity revealed is U_i instead of the real signer A. It contradicts with "traceability" (Theorem 3) and thus both unforgeability and coalition resistance can be inferred by traceability. Anonymity means that the system does not leak any useful information about signers given their signatures. From the description of unlinkability, we can see that it is also a kind of user identity information, meaning that anonymity is a stronger security requirement than unlinkability. Therefore, unlinkability can be deduced from anonymity.

4.5 ABA Scheme with One-time Attribute Trees

Scheme 1 proposed in Section 4.4 has brought the flexibility of generating different attribute trees without regenerating attribute keys. However, Scheme 1 still has two drawbacks. First of all, the flexibility described above is achieved by increasing the costs of storage and computation for a big central attribute tree. The second drawback is related to the representation ability of its attribute trees. Even though different attribute subtrees can be obtained by simplifying the central attribute tree, the simplified attribute trees have to be based on the central attribute tree.

To make it more clear, consider the example from Subsection 4.2.2. The central attribute tree Γ^{Ext} is based on attribute set $\{att_1, \cdots, att_5\}$ and the logic representation $(att_1 \vee att_2) \vee (att_3 \wedge (att_4 \wedge att_5))$. As shown in Fig. 4.3, based on the attribute subset $\Psi_i = \{att_1, att_3\}$ and the logic representation $att_1 \vee att_3$, the central attribute tree Γ^{Ext} has been simplified and attribute subtree Γ_i has been built. Similarly, other attribute subtrees based on different logic representations can also be achieved by simplifying Γ^{Ext}, but there exists a limitation here. For instance, we can never get an attribute subtree based on logic representation $att_1 \wedge att_3$ from Γ^{Ext}, which is built

from the logic presentation $(att_1 \vee att_2) \vee (att_3 \wedge (att_4 \wedge att_5))$. As a result, we have to rebuild another central attribute tree to get an attribute subtree related to logical representation $att_1 \wedge att_3$. This of course will require more storage and computation resources.

In order to solve both of the above problems, we will propose an efficient ABA scheme in this section such that it will be possible to generate new attribute trees at any time point without regenerating attribute keys or other system parameters. Since there is no need to store attribute trees that are built by this approach, we can call this type of attribute trees one-time attribute trees.

4.5.1 General Idea of One-time Attribute trees

In Subsection 3.2, we discussed two types of ABA schemes, i.e., KP-ABA and CP-ABA. Attribute keys generated in KP-ABA schemes partly depend on the relations among attributes, while attribute keys in CP-ABA schemes are only based on the attributes and independent of their relations. Scheme 1 proposed in Section 4.4 has gained some flexibility and independence between attributes keys and their relations, but this flexibility is still restricted as discussed at the beginning of this chapter. If we go deeper into the technical details about how attribute trees are generated, we can summarize the differences among attribute trees in KP-ABA schemes, Scheme 1 and Scheme 2 (will be proposed later in this section) into the following three aspects:

1. In KP-ABA schemes [13], an attribute tree Γ with root value rt is built first. Attribute keys are generated based on the root value rt such that attribute keys are bound to both attributes and their relations.

2. In KP-ABA schemes, attribute keys are generated based on an attribute tree. However, attribute keys are generated first in Scheme 1, and then based on these attribute keys, a large central attribute tree Γ is built. Next, different attribute subtrees (denoted by Γ_i) can be obtained by simplifying this big central attribute tree. Assume f and f_i represent the logical statements expressed by attribute trees Γ and Γ_i respectively. Then f_i cannot be a random logical statement but has to be deduced from f. Therefore, the flexibility of attribute tree generation is limited.

3. In Scheme 2, attribute keys are also generated first (as in Scheme 1). The difference is that we do not need to build a central attribute tree first and then

simplify it to obtain attribute subtrees later. Each time before authentication, verifiers build a fresh attribute tree and it will be used only once. Therefore, there is no need to store attribute trees and related parameters in the system. This flexibility is achieved by using different underlying cryptographic primitives during the signature generation and verification phase. More details will be available in Subsection 4.5.3.

4.5.2 Building Blocks: Algorithms

Before describing the cryptographic construction of Scheme 2, we define its algorithms to give a general idea how the scheme works.

• **Scheme2.spg**($\mathbf{k_0}$)$\rightarrow$ $\{\mathbf{MPK}, \mathbf{MSK}\}$: The algorithm for system parameter generation takes system security parameter k_0 as its input and outputs the system public key set MPK and private key set MSK.

• **Scheme2.akg**($\mathbf{MPK}, \Psi$)$\rightarrow$ $\{\mathbf{APK}_\Psi, \mathbf{ASK}_\Psi\}$: The algorithm for attribute key generation takes the system public key set MPK and attribute set Ψ as input and outputs the public attribute key set APK_Ψ and the private attribute key set ASK_Ψ.

• **Scheme2.ukg**($\mathbf{MPK}, \mathbf{MSK}, \mathbf{U_i}$)$\rightarrow$ $\mathbf{bsk_i}$: The algorithm for user key generation takes the system public key set MPK, private key set MSK and the identity of U_i as input and outputs U_i's private key base $bsk_i =< A_i, x_i >$, where x_i is a secret value based on which user U_i's user key A_i is generated.

• **Scheme2.uakg**($\mathbf{MPK}$, $\mathbf{A_i}$, Ψ_i)$\rightarrow$ $\mathbf{USK_i}$: The algorithm for user attribute key generation takes the system public key set MPK, U_i's private key A_i and its attribute key set Ψ_i as input and outputs U_i's private attribute key set USK_i.

• **Scheme2.atg**(Ψ')$\rightarrow$ $\{\Gamma', \mathbf{PTree}_{\Psi'}\}$: This algorithm provides attribute tree generation and is carried out by the verifier. It takes attribute set Ψ' as input and generates attribute tree Γ' and related parameters $PTree_{\Psi'}$.

• **Scheme2.sg**($\mathbf{MPK}, \mathbf{MSK}$, Γ, Ψ', $\mathbf{USK_i}$, $\mathbf{M}$)$\rightarrow$ δ The algorithm for signature generation takes the system public and private key sets MPK and MSK, attribute tree Γ, the selected attribute set Ψ', U_i's private key set USK_i and message M as input and outputs the related signature δ.

• **Scheme2.sv**($\mathbf{M}, \delta, \Psi'$)$\rightarrow$ **b**: The algorithm for signature verification takes message M together with its signature δ and the selected attribute set Ψ' as input to check whether the signature δ is valid or not. It outputs the checking result $b \in \{0,1\}$.

• **Scheme2.so**($\mathbf{M}, \delta, \Psi'$, **tk**)$\rightarrow$ $\mathbf{U_i}$: The algorithm for signature opening takes message M together with its signature δ, the selected attribute set Ψ' and the tracking key tk as its input to trace the identity information of the signer U_i, which is U_i's index i here.

4.5.3 Scheme Construction

In this subsection, we introduce definitions that will be used in the construction of Scheme 2. We explain important notation in Table. 4.2.

Definition 10. ***(Secure Keyed One-Way Hash Functions** [75]) $H_K : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a keyed hash function that generates a message digest of length m. H_K is a secure keyed one-way hash function if it satisfies the following properties:*

1. *Given K and message M, it is easy to compute $H_K(M)$.*
2. *Without the knowledge of K, it is difficult to recover M given $H_K(M)$ or to compute $H_K(M)$ given M.*
3. *Without the knowledge of K, it is hard to find a pair of different messages M_1 and M_2 such that $H_K(M_1) = H_K(M_2)$.*
4. *Given $H_K(M_1)$ but without the knowledge of K, it is hard to find a different message M_2 such that $H_K(M_1) = H_K(M_2)$.*

Now, we provide detailed description of algorithms about how to cryptographically construct Scheme 2. Scheme 2 follows the workflow as the traceable ABA schemes described in Subsection 3.2. It includes two phases, i.e., system setup and signature generation, verification and possibly opening.

System setup First of all, the ABA system should be set up before signature generation and verification. The system setup phase includes the following four algorithms.

1. **System parameter generation (Scheme2.spg)** This algorithm is performed by the central authority. Assume k_0 is the security parameter, G_1 and G_2 are

Table 4.2: Notation for Scheme 2

Notations	Description
k_0	The system security parameter.
$\mathbb{Z}_p^*$	The set of integers modulo p, where p is a large prime integer.
G_i	(Multiplicative) cyclic group of prime order p, where $1 \leq i \leq 2$.
e	$e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map.
H	$H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function [69].
$H_K()$	A secure keyed one-way hash function with K as the key.
U	$U =< U_1, \cdots, U_{N_u} >$ is the system user set.
N_u	$N_u = \|U\|$, the number of users in U.
Ψ	The system attribute set.
Ψ_i	$\Psi_i \subset \Psi$, attribute subset owned by U_i.
Ψ'	$\Psi' \subset \Psi_i$, attribute subset required by the signer.
N_a	$N_a = \|\Psi\|$, the number of attributes in set Ψ.
N_i	$N_i = \|\Psi_i\|$, the number of attributes in set Ψ_i.
MPK	The system public parameter set.
MSK	The system private parameter set.
bsk_i	The private key base of U_i and $bsk_i = \{A_i, x_i\}$.
apk_j	Public attribute key of attribute att_j, where $1 \leq j \leq N_a$.
ask_j	Private attribute key of attribute att_j, where $1 \leq j \leq N_a$.
$T_{i,j}$	U_i's private attribute key related to attribute att_j.
APK_Ψ	Public attribute key set related to attribute set Ψ.
ASK_Ψ	Private attribute key set related to attribute set Ψ.
USK_i	Private key set of U_i.
Γ'	Attribute tree generated based on attribute set Ψ'.
$\mathbb{L}(\Gamma')$	The leaf node set of attribute tree Γ'.
rt'	Root of attribute tree Γ'.
$ind(x)$	The index of node x.
V	The verifier.
M	The message based on which U_i generates the signature.
δ	Signature generated by user U_i based on attribute subset Ψ'.
K_v	The shared key between the verifier and the signer, computed by the verifier.
K_s	The shared key between the verifier and the signer, computed by the signer.

two multiplicative groups of prime order p and $e : G_1 \times G_1 \to G_2$ is a bilinear group. Assume DeLP and q-SDH are hard to solve in G_1. DLP is hard to solve in both G_1 and G_2. Select $g_1 \in G_1$ and $g_2 \in G_2$ as their generators. Select a secure keyed one-way hash function $H_K : \{0,1\}^* \to \mathbb{Z}_p^*$. Select $h \in G_1$ and $\xi_1, \xi_2 \in \mathbb{Z}_p^*$. Set $u, v \in G$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select $x_0 \in \mathbb{Z}_p^*$ as the top secret. The system public key is $MPK =< G_1, G_2, g_1, g_2, h, u, v, w_0 = g^{x_0} >$ and the system private key is $MSK =< x_0, \xi_1, \xi_2 >$. The pair $< \xi_1, \xi_2 >$ should be handed to the opener and will be used as the tracing key to open signers' signatures later.

2. **Attribute key generation (Scheme2.akg)** This algorithm is run by the attribute authority. Assume the system attribute set is $\Psi = \{att_1, \cdots, att_{N_a}\}$, where $N_a = |\Psi|$ is the size of Ψ. For all att_j $(1 \le j \le N_a)$ randomly select a value $t_j \in \mathbb{Z}_p^*$ as the secret attribute key ask_j and compute $apk_j = g_1^{t_j}$ as the public attribute key. The system public attribute key set is $APK_\Psi =< g_1^{t_1}, \cdots, g_1^{t_{N_a}} >$ and the system private attribute key set is $ASK_\Psi =< t_1, \cdots, t_{N_a} >$.

3. **User key generation (Scheme2.ukg)** This algorithm is carried out by the attribute authority. Suppose the system user set is $U =< U_1, \cdots, U_{N_u} >$, where $N_u = |U|$ and $1 \le i \le N_u$. For each user U_i $(1 \le i \le N_u)$, randomly select $x_i \in \mathbb{Z}_p^*$ and compute $A_i = g_1^{1/(x_0+x_i)}$. Then U_i's private key base is $bsk_i =< A_i, x_i >$. In a traceable ABA system, A_i should be registered in the opener's database.

4. **User attribute key generation (Scheme2.uakg)** Assume Ψ_i is the attribute subset owned by U_i and $N_i = |\Psi_i|$ is the size of Ψ_i. For an attribute $att_j \in \Psi_i$ $(1 \le j \le N_i)$, U_i's private attribute key is computed as $T_{i,j} = g_1^{x_i} H(att_j)^{t_j}$. Then U_i's private key set is denoted by $USK_i =< A_i, x_i, T_{i,1}, \cdots, T_{i,N_i} >$.

Signature generation and verification Once the ABA system is setup, the authentication between the verifier and the signer can proceed as follows.

1. **Verifier: attribute tree generation (Scheme2.atg)** Assume the attribute set required by the verifier is denoted by Ψ' and the related attribute tree is Γ'. The verifier randomly chooses $\alpha \in \mathbb{Z}_p^*$ and computes $K_v = e(g_1, w_0)^\alpha = e(g_1, g_1)^{\alpha x_0}$. Based on Ψ', the verifier builds the attribute tree Γ' with rt' as its root as described in Subsection 4.2.1 satisfying $q_{rt}(0) = \alpha$. Let $\mathbb{L}(\Gamma')$ be the leaf set of Γ'. For all $y \in \mathbb{L}(\Gamma')$, compute $C_y = g_1^{q_y(0)}$ and $C'_y = H(y)^{q_y(0)}$. The verifier sends $\{\Gamma', g_1^\alpha, \forall y \in \mathbb{L}(\Gamma') : C_y, C'_y\}$ to the signer.

2. **Signer: signature generation (Scheme2.sg)** Assume the signer is U_i and U_i possesses all the required attributes, so we have $\Psi_i \supset \Psi'$. Since y is the attribute leaf node in Γ' and therefore there must be an attribute $att_j \in \Psi_i$ $(1 \leq j \leq N_i)$, corresponding to it. For the convenience of representation, we will use y instead of att_j in the following. After receiving the message from the verifier, the signer calculates

$$DecryptNode(T_{i,j}, C_y, C'_y, y) = \frac{e(T_{i,j}, C_y)}{e(apk_j, C'_y)} = e(g_1, g_1)^{x_i q_y(0)}.$$

If x is an interior node, the algorithm *DecryptNode* proceeds as follows: for all x' children z, algorithm $DecryptNode(T_{i,j}, C_y, C'_y, y)$ is called and the output is stored as F_z. Assume S_x is the subset of all x's children z belonging to Ψ_i and define $\Delta_{S_x, index(z)} = \prod_{l \in \{S_x - index(x)\}} \frac{l}{index(z) - l}$. Then we have

$$\begin{aligned}
F_x &= \prod_{z \in S_x} F_z^{q_z(0)\Delta_{S_x, ind(z)}} \\
&= \prod_{z \in S_x} \left(e(g_1, g_1)^{x_i q_z(0)}\right)^{\Delta_{S_x, ind(z)}} \\
&= \prod_{z \in S_x} \left(e(g_1, g_1)^{x_i q_{parent(z)}(ind(z))}\right)^{\Delta_{S_x, ind(z)}} \\
&= e(g_1, g_1)^{x_i q_x(0)}.
\end{aligned}$$

The signer calls algorithm *DecryptNode* for the root and gets the result $F_{rt'} = e(g_1, g_1)^{\alpha x_i}$. Then it computes $K_s = e(A_i^{-1}, g_1^{\alpha}) / F_{rt'} = e(g_1, g_1)^{x_0 \alpha}$ and uses K_s to in signature generation later. Then the signer randomly selects $\zeta, \alpha, \beta, r_\zeta$, $r_\alpha, r_\beta, r_x, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p^*$ and calculates

$$\begin{aligned}
&C_1 = u^{\zeta}, C_2 = v^{\beta}, C_3 = A_i h^{\zeta+\beta}, \delta_1 = x_i \zeta, \delta_2 = x_i \beta, \\
&R_1 = u^{r_\zeta}, R_2 = v^{r_\beta}, R_4 = C_1^{r_x} u^{-r_{\delta_1}}, R_5 = C_2^{r_x} v^{-r_{\delta_2}}, \\
&R_3 = e(C_3, g_1)^{r_x} e(h, w_0)^{-r_\zeta - r_\beta} e(h, g_1)^{-r_{\delta_1} - r_{\delta_2}}, \\
&c = H_{K_s}(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p^*, \\
&s_\zeta = r_\zeta + c\zeta, s_\beta = r_\beta + c\beta, s_\alpha = r_\alpha + c\alpha, \\
&s_x = r_x + cx_i, s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_2} = r_{\delta_2} + c\delta_2.
\end{aligned}$$

In the end, the signer sends the signature $\sigma = < M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha,$

$s_{\delta_1}, s_{\delta_2} >$ to the verifier.

3. **Verifier: signature verification (Scheme2.sv)** After receiving σ, the verifier computes

$$R'_1 = u^{s_\zeta} C_1^{-c}, R'_2 = v^{s_\beta} C_2^{-c}, R'_4 = u^{-s_{\delta_1}} C_1^{s_x}, R'_5 = v^{-s_{\delta_2}} C_2^{s_x},$$
$$R'_3 = e(C_3, g_1)^{s_x} e(h, w_0)^{-s_\zeta - s_\beta} e(h, g_1)^{-s_{\delta_1} - s_{\delta_2}} (\frac{e(C_3, w_0)}{e(g_1, g_1)})^c.$$

   At last, it checks whether the equation $c = H_{K_v}(M, C_1, C_2, C_3, R'_1, R'_2, R'_3, R'_4, R'_5)$ holds. If so, the verifier believes that the signer owns the required attributes and accepts the signature, otherwise rejects it. In the above algorithms, the verification of the signer's attributes is realized by the security of the keyed one-way hash function.

Signature opening (Scheme2.so) The opener uses its tracing key $\{\varepsilon_1, \varepsilon_2\}$ to compute $A_i = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$, where A_i is considered as the identity information of U_i.

4.5.4 Correctness Analysis

Theorem 4. ***(Correctness)*** *The cryptographic construction of Scheme 2 proposed in Subsection 4.5.3 is correct, which means:*

1. K_v=K_s.
2. $A_i = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$ *holds.*
3. $R_i = R'_i$, *where* $1 \leq i \leq 5$.

Proof. 1) From the algorithm descriptions, we know that $K_v = e(g_1, g_1)^{x_0 \alpha}$. As long as the signer has the required attributes expressed by attribute tree Γ', it can compute K_s which also equals to $e(g_1, g_1)^{x_0 \alpha}$.
2) $C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2}) = A_i h^{\zeta+\beta}/((u^\zeta)^{\varepsilon_1}(v^\beta)^{\varepsilon_2}) = A_i$.
3) R'_i $(1 \leq i \leq 5)$ are calculated as follows:

$$R'_1 = u^{s_\zeta} C_1^{-c} = u^{r_\zeta + c\zeta} (u^\zeta)^{-c} = R_1$$
$$R'_2 = v^{s_\beta} C_2^{-c} = v^{r_\beta + c\beta} (v^\beta)^{-c} = R_2$$
$$R'_3 = e(C_3, g_1)^{s_x} e(h, w_0)^{-s_\alpha - s_\beta} e(h, g_1)^{-s_{\delta_1} - s_{\delta_2}} (\frac{e(C_3, w_0)}{e(g_1, g_1)})^c$$

$$= e(C_3, g_1)^{r_x + cx_i} e(h, w)^{-(r_\alpha + c\alpha) - (r_\beta + c\beta)} e(h, g_1)^{-(r_{\delta_1} + c\delta_1) - (r_{\delta_2} + c\delta_2)} (\frac{e(C_3, w)}{e(g_1, g_1)})^c$$

$$= R_3 \left(e(C_3, g_1)^{x_i} e(h, w_0)^{-(\alpha+\beta)} e(h, g_1)^{-(\delta_1+\delta_2)} \frac{e(C_3, w_0)}{e(g_1, g_1)} \right)^c$$

$$= R_3 \left(e(C_3 h^{-(\alpha+\beta)}, w_0 g_1^{x_i}) e(C_3, w_0)^{-1} \frac{e(C_3, w_0)}{e(g_1, g_1)} \right)^c$$

$$= R_3 \left(e(A_i, w_0 g_1^{x_i}) (g_1, g_1)^{-1} \right)^c = R_3$$

$$R'_4 = u^{-s_{\delta_1}} C_1^{s_x} = u^{-r_{\delta_1} - c\delta_1} (u^\zeta)^{r_x + cx_i} = u^{-r_{\delta_1} - c\zeta x_i} (u^\zeta)^{r_x + cx_i} = C_1^{r_x} u^{-r_{\delta_1}} = R_4$$

$$R'_5 = v^{-s_{\delta_2}} C_2^{s_x} = v^{-r_{\delta_2} - c\delta_2} (v^\beta)^{r_x + cx_i} = v^{-r_{\delta_2} - c\beta x_i} (v^\beta)^{r_x + cx_i} = C_2^{r_x} v^{-r_{\delta_2}} = R_5$$

From the above proof, we can come to the conclusion that Theorem 4 holds and the cryptographic construction of Scheme 2 is correct. □

4.5.5 Adversary Model

In this subsection, we introduce the adversary model for Scheme 2, and it is based on random oracle. We will use **Scheme2.** as prefix and .**Oracle** as suffix to represent these oracles. Details of these oracles are as follows.

• **Scheme2.Syspara.Oracle**: The system parameter oracle allows an adversary to query parameters generated by algorithm Scheme2.spg described in Subsection 4.5.2.

• **Scheme2.Userkey.Oracle**: An adversary can send U_i's index i to query the user key oracle about U_i's secret key base bsk_i.

• **Scheme2.Attkey.Oracle**: The attribute key oracle allows an adversary to query private attribute keys. To query about the private attribute key of attribute $att_j \in \Psi$ ($1 \leq j \leq N_a$), an adversary simply sends att_j to the oracle and the attribute key oracle will reply with ask_j. There is no need to query about apk_j because it is public and is already known to the adversary.

• **Scheme2.UserAttkey.Oracle**: The user attribute key oracle allows an adversary to query about a user's private attribute keys. Assume the user and the attribute the adversary wants to query are U_i and att_j respectively, where $1 \leq j \leq N_i$. To query about the private attribute key, the adversary sends U_i's index i and attribute att_j to the oracle. The oracle will reply with $T_{i,j}$.

• **Scheme2.Signature.Oracle**: The signature oracle allows an adversary to obtain a user's signature. Assume the user is U_i, the message to be signed is M

and the required attribute set is Ψ'. To query a signature based on $\{U_i, M, \Psi'\}$, the adversary sends $\{i, M, \Psi'\}$ to the oracle and the signature oracle and it will be replied with a valid signature δ based on $\{M, i, \Psi'\}$.

• **Scheme2.Opening.Oracle**: This oracle allows an adversary to query a signer U_i's index i. Assume the signature δ is generated by U_i based on $\{M, \Psi'\}$. To query the index of U_i, the adversary sends $\{M, \delta, \Psi'\}$ to the opening oracle and it will be replied with i.

First, we define a game called Scheme2.Game.Anonymity. Next we will use the above oracles to describe the adversary's abilities and provide definitions of the security requirements anonymity and traceability.

Scheme2.Game.Anonymity Assume A is an adversary with polynomially bounded computational abilities. This game proceeds as follows.

1. **System setup**: At the very beginning, the system runs the algorithms Scheme2.spg, Scheme2.akg, Scheme2.ukg and Scheme2.uakg to initiate the system. At this stage, adversary A can obtain all public parameters.

2. **Phase 1**: At this phase, the adversary A is allowed to query all oracles described above, with the exception that the tracing key tk should be kept secret to the adversary. Otherwise, given a signature, the adversary can trace the identity information of the signer and this game will make no sense at all.

3. **Challenge**: The adversary A decides to challenge now. It randomly selects two indexes i_b, $b \in \{0,1\}$, a message M and a attribute set Ψ'. The adversary sends $\{i_b, M, \Psi'\}$ to the system. The system replies with signature δ_b, which is computed based on $\{i_b, M, \Psi'\}$, $b \in \{0,1\}$. This challenge should have not been queried in phase 1.

4. **Phase 2**: The adversary A performs the same as described in phase 1. However, A is not allowed to query about the challenge described above.

5. **Guess**: The adversary A output its guess $b' \in \{0,1\}$. If b equals to b', A wins the game. Otherwise, A fails.

Definition 11. ***(Anonymity of Scheme 2)*** *Assume A is an adversary with polynomially bounded computational abilities. If the chance that A wins game* ***Scheme2.Game.Anonymity*** *is negligible, then Scheme 2 provides anonymity.*

Scheme2.Game.Traceability Assume the adversary is denoted by A with polynomially bounded computational abilities. The game proceeds as follows.

1. **System setup**: This is the system initiation phase, which is performed by running the algorithms Scheme2.spg, Scheme2.akg, Scheme2.ukg and Scheme2.uakg. The adversary can obtain all public parameters, the tracing key tk, users' private key base list, the system private attribute key set ASK and users' private attribute keys. Since adversary A has obtained the tracing key in this stage, it can behave like an opener now. Therefore, given a signature, A can reveal the identity information of the signer which is represented as the signer's index here.

2. **Phase 1**: In this phase, the adversary A is allowed to query all oracles that have been described above.

3. **Challenge**: The adversary decides to challenge now. It chooses an attribute requirement set Ψ', message M to be singed and the index i of user U_i whom it wants to impersonate.

4. **Phase 2**: The adversary A repeats phase 1, but A cannot query about the challenge.

5. **Output**: The adversary tries to forge a valid signature δ based on $\{i, M, \Psi'\}$ and then sends it to the opener. The opener opens the signature δ and gets the signer's index i'. If $U_{i'}$ has not been registered in the system, the opener outputs 1 and A wins the game. Otherwise the opener outputs 0 and A fails the game.

Definition 12. ***(Traceability of Scheme 2)*** *Assume A is an adversary with polynomially bounded computational abilities. If the chance that the adversary A wins the game* ***Scheme2.Game.Traceability*** *is negligible, then Scheme 2 provides traceability.*

4.5.6 Security Analysis

Theorem 5. ***(Anonymity)*** *Scheme 2 is fully anonymous if DLE is IND-CPA secure under the same attribute set. More specifically, given $A_{i(b)}$ with their corresponding signatures $\sigma_b =< M, C_1, C_2, C_{3(b)}, c, s_\zeta, s_\beta, s_\alpha, s_{\delta_1}, s_{\delta_2} >$ and a random toss $b \in \{0,1\}$, if an adversary A with polynomially bounded computation ability*

has a non-negligible advantage to guess the correct b, we say that the proposed scheme is not fully anonymous. Otherwise, the scheme is fully anonymous.

Proof. If adversary A can break the anonymity of the proposed scheme, A should have a non-negligible advantage to guess the correct b in the above statement. Given $A_{i(0)}$ and $A_{i(1)}$, one of which is the real signer's key, the adversary A has a non-negligible advantage to differentiate two signatures σ_0 and σ_1 which are related to $A_{i(0)}$ and $A_{i(1)}$ respectively. Therefore, A can distinguish the tuple $< C_1, C_2, C_{3(0)} >$ from the tuple $< C_1, C_2, C_{3(1)} >$ with non-negligible advantage. From the construction of the proposed scheme, we know that $< C_1, C_2, C_3 >$ can be considered as a DLE scheme with A_i as the message to be encrypted. According to Assumption 3, we know that $< C_1, C_2, C_3 >$ is secure against a chosen-plaintext attack. However, A can distinguish the tuple $< C_1, C_2, C_{3(0)} >$ from the tuple $< C_1, C_2, C_{3(1)} >$ with non-negligible advantage. It contradicts with each other. Therefore, the proposed scheme is fully anonymous. □

Theorem 6. ***(Traceability)*** *As long as the keyed one-way hash function H is collision resistant, Scheme 2 is fully traceable if given a valid signature* $\sigma =< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_{\delta_1}, s_{\delta_2} >$*, 1) the opener can trace the identity of the signer and 2) an adversary A cannot forge a valid signature* σ'.

Proof. We first prove the statement 1). According to the correctness analysis, we know 1) can be proven. Next we prove the statement 2). It can be divided into two situations: a) the adversary uses $\{M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_{\delta_1}, s_{\delta_2}\}$ to generate c to obtain the same signature σ; b) the adversary uses a different tuple $\{M', C_1', C_2', C_3', c', s_\zeta', s_\beta', s_\alpha', s_{\delta_1}', s_{\delta_2}'\}$ to generate a valid but different signature σ'. For situation a), the adversary needs the right K_v or K_s. Since DLP is hard to solve in G_1, the adversary cannot compute α from g_1^α, and therefore it cannot compute K_v via $e(g_1, g_1^{x_0})^\alpha$. The only way left is to compute as the signer does. However, from the algorithm described in Subsection 4.5.3, we can see that without the required attribute keys, it is impossible to recover K_s either. Therefore, it is impossible for A to generate the same signature σ. As a conclusion for situation b), without knowing the shared secret key K_v or K_s and given the security of keyed one-way hash function, it is out of the ability of the adversary A to forge a valid signature σ'. □

4.6 Conclusions

In this chapter, we have proposed two cases, i.e., Scheme 1 and Scheme 2, to demonstrate how cryptographically construct CP-ABA schemes. Both schemes are traceable and provides the possibility to build attribute trees dynamically. In Scheme 1, the dynamic is realized by the down-to-top attribute tree building approach explained in Subsection 4.2.2. In this approach, a big central attribute tree is built first, based on which specific attribute subtrees can be obtained by simplifying the central attribute tree. However, the storage and computation cost for this big central attribute tree is high, which makes this approach more valuable in a dynamic environment where attribute requirements change frequently. In addition, since the specific attribute subtrees are based on the central attribute tree, not all specific attribute subtrees can be obtained as described in the beginning of Subsection 4.5. To overcome both disadvantages, Scheme 2 was proposed. It is different from Scheme 1, since the dynamic of Scheme 2 is based on the signature generation and verification mechanism rather than merely the attribute tree building approach. By using this approach, different attribute trees can be generated for each authentication. Therefore, there is no need to store attribute trees and thus it can save the storage cost.

For both Scheme 1 and Scheme 2, we first defined necessary algorithms used in both schemes and then demonstrated how to construct them. Next, we provided correct analysis and the proofs of their security requirements. They are based on random oracles and game theory. We begun with the introduction of necessary oracles and then based on these oracles, we described the games related to anonymity and traceability respectively. Finally, we presented the security requirements of anonymity and traceability in both schemes as theorems with detailed proofs.

Chapter 5

HABA Schemes

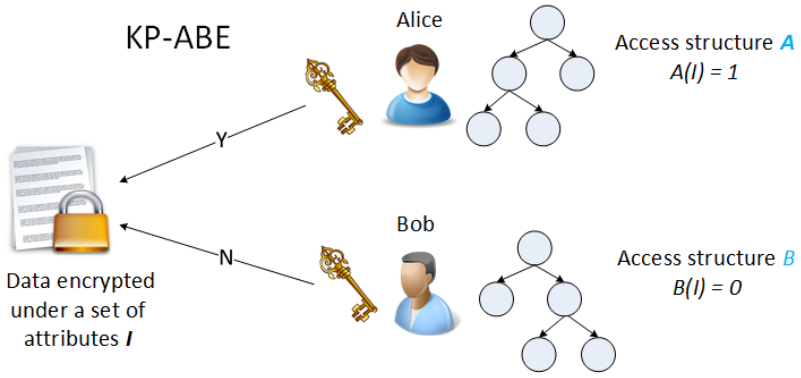
In Chapter 3, we discussed different types of ABA schemes, including HABA schemes [24, 45]. In this chapter, we will focus on the hierarchies and their cryptographically enforcement in HABA schemes. This chapter can be considered as future development of the results from [22]. We start with reviewing related work on attribute-related hierarchy. Since the use of hierarchies in ABA schemes is comparatively new but has already been well studied in ABE schemes [76–78], we will survey recent research results on hierarchical attribute-based encryption (HABE), compare it with ABA hierarchies and discuss how ABE hierarchies can be adopted and implemented in HABA schemes. Next, we will demonstrate the construction of two HABA schemes, of which one utilizes user-related hierarchies and the other utilizes attribute-related hierarchies. In addition, we provide detailed analysis of both schemes with respect to correctness and security requirements.

5.1 Related Work

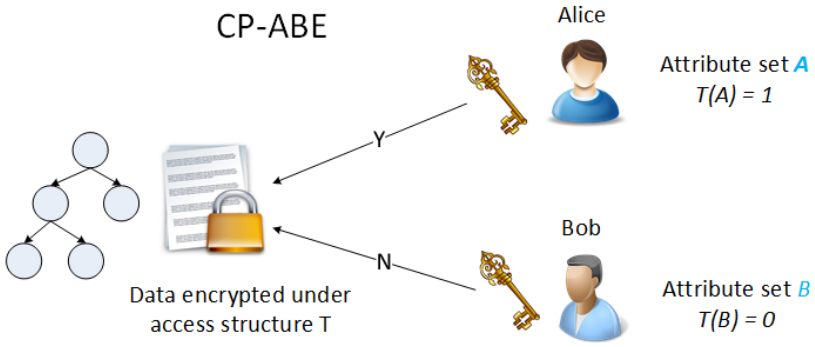
To our best knowledge, research results on HABA schemes are quite limited so far. Most work related to attribute-related hierarchies are based on HABE, and there are only few results in hierarchical attribute-based access control (HABAC) [79]. ABE is a special type of public key encryption, where data are encrypted by a set of attribute keys and only users with related decryption attribute keys can decrypt the ciphertext. One goal of ABE is fine-grained data access and it is widely used in cloud based services. Even though ABE and ABA schemes are different with respect to, for example, their main goals and security requirements, they still have a lot in common, such as attribute tree building and their fundamental cryptographic

primitives.

In this subsection, we will survey on recent research on HABE [80] in the rest of this section. ABE schemes can be divided into two types, KP-ABE [81–83] and CP-ABE [84, 85]. As shown in Fig. 5.1a, attribute keys generated in KP-ABE are associated with AC policies and data are encrypted according to descriptive sets of attributes [86]. Therefore, only users possessing attribute keys required by specific policies can decrypt the ciphertext. The situation for CP-ABE is different and can be described as Fig. 5.1b. Attribute keys are merely associated with attributes while data are encrypted according to access policies. HABE can be built either based on KP-ABE or CP-ABE schemes, so we will not classify whether the surveyed HABE schemes are KP-ABE or CP-ABE in the following.



(a) KP-ABE



(b) CP-ABE

Figure 5.1: CP-ABE and KP-ABE

There are mainly four types of hierarchies in ABE schemes, including delegation [45, 87–89], hierarchical structures of attributes [90], hierarchical structures of users [91] and hierarchical security classes [77]. In a delegation ABE system [88],

users are allowed to have the ability to generate attribute secret keys for other users, which can enable a more fine-grained access control in a large scale system. The hierarchical HABE scheme proposed in [90] is based on a hierarchical structure of attributes. It has introduced the concept of attribute "cover". In a system with n attribute trees, all attributes are divided into n trees with roots $\omega_{10}, \cdots, \omega_{n0}$. Assume that an attribute ω is from the ith tree and its depth is k, which means that the path from root ω_{i0} to ω is of length k. The path can be denoted by $(\omega_{i0}, \omega_{i1}, \cdots, \omega_{ik})$, where $\omega_{ik} = \omega$. It is said that ω covers ω' with the path $(\omega'_{j0}, \omega'_{j1}, \cdots, \omega'_{jk})$ if $\omega_{i\delta} = \omega'_{j\delta}$ holds for $0 \leq \delta \leq k$. Based on this tree hierarchy, the technology of hierarchical identity-based encryption (HIBE) is brought in and combined with secret sharing to achieve the proposed HABE scheme. Refer to [90] for more details. In [91], authors proposed a hierarchical attribute-set-based encryption (HASBE) scheme for cloud computing. In this scheme, users are organized in a hierarchical structure. There is a trust authority by which several domain authorities can be authorized. Both data owners and users are authorized by domain authorities with several discrete attributes or sets of attributes or both. By implementing this compound attribute authorization approach and user hierarchy structure, the scheme can gain more flexibility in attributes authorization as well as user revocation. Authors in [77] have described a scenario in the cloud as follows. As described in Fig. 5.2, documents are encrypted by key K_2 and then the encrypted documents are outsourced on the cloud servers. When general managers edit the documents, the edited segments are encrypted by key K_1. Similarly, document segments edited by bosses are encrypted by key K_0. Assume the hierarchical relation of these three keys can be represented by $K_2 \leq K_1 \leq K_0$, where K_2 is a child node of K_1 and similar for K_1 and K_0. According to the hierarchical relations described above, bosses can browse all content of the document, while general managers can only access those encrypted documents by either K_2 or K_1. The situation described above can be viewed as an application of classed security for attribute keys. Based on this scenario, authors of [77] proposed a practical document sharing scheme by combining ABE and hierarchical key management.

Except for HABE schemes, hierarchies have also been studied in several other attribute-based schemes, for example, a combination of HABAC with ABA in [24]. The schemes proposed in [24, 55] can be considered as a modified version of user-related HABE schemes. Authors of [79] proposed a hierarchical group and attribute-based access control (HGABAC) model, which is a combination of hierarchical group authentication and ABAC. The object group hierarchy allows users

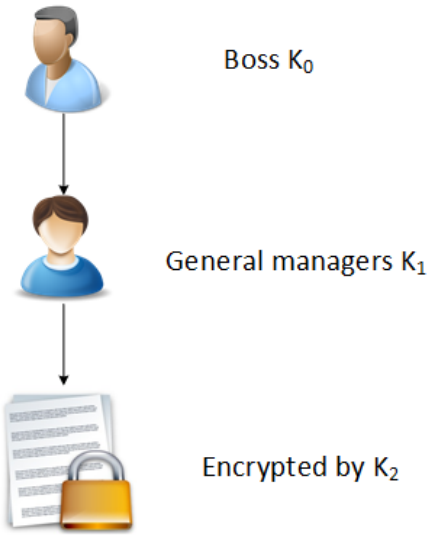


Figure 5.2: A scenario of classed security

to be categorized into collections of similar objects. Meanwhile, common attributes can be applied to the whole groups, which can therefore reduce work overhead. In addition, the model provides novel approaches to represent traditional models in an ABAC framework, such as discretionary access control (DAC) [92–94], mandatory access control (MAC) [95–97] and RBAC [98–102]. However, this work provides a definition of a HABAC model rather than cryptographic construction, which differs from our main focus in this chapter.

5.2 Motivations

There are some large-scaled virtual organizations which are collections of co-operating organizations. For the convenience of data sharing and better cooperation, they are united as a large-scaled virtual organization. However, on the one hand, organizations still need to manage and have control over their own departments and employees. On the other hand, it is more natural that employees in the same department are authorized with several common attributes. In addition, the chance that they can access the same resources is comparatively high. Given the above reasons, we can divide different organizations as different domain authorities, which can also authorize lower-leveled domain authorities. All domain authorities can authorize and manage users. This idea is practical and quite intuitive.

Another scenario is similar to the classed security (described in Fig. 5.2) for at-

tribute keys. Data are not always of the same importance or credential levels. Some data are public and can be viewed by every user on the Internet. However, some may be highly classified. For some highly classified data, before a user is allowed to access them, it is not enough to merely show that he possesses the required attributes but also he should show that his attribute keys are authorized with a certain privilege level. Only if the privilege level is equal to or above the required privilege level of the data, he can be granted with the right to access the data.

Based on the above two types of usage scenarios, we will study two types of HABA schemes and their cryptographic construction in this chapter, which are user-related and attribute-related HABA respectively. However, the meanings of user- and attribute-related for ABA schemes described in this dissertation differ from those we have discussed above for ABE schemes. The details will be explained later. In Sections 5.3 and 5.4, we will provide one example for each type of HABA schemes to demonstrate how to cryptographically construct them. Moreover, we will thoroughly analyze both schemes, including correctness and security analysis.

5.3 Case Study 1: A User-related HABA Scheme

In the rest of this dissertation, we will use Scheme 3 to denote the user-related HABA scheme presented in this section. In Scheme 3, users are organized in a hierarchical structure. Similar to organizations and departments in a large-scaled virtual organization, authorities that manage users are divided into different domain authorities of different levels. Authorities of a higher level can authorize authorities of a lower level. Scheme 3 follows the workflow described in Subsection 3.1.4. Its main idea can be illustrated as Fig. 5.3 and the details are expressed as follows.

1. The authority of level 0 is a trusted authority and the rest are domain authorities.

2. The trusted authority has the highest privilege and the level N authorities have the lowest privilege.

3. Authorities of level i ($1 \leq i \leq N$) can only be authorized by an authority of level $i-1$.

4. The trusted authority can only authorize domain authorities rather than users, while all domain authorities are able to authorize both users and domain authorities of a lower level.

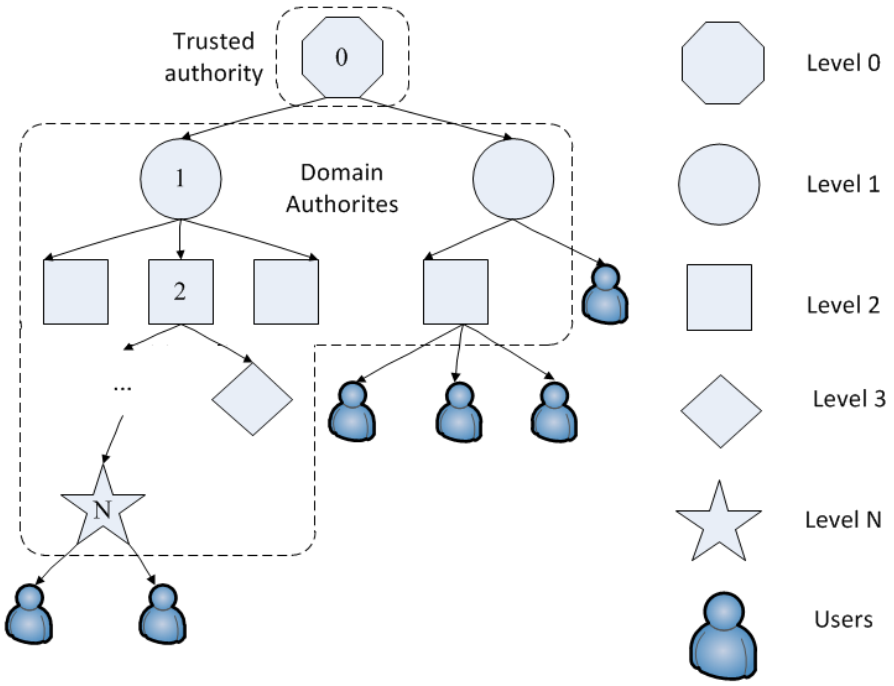


Figure 5.3: Structure of user-related HABA schemes

5. Users can be authorized by different domain authorities only in their authorization domains.

6. We will use $Auth_0$ and $Auth_i$ $(1 \leq i \leq N)$ to denote the trusted authority (level 0) and domain authorities of level i respectively.

By implementing the HABA structure described above, data owners cannot only define authentication attribute requirements but also specify that only users belonging to a specific domain authority can be authenticated. Therefore, the authentication can be more fine-grained and more flexible.

5.3.1 Building Blocks: Algorithms

To gain a big view of Scheme 3, we will first define all building blocks.

• **Scheme3.spg($\mathbf{k_0}$)**$\rightarrow \{\mathbf{MPK}, \mathbf{MSK}\}$: This algorithm is performed by the trusted authority to generate system parameters. Its input is the system security parameter k_0 and its output consists of the system public key set MPK and private key set MSK.

• **Scheme3.akg**($\mathbf{MPK}, \Psi$)$\rightarrow \{\mathbf{APK}_\Psi, \mathbf{ASK}_\Psi\}$: This algorithm is performed by the attribute authority to generate system attribute keys. Its input is the system public key set MPK and the system attribute set Ψ. Its output is the public attribute key set APK_Ψ and the private attribute key set ASK_Ψ.

• **Scheme3.auth$_1$**($\mathbf{MPK}$)$\rightarrow \{\mathbf{PK}_{\mathbf{Auth_1}}, \mathbf{SK}_{\mathbf{Auth_1}}\}$: This algorithm is performed by the trusted authority to authorize domain authorities of level 1, i.e., $Auth_1$. Its input is the system public key set MPK and its output is $Auth_1$'s public key set PK_{Auth_1} and private key set SK_{Auth_1}.

• **Scheme3.auth$_i$**($\mathbf{MPK}$, $\mathbf{PK}_{\mathbf{Auth_{i-1}}}, \mathbf{SK}_{\mathbf{Auth_{i-1}}}$) $\rightarrow \{\mathbf{PK}_{\mathbf{Auth_i}}, \mathbf{SK}_{\mathbf{Auth_i}}\}$: This algorithm is performed by domain authority $Auth_{i-1}$ ($1 \leq i \leq N$) to authorize domain authorities of level i ($Auth_i$). When i equals to 1, it means that domain authority $Auth_1$ is authorized by the trusted authority $Auth_0$. Its input is the system public key set MPK, $Auth_{i-1}$'s public key set $PK_{Auth_{i-1}}$ and private key set $SK_{Auth_{i-1}}$. Its output is $Auth_i$'s public key set PK_{Auth_i} and private key set SK_{Auth_i}.

• **Scheme3.ukg**($\mathbf{MPK}, \mathbf{PK}_{\mathbf{Auth_i}}, \mathbf{SK}_{\mathbf{Auth_i}}, \mathbf{U_k}$)$\rightarrow \mathbf{bsk_k}$: This algorithm is performed by domain authority $Auth_i$ to generate user keys for U_k. Its input is the system public key set MPK, public key set PK_{Auth_i} and private key set SK_{Auth_i} of domain authority $Auth_i$ and the identity of U_k. Its output is U_k's private key base $bsk_k =< A_{i,k}, u_k >$.

• **Scheme3.uakg**($\mathbf{MPK}, \mathbf{A_{i,k}}, \Psi_k$)$\rightarrow \mathbf{USK_k}$: This algorithm is performed by the attribute authority to generate users' attribute keys. Its input is the system public key set MPK, U_k's private key $A_{i,k}$ and its attribute set Ψ_k. Its output is U_k's private attribute key set USK_k.

• **Scheme3.atg**(Ψ')$\rightarrow \{\Gamma', \mathbf{PTree}_{\Psi'}\}$: This algorithm is performed by the verifier to generate an attribute tree based on the selected attribute set Ψ'. Its input is attribute set Ψ'. Its output is attribute tree Γ' with its related parameters $PTree_{\Psi'}$.

• **Scheme3.sg**($\mathbf{MPK}, \mathbf{MSK}$, Γ', Ψ', $\mathbf{USK_k}$, $\mathbf{M}$)$\rightarrow \delta$ This algorithm is performed by the user U_k to generate a signature. Its input is the system public and private key sets MPK and MSK, attribute tree Γ', selected attribute set Ψ', U_k's private key set USK_k and the message M. Its output is signature δ.

• **Scheme3.sv**($\mathbf{M}, \delta, \Psi'$)$\rightarrow$ **b** (**b** $\in \{\mathbf{0}, \mathbf{1}\}$): This algorithm is performed by the verifier to check whether the signature δ is valid. Its input is message M together with its signature δ and the selected attribute set Ψ'. Its output is the checking result $b \in \{0,1\}$.

• **Scheme3.so**($\mathbf{M}, \delta, \Psi'$, **tk**)$\rightarrow$ $\mathbf{U_k}$: This algorithm is performed by the opener to retrieve the signer's identity out of the signature δ. Its input is the message M together with its signature δ, the selected attribute set Ψ' and the opener's tracing key tk. Its output is the signer U_k's index k.

5.3.2 Scheme Construction

The basic cryptographic primitives for which Scheme 3 are the same for Scheme 1 and Scheme 2. We will utilize the top-to-down approach to generate attribute trees as explained in Subsection 4.2.1. More details can be found in Chapter 4. In the rest of this subsection, we will use the user-related structure described in Fig. 5.3 to explain the cryptographic construction of Scheme 3. The whole process consists of two parts: 1) system setup and 2) signature generation, verification and opening. Important notations in Scheme 3 and their meanings are listed in Table 5.1. The algorithm details are given below.

System setup There are five types of authorities in the system: the trusted authority, domain authorities, the attribute authority responsible for attribute key generation, the revocation authority that manages revocation information and the opener that traces signers' identities. The path from the authority of level N to the top authority (illustrated in Fig. 5.3) will be used as an example to demonstrate how to construct a user-related HABA scheme. The system setup includes five phases and can be described as follows.

1. **System parameter generation (Scheme3.spg)** This phase is performed by the trusted authority $Auth_0$. Assume k_0 is the system security parameter, and G_1, G_2 are two multiplicative groups of prime order p with $g_1 \in G_1$ and $g_2 \in G_2$ as their generators respectively. Let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear mapping. Assume DeLP is hard to solve in G_1. DLP is hard to solve in both G_1 and G_2. Select $h \in G_1$ and $\xi_1, \xi_2 \in \mathbb{Z}_p^*$ and set $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select $x_0, \beta_0 \in \mathbb{Z}_p^*$ as the top secret and compute $w_0 = g_1^{x_0}$, $f_0 = g_1^{1/\beta_0}$ and $h_0 = g_1^{\beta_0}$. The public key set of $Auth_0$ is $MPK =< G_1, G_2, g_1, g_2, h, u, v, f_0, h_0, w_0 >$ and the private key set is $MSK =<$

Table 5.1: Notation for Scheme 3

Notations	Description
k_0	The system security parameter.
$\mathbb{Z}_p^*$	The set of integers modulo p, where p is a large prime integer.
G_i	(Multiplicative) cyclic groups of prime order p, where $1 \leq i \leq 2$.
e	$e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map.
H	$H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function [69].
$H_K()$	A secure keyed one-way hash function with K as the key.
N	The total number of levels of domain authorities.
$Auth_0$	The trusted authority and its level is 0.
$Auth_i$	Domain authority of level i ($1 \leq i \leq N$).
U	$U =< U_1, \cdots, U_{N_u} >$ is the system user set.
N_u	$N_u = \|U\|$, the number of users in U.
Ψ	System attribute set.
Ψ_k	$\Psi_k \subset \Psi$, attribute subset owned by U_k ($1 \leq k \leq N_a$).
Ψ'	$\Psi' \subset \Psi_i$, attribute subset required by the signer.
N_a	$N_a = \|\Psi\|$, the number of attributes in attribute set Ψ.
N_k	$N_k = \|\Psi_k\|$, the number of attributes in attribute set Ψ_k.
MPK	The system public parameter set, or the public key set of the trusted authority.
MSK	The system private parameter set, or the private key set of the trusted authority.
PK_{Auth_i}	The public key set of domain authority $Auth_i$, $1 \leq i \leq N$.
SK_{Auth_i}	The private key set of domain authority $Auth_i$, $1 \leq i \leq N$.
bsk_k	The private key base of U_k. It is authorized by domain authority $Auth_i$ and denoted by $bsk_k = \{A_{i,k}, u_k\}$.
apk_j	Public attribute key of attribute att_j, where $1 \leq j \leq N_a$.
ask_j	Private attribute key of attribute att_j, where $1 \leq j \leq N_a$.
$T_{k,j}$	U_k's private attribute key of attribute att_j.
APK_Ψ	Public attribute key set related to attribute set Ψ.
ASK_Ψ	Private attribute key set related to attribute set Ψ.
USK_k	Private key set of user U_k.
Γ'	Attribute tree generated based on attribute set Ψ'.
$\mathbb{L}(\Gamma')$	The leaf node set of attribute tree Γ'.
rt'	The root of attribute tree Γ'.
$ind(x)$	The index of node x.
$par(z)$	The parent node of node z.
V	The verifier.
M	The message based on which U_i generates the signature.
δ	Signature generated by user U_i based on attribute subset Ψ'.
K_v	The shared key between the verifier and the signer, computed by the verifier.
K_s	The shared key between the verifier and the signer, computed by the signer.

$x_0, \beta_0, \xi_1, \xi_2 >$, where the pair $< \xi_1, \xi_2 >$ is handed to the opener as its tracing key tk.

2. **Attribute key generation (Scheme3.akg)** Assume the system attribute set is $\Psi = \{att_1, \cdots, att_{N_a}\}$, where $N_a = |\Psi|$. For each attribute att_j, the attribute authority selects a random value $t_j \in \mathbb{Z}_p^*$ as the private attribute key ask_j and computes $apk_j = g_1^{t_j}$. The system public attribute key is $APK_\Psi =< g_1^{t_1}, \cdots, g_1^{t_{N_a}} >$ and the system private attribute key is $ASK_\Psi =< t_1, \cdots, t_{N_a} >$.

3. **The authorization of domain authorities (Scheme3.auth$_i$)** This phase is carried out by domain authority $Auth_{i-1}$. Suppose the domain authority to be authorized is denoted by $Auth_i$ $(1 \leq i \leq N)$. To authorize $Auth_i$, $Auth_{i-1}$ randomly selects $x_i \in \mathbb{Z}_p^*$ and computes $A_i = A_{i-1} f_0^{x_i} = g_1^{(x_0+\cdots+x_i)/\beta_0}$ and $w_i = g_1^{x_i}$. The public key set of $Auth_i$ is $PK_{Auth_i} =< G_1, G_2, g_1, g_2, w_i >$ and the private key set of $Auth_i$ is $SK_{Auth_i} =< x_i, A_i >$.

4. **User key generation (Scheme3.ukg)** Suppose the user group $Auth_i$ manager is $U =< U_1, \cdots, U_{N_u} >$, where $N_u = |U|$ and $1 \leq k \leq N_u$. For each user U_k $(1 \leq k \leq n)$, $Auth_i$ randomly selects $u_k \in \mathbb{Z}_p^*$ and computes $A_{i,k} = A_i f_0^{u_k} = g_1^{(x_0+\cdots+x_i+u_k)/\beta_0}$. In a traceable ABA scheme, $A_{i,k}$ should be registered in the opener's database. U_k's private key base is $usk_k =< u_k, A_{i,k} >$.

5. **User attribute key generation (Scheme3.uakg)** In this phase, the user interacts with the attribute authority to generate its private attribute keys. For $att_j \in \Psi_k$ owned by U_k, U_k's private attribute key is computed as $T_{k,j} = A_i H(att_j)^{t_j} = g_1^{(x_0+\cdots+x_i+u_k)/\beta_0} H(att_j)^{t_j}$.

Signature generation and verification Once the system is set up, data owners can define their own authentication requirements, which will be implemented by cloud servers to authenticate users who want to access data.

1. **Verifier: attribute tree generation (Scheme3.atg)** Assume the attribute set the verifier requires is Ψ' and the signer to be authenticated should be authorized by $Auth_i$ $(1 \leq i \leq N)$, which is authorized by $Auth_{i-1}$ and so forth until it reaches the trusted authority $Auth_0$. To generate an attribute tree, the verifier first randomly selects a secret $\alpha \in \mathbb{Z}_p^*$. Then the verifier constructs an attribute tree Γ' with α as its root value as described in Subsection 4.2.1. Assume the root of Γ' is rt'. Let $\mathbb{L}(\Gamma')$ be the leaf node set of the attribute tree Γ'. The verifier computes $\forall y \in \mathbb{L}(\Gamma'), C_y = g_1^{q_y(0)}$ and $C'_y = H(y)^{q_y(0)}$. Meanwhile, the

verifier computes $K_s = (e(f_0, g_1^{x_0}) \cdots e(f_0, g_1^{x_i}))^{\alpha} = e(g_1, g_1)^{(x_0+\cdots+x_i)\alpha/\beta_0}$. In the end, the verifier sends $\{\Gamma', g_1^{\alpha}, \forall y \in \mathbb{L}(\Gamma') : C_y, C'_y\}$ to the signer.

2. **Signer: signature generation (Scheme3.sg)** Assume the attribute set owned by the signer is Ψ_k and it satisfies $\Psi \supset \Psi_k \supset \Psi'$. For each leaf node y of Γ', there must be an attribute $att_j \in \Psi_k$ $(1 \leq j \leq N_k)$ corresponding to it. After the signer receives the message from the verifier, it computes

$$\begin{aligned} &DecryptNode(T_{k,j}, C_y, C'_y, y) \\ &= \frac{e(T_{k,j}, C_y)}{e(apk_j, C'_y)} \\ &= e(g_1, g_1)^{(x_0+\cdots+x_i+u_k)q_y(0)/\beta_0}, \end{aligned}$$

where we use y instead of att_j for simplicity.

If x is an interior node, algorithm $DecryptNode(T_{k,j}, C_y, C'_y, y)$ proceeds as follows: for all x' children z, algorithm $DecryptNode(T_{k,j}, C_y, C'_y, y)$ is called and the output is stored as F_z. Assume S_x is the subset of all x's children z belonging to Ψ_k. We define $\Delta_{S_x, ind(z)} = \prod_{l \in \{S_x - ind(x)\}} \frac{l}{ind(z) - l}$. Then we have

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{q_z(0)\Delta_{S_x, ind(z)}} \\ &= \prod_{z \in S_x} (e(g_1, g_1)^{(x_0+\cdots+x_i+u_k)q_z(0)/\beta_0})^{\Delta_{S_x, ind(z)}} \\ &= \prod_{z \in S_x} (e(g_1, g_1)^{(x_0+\cdots+x_i+u_k)q_{parent(z)}(ind(z))/\beta_0})^{\Delta_{S_x, ind(z)}} \\ &= e(g_1, g_1)^{(x_0+\cdots+x_i+u_k)q_x(0)/\beta_0}. \end{aligned}$$

The signer calls algorithm $DecryptNode(T_{k,j}, C_y, C'_y, y)$ for the root rt' and gets the result $F_{rt'} = e(g_1, g_1)^{(x_0+\cdots+x_i+u_k)\alpha/\beta_0}$. Next the signer computes $K_s = F_{rt'} / e(f_0^{u_k}, g_1^{\alpha}) = e(g_1, g_1)^{(x_0+\cdots+x_i)\alpha/\beta_0} = K_v$. Then the signer randomly selects $\zeta, \alpha, \beta, r_\zeta, r_\alpha, r_\beta, r_k, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p^*$ and calculates

$$\begin{aligned} &C_1 = u^{\zeta}, C_2 = v^{\beta}, C_3 = A_{i,k} h^{\zeta+\beta}, \\ &\delta_1 = u_k \zeta, \delta_2 = u_k \beta, \\ &R_1 = u^{r_\zeta}, R_2 = v^{r_\beta}, R_4 = C_1^{r_k} u^{-r_{\delta_1}}, R_5 = C_2^{r_k} v^{-r_{\delta_2}}, \\ &R_3 = e(C_3, g_1)^{r_k} e(h, w_i)^{-r_\zeta - r_\beta} e(h, g_1)^{-r_{\delta_1} - r_{\delta_2}}, \end{aligned}$$

$$c = H_{K_s}(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p^*,$$
$$s_\zeta = r_\zeta + c\zeta, s_\beta = r_\beta + c\beta, s_\alpha = r_\alpha + c\alpha,$$
$$s_k = r_k + cu_k, s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_2} = r_{\delta_2} + c\delta_2.$$

Finally, the signer sends the signature $\sigma =< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$ to the verifier.

Verifier: signature verification (Scheme3.sv) The verifier computes

$$R_1' = u^{s_\zeta} C_1^{-c}, R_2' = v^{s_\beta} C_2^{-c},$$
$$R_4' = u^{-s_{\delta_1}} C_1^{s_k}, R_5' = v^{-s_{\delta_2}} C_2^{s_k},$$
$$R_3' = e(C_3, g_1)^{s_k} e(h, w_i)^{-s_\zeta - s_\beta} e(h, g_1)^{-s_{\delta_1} - s_{\delta_2}} (\frac{e(C_3, w_i)}{e(g_1, g_1)})^c$$

and checks whether $c = H_{K_v}(M, C_1, C_2, C_3, R_1', R_2', R_3', R_4', R_5')$ holds. If so, the verifier believes that the signer owns the required attributes and accepts the signature, or otherwise rejects it.

Signature Opening (Scheme3.so) This phase is performed by the opener. If the revealing of signer's identity is required, the opener computes $A_{i,k} = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$. Next the opener searches items in its database and compare them with $A_{i,k}$. If there is an item that equals to $A_{i,k}$, it returns the index k of user U_k.

5.3.3 Correctness Analysis

In this subsection, we analyze the correctness of Scheme 3.

Theorem 7. ***(Correctness)*** *The construction of Scheme 3 proposed in Subsection 5.3.2 is correct, that is, the following three equations are satisfied:*

1. K_v=K_s,
2. $R_i = R_i'$, *where* $1 \leq i \leq 5$, *and*
3. $A_{i,k} = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$ *holds.*

Proof. 1) From the construction algorithm in Subsection 5.3.2, we know that as long as the signer has the required attributes described in attribute tree Γ', it can compute K_s which also equals to $K_v = e(g_1, g_1)^{(x_0 + \cdots + x_i)\alpha/\beta_0}$.

2) R'_i $(1 \leq i \leq 5)$ are calculated as follows:

$$
\begin{aligned}
R'_1 &= u^{s_\zeta} C_1^{-c} = u^{r_\zeta + c\zeta} (u^\zeta)^{-c} = R_1 \\
R'_2 &= v^{s_\beta} C_2^{-c} = v^{r_\beta + c\beta} (v^\beta)^{-c} = R_2 \\
R'_3 &= e(C_3, g_1)^{s_k} e(h, w_i)^{-s_\alpha - s_\beta} e(h, g_1)^{-s_{\delta_1} - s_{\delta_2}} (\frac{e(C_3, w_i)}{e(g_1, g_1)})^c \\
&= e(C_3, g_1)^{r_k + cu_k} e(h, w_i)^{-(r_\alpha + c\alpha) - (r_\beta + c\beta)} e(h, g_1)^{-(r_{\delta_1} + c\delta_1) - (r_{\delta_2} + c\delta_2)} (\frac{e(C_3, w_i)}{e(g_1, g_1)})^c \\
&= R_3 \left(e(C_3, g_1)^{u_k} e(h, w_i)^{-(\alpha+\beta)} e(h, g_1)^{-(\delta_1 + \delta_2)} \frac{e(C_3, w_i)}{e(g_1, g_1)} \right)^c \\
&= R_3 \left(e(C_3 h^{-(\alpha+\beta)}, w_i g_1^{u_k}) e(C_3, w_i)^{-1} \frac{e(C_3, w_i)}{e(g_1, g_1)} \right)^c \\
&= R_3 \left(e(A_i, w_i g_1^{u_k}) (g_1, g_1)^{-1} \right)^c = R_3 \\
R'_4 &= u^{-s_{\delta_1}} C_1^{s_k} = u^{-r_{\delta_1} - c\delta_1} (u^\zeta)^{r_k + cu_k} \\
&= u^{-r_{\delta_1} - c\zeta u_k} (u^\zeta)^{r_k + cu_k} = C_1^{r_k} u^{-r_{\delta_1}} = R_4
\end{aligned}
$$

3) $C_3 / (C_1^{\varepsilon_1} C_2^{\varepsilon_2}) = A_{i,k} h^{\zeta+\beta} / ((u^\zeta)^{\varepsilon_1} (v^\beta)^{\varepsilon_2}) = A_{i,k}$. □

5.3.4 Adversary Model

Similar to presentation in Chapter 4, the adversary model for Scheme 3 is also based on random oracle and game theory. A is an adversary with polynomially bounded computational abilities and is allowed to query oracles in games **Scheme3.Game.Anonymity** and **Scheme3.Game.Traceability** that are to be described later in this subsection. We use *Scheme*3. as prefix and *.Oracle* as suffix to represent these oracles.

• **Scheme3.Syspara.Oracle**: This oracle allows the adversary A to query parameters generated by algorithm Scheme3.spg described in Subsection 5.3.2.

• **Scheme3.Userkey.Oracle**: This oracle allows the adversary A to query about users' secret key base. To query U_k's secret key base usk_k, A simply sends U_k's index k to the oracle and the oracle will reply with U_k's user key base bsk_k.

• **Scheme3.Attkey.Oracle**: This oracle allows the adversary A to query private attribute keys by sending the attribute $att_j \in \Psi$ $(1 \leq j \leq N_a)$ to the oracle,

and the oracle will reply with ask_j. It is unnecessary to query apk_j because it is a public parameter.

• **Scheme3.Auth.Oracle**: This oracle allows the adversary A to query parameters about domain authority $Auth_i$ by sending the index of the domain authority, which is denoted by i here.

• **Scheme3.UserAttkey.Oracle**: This oracle allows the adversary A to query users' private attribute keys. To query U_k's private key related to attribute att_j ($1 \leq j \leq N_i$), A sends U_k's index k and attribute att_j to the oracle, and the oracle will reply with $T_{k,j}$.

• **Scheme3.Signature.Oracle**: The signature oracle allows the adversary A to obtain a user U_k's signature by sending U_k's index k, message M to be encrypted and the selected attribute set Ψ'. The oracle will reply with a valid signature δ based on $\{M, k, \Psi'\}$.

• **Scheme3.Opening.Oracle**: This oracle allows the adversary A to query the signer's index by sending $\{M, \Psi', \delta\}$, where M is the message to be signed, Ψ' is the selected attribute set and δ is the signature generated based on M and Ψ'. The oracle will reply with the signer's index k.

Based on the above oracles, we have the following game which is named as **Scheme3.Game.Anonymity** and the game proceeds as follows.

1. **System setup**: The system is initiated by running the algorithms Scheme3.spg, Scheme3.akg, Scheme3.$auth_i$, Scheme3.ukg, and Scheme3.uakg. The adversary A can obtain all public parameters after the system initiation is completed.

2. **Phase 1**: The adversary A can queries all oracles described above in phase 1, but A cannot gain any knowledge of the tracing key tk. Otherwise, A can behave like an opener and trace signers' identities, which will make this anonymity game meaningless.

3. **Challenge**: The adversary A randomly selects two indexes i_b ($b \in \{0,1\}$), a message M and an attribute set Ψ'. Next A sends $\{i_b, M, \Psi'\}$ to start the challenge. Then A gets response with a signature δ_b ($b \in \{0,1\}$) which is computed based on $\{i_b, M, \Psi'\}$ ($b \in \{0,1\}$). The most important is that this challenge cannot have been queried in phase 1.

4. **Phase 2**: The behavior of adversary A in this phase is the same as described in phase 1, with the exception that A is not allowed to query about the challenge described above.

5. **Guess**: The adversary A shows its guess $b' \in \{0,1\}$. If $b = b'$ holds, A wins the game and otherwise it fails.

Definition 13. ***(Anonymity of Scheme 3)*** *Scheme 3 provides anonymity if the probability that an adversary A with polynomially bounded computational abilities can win the game* ***Scheme3.Game.Anonymity*** *is negligible.*

Next we introduce the traceability game of Scheme 3, which is denoted by **Scheme3.Game.Traceability** and the game proceeds as follows.

1. **System setup**: The system initiation is performed by running the algorithms Scheme3.spg, Scheme3.akg, Scheme3.$Auth_i$, Scheme3.ukg and Scheme3.uakg. The adversary can gain all public parameters. Moreover, the adversary A can obtain the tracing key, so it can act as an opener in this game.

2. **Phase 1**: The adversary A is allowed to query all oracles described above.

3. **Challenge**: The adversary decides to challenge now. It selects an attribute set Ψ', message M and the index k of user U_k whom it wants to impersonate.

4. **Phase 2**: The adversary A repeats phase 1 with the exception that A cannot query about the challenge described above.

5. **Output**: The adversary A tries to forge a signature δ based on $\{k, M, \Psi'\}$ and sends it to the opener. The opener checks the validity of the signature δ. If it is valid, the opener opens the signature and retrieves the signer's index which is denoted by k' here. If $U_{k'}$ is not a registered user, the opener outputs 1 and A wins this game. Otherwise the opener outputs 0 and A fails this game.

Definition 14. ***(Traceability of Scheme 3)*** *Scheme 3 provides traceability if the probability that an adversary A with polynomially bounded computational abilities can win the game* ***Scheme4.Game.Traceability*** *is negligible.*

5.3.5 Security Analysis

The security analysis in this subsection is based on Assumptions 2 and 3 described in Subsection 4.4.6 in Chapter 4 (please refer to Subsection 4.4.6 for more details).

Theorem 8. ***(Anonymity)*** *Scheme 3 is anonymous if DLE is IND-CPA secure. More specifically, given $A_{i,k(b)}$ with related signatures $\sigma_b =< M, C_1, C_2, C_{3(b)}, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$ with a random toss $b \in \{0,1\}$, if an adversary A with polynomially bounded computation abilities has a non-negligible advantage to guess the correct b, we say that the Scheme 3 is not anonymous. Otherwise, Scheme 3 is anonymous.*

Proof. Suppose that the adversary A has a non-negligible advantage to guess the correct b, and then A should also have a non-negligible advantage to differentiate σ_0 from σ_1. More precisely, given $A_{i,k(0)}$ and $A_{i,k(1)}$, the adversary A can distinguish tuple $< C_1, C_2, C_{3(0)} >$ from tuple $< C_1, C_2, C_{3(1)} >$. According to Definition 6, tuple $< C_1, C_2, C_{3(b)} >$ can be considered as the ciphertext of $A_{i,k(b)}$ based on DLE. Thus, if the adversary A has a non-negligible advantage to guess the guess b, it means A can break DLE, which contradicts to Assumption 3. Therefore, the proposed scheme is thus anonymous. □

Theorem 9. ***(Traceability)*** *Assume the keyed x hash function H is collision resistant. Scheme 3 is fully traceable if 1) given a valid signature $\sigma =< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$, the opener can trace the identity of the signer and 2) an adversary A with polynomially bounded computation ability cannot forge a valid signature σ' that can pass the verifier's verification.*

Proof. From the correctness proof, we know that statement 1) is correct. Next we prove the correctness of statement 2). One the one hand, since DLP is hard to solve in G_1, it is impossible for the adversary A to compute α from g_1^α, so the adversary cannot compute K_v by $(e(g_1, g_1)^{x_0+\cdots+x_i})^\alpha$. On the other hand, A doesn't own the secret value $T_{k,j}$, so it is also impossible for A to recover K_s as the signer does. As a result, the only chance for the adversary is either to compute the correct hash value based on $< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$ or to find a different pair $(< M', C_1', C_2', C_3', s_\zeta', s_\beta', s_\alpha', s_k', s_{\delta_1}', s_{\delta_2}' >, c')$ satisfying $c' = H_{K_s}(M', C_1', C_2', C_3', R_1', R_2', R_3', R_4', R_5')$ without the knowledge of either K_s or K_v. Since we assume H is collision resistant, neither of these two solutions is possible. Therefore, Scheme 3 is traceable. □

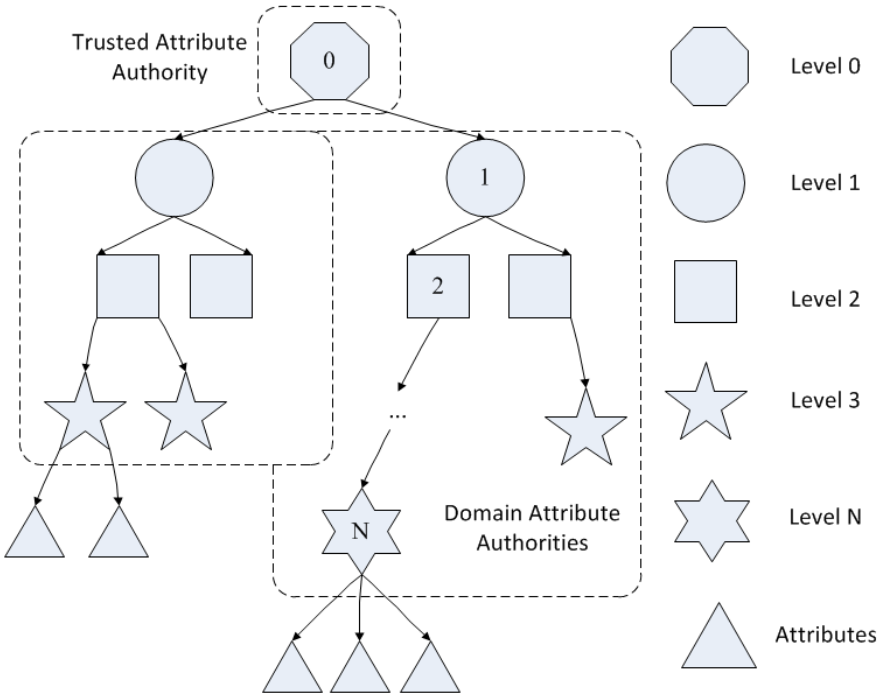


Figure 5.4: Structure of attribute-related HABA schemes

5.4 Case Study 2: An Attribute-related HABA Scheme

In this section, we will demonstrate how to construct an attribute-related HABA scheme. We refer to the scheme proposed in this subsection Scheme 4. Scheme 4 follows the workflow of attribute-related HABA schemes described in Subsection 3.1.5. The main idea of Scheme 4 is as shown in Fig. 5.4 and can be explained as follows.

1. All authorities in Fig. 5.4 are attribute authorities.
2. There are $N+1$ levels attribute authorities.
3. The authority of level 0 is the trusted authority (denoted by $Auth_0$) in the system and it has the highest trust level.
4. The domain authorities of level N have the lowest trust level.
5. Attribute authorities of trust level i ($1 \leq i \leq N$) are denoted by $Auth_i$ and they should only be authorized by an authority of trust level $i-1$.

6. The trusted authority can only authorize domain attribute authorities.
7. Attribute keys should only be authorized by domain authorities.
8. Attribute keys can be authorized by different domain authorities.
9. Trust levels of attribute keys should be no higher than the trust level of their authorization domain authorities.

5.4.1 Building Blocks: Algorithms

In this subsection, we define all algorithms of Scheme 4 before we formally describe its cryptographic constructions to better understand the scheme. Important notations and their meanings are given in Table 5.2 and assumptions of algorithms are as follows.

• **Scheme4.spg**($\mathbf{k_0}$)$\rightarrow \{\mathbf{MPK}, \mathbf{MSK}, \mathbf{bsk_{Auth_0}}\}$: The central authority runs this algorithm to generate system parameters and secret keys for the top attribute authority $Auth_0$, with the system security parameter k_0 as its input and the system public key set MPK, private key set MSK and the top authority $Auth_0$'s private key base bsk_{Auth_0} as its output.

• **Scheme4.ukg**($\mathbf{MPK}, \mathbf{MSK}, \mathbf{U_k}$)$\rightarrow$ $\mathbf{bsk_k}$: The central authority runs this algorithm to generate user keys, with the system public key set MPK, the system private key set MSK and user identity information U_k as its input and the user U_k's private key base $bsk_k =< D_k, u_k >$ as its output.

• **Scheme4.akg**($\mathbf{MPK}, \Psi$)$\rightarrow \{\mathbf{PK}_{Auth_0}, \mathbf{SK}_{Auth_0}\}$: The top attribute authority $Auth_0$ runs this algorithm to generate its own attribute keys, with the system public key set MPK and attribute set Ψ as the input and its public attribute key set PK_{Auth_0} and private attribute key set SK_{Auth_0} as the output.

• **Scheme4.auth$_i$**($\mathbf{MPK}$, $\mathbf{PK_{Auth_{i-1}}}, \mathbf{SK_{Auth_{i-1}}}$) $\rightarrow \{\mathbf{PK_{Auth_i}}, \mathbf{SK_{Auth_i}}\}$: The domain attribute authority $Auth_{i-1}$ $(1 \leq i \leq N)$ runs this algorithm to authorize domain attribute authority $Auth_i$ with the attribute set Ψ. Its input is MPK, $Auth_{i-1}$'s public key set $PK_{Auth_{i-1}}$ and private key set $SK_{Auth_{i-1}}$. Its output is $Auth_i$'s public key set PK_{Auth_i} and private key set SK_{Auth_i}.

• **Scheme4.uakg**($\mathbf{MPK}, \mathbf{U_k}, \mathbf{bsk_k}, \mathbf{SK_{Auth_i}}, \mathbf{PK_{Auth_i}}$, Ψ_k)$\rightarrow$ $\mathbf{USK_k}$: The domain attribute authority $Auth_i$ runs this algorithm to authorize user U_k with

attribute keys based on the attribute set U_k owned by U_k. Its input is is the system public key set MPK, U_k's private key D_k, $Auth_i$'s public key set PK_{Auth_i}, private key set PK_{Auth_i} and the attribute set Ψ_k owned by U_k. Its output is U_k's private attribute key set USK_k.

• **Scheme4.atg**(Ψ')$\rightarrow \{\Gamma', \mathbf{PTree}_{\Psi'}\}$: The verifier runs this algorithm to generate an attribute tree based on the selected attribute set Ψ'. Its input is attribute set Ψ'. Its output is attribute tree Γ' and the related parameters $PTree_{\Psi'}$.

• **Scheme4.sg**($\mathbf{MPK}, \mathbf{MSK}$, Γ', Ψ', $\mathbf{USK_k}$, $\mathbf{M}$)$\rightarrow \delta$ The signer or user U_k runs this algorithm to generate a signature. Its input is the system public and private key sets MPK and MSK, attribute tree Γ', a selected attribute set Ψ', U_k's private key set USK_k and the message M. Its outputs is signature δ.

• **Scheme4.sv**($\mathbf{M}, \delta$, Ψ')$\rightarrow$ **b**: The verifier runs this algorithm to check whether the signature δ is valid or not. Its input is message M together with its signature δ and the selected attribute set Ψ'. Its output is the checking result $b \in \{0,1\}$.

• **Scheme4.so**($\mathbf{M}, \delta, \Psi', \mathbf{tk}$)$\rightarrow \mathbf{U_k}$: The opener runs this algorithm to retrieve a signer U_k's index from the signature δ. Its input is the message M together with its signature δ, the selected attribute set Ψ' and the opener's tracing key tk. Its output is U_k's index k.

5.4.2 Scheme Construction

The construction of our attribute-related HBA scheme includes two phases: 1) system setup and 2) signature generation, verification and possibly opening. We will use the path from $Auth_N$ to $Auth_0$ illustrated in Fig. 5.4 as an example to demonstrate how to build Scheme 4. Important notation utilized in the construction is included in Table 5.2.

System setup This phase is performed by a central authority (not $Auth_0$) as described in the core ABA structures in Subsection 3.1.1. For simplicity, we can assume that all users are managed by this central authority. System setup includes five steps.

Table 5.2: Notation for Scheme 4

Notations	Description
k_0	The system security parameter.
$\mathbb{Z}_p^*$	The set of integers modulo p, where p is a large prime integer.
G_i	(Multiplicative) cyclic group of prime order p, where $1 \leq i \leq 2$.
e	$e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map.
H	$H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a hash function [69].
$H_K()$	A secure keyed one-way hash function with K as the key.
N	The total number of levels of domain authorities
$Auth_0$	The trusted authority and its level is 0.
$Auth_i$	Domain authority of level i ($1 \leq i \leq N$).
U	$U =< U_1, \cdots, U_{N_u} >$ is the system user set.
N_u	$N_u = \|U\|$, the number of users in U.
Ψ	System attribute set.
Ψ_k	$\Psi_k \subset \Psi$, attribute subset owned by U_k ($1 \leq k \leq N_a$).
Ψ'	$\Psi' \subset \Psi_i$, attribute subset required by the signer.
N_a	$N_a = \|\Psi\|$, the number of attributes in set Ψ.
N_k	$N_k = \|\Psi_k\|$, the number of attributes in set Ψ_k.
MPK	The system public parameter set of the central authority.
MSK	The system private parameter set of the central authority.
PK_{Auth_i}	The public key set of attribute authority $Auth_i$, $0 \leq i \leq N$.
SK_{Auth_i}	The private key set of attribute authority $Auth_i$, $0 \leq i \leq N$.
apk_j	Public attribute key of attribute att_j, where $1 \leq j \leq N_a$.
ask_j	Private attribute key of attribute att_j, where $1 \leq j \leq N_a$.
A_i	The private key of domain authority $Auth_i$.
D_k	The private key of user U_k.
bsk_{Auth_i}	The private key base of attribute authority $Auth_i$ ($0 \leq i \leq N$) and denoted by $bsk_{Auth_i} =< A_i, x_i >$.
bsk_k	The private key base of U_k. It is authorized by domain authority $Auth_i$ and denoted by $bsk_k = \{D_k, u_k\}$.
$T_{i,j}$	The private attribute key of domain authority $Auth_i$ for attribute att_j, authorized by domain attribute authority $Auth_{i,-1}$.
$T_{i,j,k}$	U_k's private attribute key of attribute att_j, authorized by domain authority $Auth_i$.
APK_Ψ	The public attribute key set related to attribute set Ψ.
ASK_Ψ	The private attribute key set related to attribute set Ψ.
USK_k	The private key set of U_k.
Γ'	Attribute tree generated based on attribute set Ψ'.
$\mathbb{L}(\Gamma')$	The leaf node set of attribute tree Γ'.
rt'	Root of attribute tree Γ'.
$ind(x)$	The index of node x.
$par(z)$	The parent node of node z.

1. **System parameter generation (Scheme4.spg)** This algorithm is performed by the central authority. Assume k_0 is the system security parameter. G_1, G_2 are two multiplicative groups of prime order p with g_1 and g_2 as their generators respectively. Let $e : G_1 \times G_1 \to G_2$ be a bilinear group. Assume DeLP is hard to solve in G_1. DLP is hard to solve in both G_1 and G_2. Select $h \in G_1$, ξ_1, $\xi_2 \in \mathbb{Z}_p^*$ and set u, $v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. Select u_0, $\beta_0 \in \mathbb{Z}_p^*$ as the top secret and compute $h_0 = g_1^{\beta_0}$ and $f_0 = g_1^{1/\beta_0}$. The master public key set of the central authority is $MPK =< G_1, G_2, g_1, g_2, h, u, v, g_1^{u_0}, h_0, f_0 >$ and the master private key set of $MSK =< u_0, \beta_0, \xi_1, \xi_2 >$, where the pair of $< \xi_1, \xi_2 >$ is handed to the opener as its tracing key tk.

   Next the central authority randomly chooses $x_0 \in \mathbb{Z}_p^*$ for $Auth_0$ and computes $A_0 = g_1^{(u_0+x_0)/\beta_0}$. The pair $bsk_{Auth_0} =< A_0, x_0 >$ is $Auth_0$'s private key and A_0 should be registered in the opener's database if there is a necessity to trace the identity information of the trusted authority $Auth_0$.

2. **User key generation (Scheme4.ukg)** This algorithm is performed by the central authority. Assume U is the system user group and $N_u = |U|$ is its size. For a user $U_k \in U$ ($1 \le k \le N_u$), the central authority randomly selects $u_k \in \mathbb{Z}_p^*$ and computes $D_k = g_1^{(u_0+u_k)/\beta_0}$. U_k's private key base is the pair $bsk_k =< D_k, u_k >$. D_k should be registered in the opener's database for tracing its identity information later.

3. **Attribute key generation (Scheme4.akg)** This algorithm is performed by the trusted attribute authority $Auth_0$. Assume that attribute set $\Psi = \{att_1, \cdots, att_{N_a}\}$ ($N_a = |\Psi|$) is owned by all authorities on the path in Fig. 5.4, i.e., from $Auth_N$ to $Auth_0$. For attribute $att_j \in \Psi$ ($1 \le j \le N_a$), $Auth_0$ selects a random value $t_{0,j} \in \mathbb{Z}_p^*$ and computes $apk_{0,j} = g_1^{t_{0,j}}$. The system public attribute key published by $Auth_0$ is $PK_{Auth_0} =< e(g_1, g_1^{x_0}), g_1^{t_{0,1}}, \cdots, g_1^{t_{0,N_A}} >$ and the system private attribute key set owned by $Auth_0$ is $SK_{Auth_0} =< t_{0,1}, \cdots, t_{0,N_A} >$.

4. **Domain authority authorization (Scheme4.auth$_i$)** The authorization of Ψ to $Auth_i$ ($1 \le i \le N$) is performed by domain authority $Auth_{i-1}$. Assume $T_{0,j} = g_1^{(u_0+x_0)/\beta_0} H(att_j)^{t_{0,j}}$ is the private attribute key owned by $Auth_0$ related to attribute att_j. To generate the private key for domain authority $Auth_i$, $Auth_{i-1}$ randomly selects $x_i \in \mathbb{Z}_p^*$ and computes $w_i = g_1^{x_i}$ as well as $A_i = A_{i-1} f_0^{x_i} = g_1^{(u_0+x_0+\cdots+x_i)/\beta_0}$. The private key for $Auth_i$ is $bsk_{Auth_i} =< A_i, x_i >$

and A_i should be registered in the opener's database. For each $att_j \in \Psi$ owned by $Auth_i$, $Auth_{i-1}$ randomly chooses $t_{i,j}$ and computes

$$T_{i,j} = T_{i-1,j} f_0^{x_i} H(att_j)^{t_{i,j}} = g_1^{(u_0+x_0+\cdots+x_i)/\beta_0} H(att_j)^{t_{0,j}+\cdots+t_{i,j}}.$$

The public attribute key of $Auth_i$ related to att_j is $apk_{i,j} = g_1^{t_{i,j}}$. The public key set published by $Auth_i$ is $PK_{Auth_i} =< e(g_1, x_i), g_1^{t_{i,1}}, \cdots, g_1^{t_{i,N_a}} >$ and the private key set owned by $Auth_i$ is $SK_{Auth_i} =< t_{i,1}, \cdots, t_{i,N_a} >$.

5. **User attribute key generation (Scheme4.uakg)** Suppose $U_k \in U$ $(1 \leq k \leq N)$ wants to obtain attribute keys related to Ψ from $Auth_i$. U_k interacts with the domain authority $Auth_i$ and generates U_k's private attribute key related to att_j as $T_{i,j,k} = g_1^{u_k} T_{i,j} = g_1^{(u_0+u_i+x_0+\cdots+x_i)/\beta_0} H(att_j)^{t_{0,j}+\cdots+t_{i,j}}$, where $t_{i,j} \in \mathbb{Z}_p^*$ is randomly selected by the domain authority $Auth_i$.

Signature generation and verification Suppose a verifier requires that the attributes held by the signer should be authorized by $Auth_i$ and the attribute set demanded is Ψ'. The signature generation and verification proceeds as follows.

1. **Verifier: attribute tree generation (Scheme4.atg)** Assume the required attribute set selected by the verifier is Ψ'. The verifier randomly selects a secret $\alpha \in \mathbb{Z}_p^*$ and constructs an attribute tree Γ' with rt' as its root. Next the verifier computes C_y and C'_y in the same way as described in Subsection 5.3.2. Meanwhile, the verifier computes

$$K_v = (e(f_0, g_1^{x_0}) \cdots e(f_0, g_1^{x_i}))^{\alpha} = e(g_1, g_1)^{(x_0+\cdots+x_i)\alpha/\beta_0}.$$

Then the verifier sends the message $\{\Gamma', g_1^{\alpha}, \forall y \in Leaf(\Gamma') : C_y, C'_y\}$ to the signer.

2. **Signer: signature generation (Scheme4.sg)** Assume the attribute set owned by U_k is denoted by Ψ_k ($N_k = |\Psi_k|$) and it satisfies the relation $\Psi \supset \Psi_k \supset \Psi'$. Therefore, there should exist an attribute $att_j \in \Psi_k$ $(1 \leq j \leq N_k)$ corresponding to every leaf node in attribute tree Γ', so we will use y to replace the symbol att_j in the following for simplicity. Once the signer receives message $\{\Gamma', g_1^{\alpha}, \forall y \in Leaf(\Gamma') : C_y, C'_y\}$ from the signer, it computes

$$DecryptNode(T_{i,j,k}, C_y, C'_y, y)$$

$$= \frac{e(T_{i,j,k}, C_y)}{e(Apk_{0,j} \cdots Apk_{i,j}, C'_y)}$$
$$= e(g_1, g_1)^{(u_0+u_k+x_0+\cdots+x_i)q_y(0)/\beta_0}.$$

The verifier's goal in the next step is to compute compute $F_{rt'}$ by calling algorithm $DecryptNode(T_{i,j,k}, C_y, C'_y, y)$ for the root node rt'. The process is the same as algorithm Scheme3.sg described in Subsection 5.3.2. Then it singer computes K_s by

$$K_s = F_{rt'}/e(f_0^{u_k}, g_1^{\alpha}) = e(g_1, g_1)^{(x_0+\cdots+x_i)\alpha/\beta_0}.$$

From the above equation, we can see that $K_s = K_v$ holds. Next the signer selects $\zeta, \alpha, \beta, r_\zeta, r_\alpha, r_\beta, r_k, r_{\delta_1}, r_{\delta_2} \in \mathbb{Z}_p^*$ randomly and calculates

$$\begin{aligned}
&C_1 = u^{\zeta}, C_2 = v^{\beta}, C_3 = D_k h^{\zeta+\beta},\\
&\delta_1 = u_k\zeta, \delta_2 = u_k\beta,\\
&R_1 = u^{r_\zeta}, R_2 = v^{r_\beta}, R_4 = C_1^{r_k} u^{-r_{\delta_1}}, R_5 = C_2^{r_k} v^{-r_{\delta_2}},\\
&R_3 = e(C_3, g_1)^{r_k} e(h, w_i)^{-r_\zeta - r_\beta} e(h, g_1)^{-r_{\delta_1} - r_{\delta_2}},\\
&c = H_{K_s}(M, C_1, C_2, C_3, R_1, R_2, R_3, R_4, R_5) \in \mathbb{Z}_p^*,\\
&s_\zeta = r_\zeta + c\zeta, s_\beta = r_\beta + c\beta, s_\alpha = r_\alpha + c\alpha,\\
&s_k = r_k + cu_k, s_{\delta_1} = r_{\delta_1} + c\delta_1, s_{\delta_2} = r_{\delta_2} + c\delta_2.
\end{aligned}$$

Then the signer sends the signature $\sigma =< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$ to the verifier. From the way how the signature σ is generated, we can see that it is almost the same as the the signature generation phase described in Subsection 5.3.2 for Scheme 3. The only difference is that here C_3 is generated based on D_k instead of $A_{i,k}$ in Scheme 3.

3. **Verifier: signature verification (Scheme4.sv)** The verifier first computes R'_1, R'_2, R'_3, R'_4 and R'_5 the same as described in Subsection 5.3.2 for Scheme 3. The next step is to compute $c' = H_{K_v}(M, C_1, C_2, C_3, R'_1, R'_2, R'_3, R'_4, R'_5)$ and checks whether c' equals to c that it has received. If so, the signer is successfully authenticated and otherwise the authentication fails.

Signature Opening This algorithm is carried out by the opener. If there is a necessity to reveal the identity information of a signature δ generated by U_k based on $\{M, \Psi'\}$, the opener computes $D_k = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$. Then the opener compares D_k with items in its database. If there exists an item matching it, the opener returns k

as the index of user U_k.

5.4.3 Correctness Analysis

In this subsection we provide proof of correctness of Scheme 4.

Theorem 10. ***(Correctness)*** *The construction of Scheme 4 proposed in Subsection 5.3.2 is correct, that is, the following properties are satisfied:*

1. K_v=K_s,
2. $R_i = R'_i$, *where* $1 \leq i \leq 5$, *and*
3. $D_k = C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2})$ *holds.*

Proof. 1) From the detailed description of algorithm Scheme4.atg in the phase of signature generation and verification in Subsection 5.4.2, we can see that since the signer possesses the attribute set $\Psi_k \supset \Psi'$, it can compute $K_v = e(g_1,g_1)^{(x_0+\cdots+x_i)\alpha/\beta_0}$ which equals to K_s.
2) R'_i $(1 \leq i \leq 5)$ can be computed as follows:

$$
\begin{aligned}
R'_1 &= u^{s_\zeta} C_1^{-c} = u^{r\zeta + c\zeta}(u^\zeta)^{-c} = R_1 \\
R'_2 &= v^{s_\beta} C_2^{-c} = v^{r_\beta + c\beta}(v^\beta)^{-c} = R_2 \\
R'_3 &= e(C_3,g_1)^{s_k} e(h,w_i)^{-s_\alpha - s_\beta} e(h,g_1)^{-s_{\delta_1} - s_{\delta_2}} \left(\frac{e(C_3,w_i)}{e(g_1,g_1)}\right)^c \\
&= e(C_3,g_1)^{r_k + cu_k} e(h,w_i)^{-(r_\alpha + c\alpha) - (r_\beta + c\beta)} e(h,g_1)^{-(r_{\delta_1} + c\delta_1) - (r_{\delta_2} + c\delta_2)} \left(\frac{e(C_3,w_i)}{e(g_1,g_1)}\right)^c \\
&= R_3 \left(e(C_3,g_1)^{u_k} e(h,w_i)^{-(\alpha+\beta)} e(h,g_1)^{-(\delta_1+\delta_2)} \frac{e(C_3,w_i)}{e(g_1,g_1)} \right)^c \\
&= R_3 \left(e(C_3 h^{-(\alpha+\beta)}, w_i g_1^{u_k}) e(C_3,w_i)^{-1} \frac{e(C_3,w_i)}{e(g_1,g_1)} \right)^c \\
&= R_3 \left(e(A_i, w_i g_1^{u_k})(g_1,g_1)^{-1} \right)^c = R_3 \\
R'_4 &= u^{-s_{\delta_1}} C_1^{s_k} = u^{-r_{\delta_1} - c\delta_1}(u^\zeta)^{r_k + cu_k} \\
&= u^{-r_{\delta_1} - c\zeta u_k}(u^\zeta)^{r_k + cu_k} = C_1^{r_k} u^{-r_{\delta_1}} = R_4
\end{aligned}
$$

3)$C_3/(C_1^{\varepsilon_1} C_2^{\varepsilon_2}) = D_k h^{\zeta+\beta}/((u^\zeta)^{\varepsilon_1}(v^\beta)^{\varepsilon_2}) = D_k$. □

5.4.4 Adversary Model

In this subsection, we describe the adversary model based on random oracle and game theory. Assume that the adversary (denoted by A) has polynomially bounded computational abilities. In this model, there are several different oracles that the adversary can query and then based on these queries, the adversary will take a challenge. If the adversary A succeeds in the challenge, it wins the game. Otherwise, A fails the game. We will use **Scheme4.** as prefix and **.Oracle** as suffix to represent these oracles. The details of these oracles are as follows.

• **Scheme4.Syspara.Oracle**: Adversary A can query this oracle about system parameters generated and the top attribute authority $Auth_0$'s private key base.

• **Scheme4.Userkey.Oracle**: Adversary A can query this oracle about user U_k's secret key base by sending the user's index k and the oracle will reply with bsk_k.

• **Scheme4.Attkey.Oracle**: Adversary A can query this oracle about attribute private keys related to attribute authority $Auth_i$ $(1 \leq i \leq N)$ by sending the attribute att_j and the index i of the attribute authority. The oracle will reply with $T_{i,j}$.

• **Scheme4.UserAttkey.Oracle**: Adversary A can query this oracle about user U_k's private attribute keys that are authorized by domain attribute authority $Auth_i$. To query, A needs to send the attribute att_j, the user's index k and the domain authority's index i. The oracle will reply with $T_{i,j,k}$.

• **Scheme4.Signature.Oracle**: Adversary A can query this oracle to obtain user U_k's signature by sending the user's index k, message M and the selected attribute set Ψ'. The oracle will reply with a signature δ that is generated based on $\{M, k, \Psi'\}$.

• **Scheme4.Opening.Oracle**: Adversary A can query this oracle about a signer U_k's index of signature δ. A needs to send $\{M, \Psi', \delta\}$ to the oracle, where M is the message to to signed, Ψ' is the selected attribute set and δ is the signature generated based on M and Ψ'. The oracle will reply with the signer's index k.

Next we describe the game **Scheme4.Game.Anonymity** as follows.

1. **System setup**: Run algorithms Scheme4.spg, Scheme4.ukg, Scheme4.akg, Scheme4. auth_i, Scheme4.uakg to generate the system public and private key sets MPK and MSK, the top attribute authority $Auth_0$'s private key based bsk_{Auth_0}, domain attribute authority $Auth_i$'s private key base bsk_{Auth_i}, domain attribute authority $Auth_i$'s public and private key sets PK_{Auth_i} and SK_{Auth_i} and users private attribute key. The adversary A can obtain all public parameters after the system initiation is completed.

2. **Phase 1**: The adversary A can query all oracles described above in this phase, but A cannot query the tracing key tk. Otherwise, A can trace the identity information of every signature, which contracts to anonymity.

3. **Challenge**: After certain rounds of queries in phase 1, the adversary A decides to take the challenge in this phase. A randomly selects two indexes i_b ($b \in \{0,1\}$), a message M and an attribute set Ψ' and sends $\{i_b, M, \Psi'\}$ to the system. A will be replied with a signature δ_b ($b \in \{0,1\}$), which is computed based on $\{i_b, M, \Psi'\}$ ($b \in \{0,1\}$). This challenge should have not been queried in phase 1.

4. **Phase 2**: In this phase, the adversary A behaves the same as in phase 1 but A is not allowed to query about the challenge.

5. **Guess**: After certain rounds of queries in phase 2, the adversary A output its guess $b' \in \{0,1\}$. A wins the game only if $b = b'$ holds.

Definition 15. ***(Anonymity of Scheme 4)*** *If an adversary A with polynomially bounded computational abilities can win the game* ***Scheme4.Game.Anonymity*** *with non-negligible probability, we say that Scheme 4 is not anonymous. Otherwise, Scheme 4 is anonymous.*

The game **Scheme4.Game.Traceability** proceeds as follows.

1. **System setup**: Run algorithms Scheme4.spg, Scheme4.ukg, Scheme4.akg, Scheme4.auth_i, Scheme4.uakg to generate system parameters as described in the game "Scheme4.Game.Anonymity". The adversary has access to all public parameters as tracing key tk.

2. **Phase 1**: The adversary A can query all oracles described above.

3. **Challenge**: The adversary decides to challenge now. It chooses the attribute requirement set Ψ', message M to be singed and the index k of user U_k whom it plans to impersonate.

4. **Phase 2**: The adversary A continues to queries oracles in this phase as described in phase 1, but still A is not allowed to query about the challenge.

5. **Output**: The adversary A forges a signature δ based on $\{k, M, \Psi'\}$. The opener first checks the validity of A's challenge δ. If it is invalid, it aborts the game and output 0 means that the adversary A fails. Otherwise, the opener continues to reveal the signer's index k. Next the opener searches its database to check whether there is any item matching index k. If so, the opener outputs 0 and A fails the game. Otherwise, the opener outputs 1, which means that A has successfully impersonated an unregistered signer to forge a valid signature.

Definition 16. ***(Traceability of Scheme 4)*** *If an adversary A with polynomially bounded computational abilities can win the game* ***Scheme4.Game.Traceability*** *with non-negligible probability, we say that Scheme 4 is not traceable. Otherwise, Scheme 4 is traceable.*

5.4.5 Security Analysis

From the signature generation and verification procedures of Scheme 3 and Scheme 4, we can see that they share a lot in common, only with two minor differences. The first difference is algorithms $DecryptNode(T_{k,j}, C_y, C'_y, y)$ and $DecryptNode(T_{i,j,k}, C_y, C'_y, y)$, where the first input parameters are $T_{k,j}$ and $T_{i,j,k}$ in these two algorithms respectively. In algorithm $DecryptNode$ in Scheme 3, the first parameter $T_{k,j}$ is related to both the signer U_k and the attribute att_j. The algorithm $DecryptNode$ in Scheme 4, the first parameter $T_{i,j,k}$ is related to the attribute authorized domain authority $Auth_i$, the attribute att_j and the signer U_k. Even though the ways how $T_{k,j}$ and $T_{i,j,k}$ are generated are different, they have similar representations and are both the signer's private attribute key based on attribute att_j. The second difference is the representation of C_3, which are denoted by $A_{i,k}h^{\zeta+\beta}$ and $D_k h^{\zeta+\beta}$ respectively. $A_{i,k}$ means that the signer U_k's private key is generated by domain authority $Auth_i$, while D_k means that it is generated by a central authority, so both $A_{i,k}$ and D_k have the same meaning and represent U_k's private keys.

Based on this, we can draw two conclusions. The first conclusion is that Scheme 4 provides both anonymity and traceability as stated in Definitions 15 and 16. The second conclusion is that the way how to prove both anonymity and traceability of Scheme 4 is the same as the proof of Scheme 3, with the exception that $DecryptNode(T_{k,j}, C_y, C'_y, y)$ and $A_{i,k}$ should be replace with $DecryptNode(T_{i,j,k}, C_y, C'_y, y)$ and D_k respectively. As a result, we will only state the formal descriptions of anonymity and traceability of Scheme 4 as Theorems 11 and 12 in the following without repeating the proof here.

Theorem 11. ***(Anonymity)*** *Scheme 4 is anonymous if DLE is IND-CPA secure. More specifically, given D_k with related signatures $\sigma_b =< M, C_1, C_2, C_{3(b)}, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$ with a random toss $b \in \{0,1\}$, if an adversary A with polynomially bounded computation ability has a non-negligible advantage to guess the correct b, we say that the Scheme 4 is not anonymous. Otherwise, the scheme is anonymous.*

Theorem 12. ***(Traceability)*** *Assume the keyed one-way hash function H is collision resistant. Scheme 4 is fully traceable if 1) given a valid signature $\sigma =< M, C_1, C_2, C_3, c, s_\zeta, s_\beta, s_\alpha, s_k, s_{\delta_1}, s_{\delta_2} >$, the opener can trace the identity of the signer and 2) an adversary A with polynomially bounded computation ability cannot forge a valid signature σ' that can pass the verifier's verification.*

5.5 Conclusions

In this chapter we have studied construction of hierarchical ABA schemes. As mentioned in Section 5.1, there have only been some work dealing with hierarchies in ABE schemes. However, to our best knowledge, there are no results related to HABA schemes. In this chapter, we have discussed two types of hierarchy in ABA schemes: user-related and attribute-related. Scheme 3 is an example of user-related HABA schemes. Users are managed by different hierarchical domain authorities, while all their attributes are authorized by the trusted attribute authority. As a result, fine-grained ABA can be implemented by requiring the authorizing domain authority in addition to the attribute requirements, so only users managed by a specific domain authority can be authenticated. Scheme 4 is an example of attribute-related HABA schemes. It is different from Scheme 3, where all users are managed and authorized by the same central authority, but their attributes are authorized by different domain authorities. In Scheme 4, attribute authorities are divided into different

levels with different privileges. Users can obtain different attribute keys from different domain attribute domain authorities, but the attribute keys will be bound with different privileges. Therefore, user authentication can be carried out by checking not only attributes but also the privilege levels of attributes.

Chapter 6

ABA Schemes without Traceability

All ABA schemes discussed in Chapter 4 and Chapter 5 are traceable. Traceability is a very useful property in a system where there is a need to resolve disputes by revealing signers' identity information, for examples, as legal evidence. However, as described in the structure of core ABA schemes in Fig. 3.1, traceability is not a mandatory security requirement for ABA schemes. Even though all schemes we have proposed till now can be adopted to systems that do not need traceability by not installing an opener, the implementation of traceable ABA schemes will still be more expensive than untraceable ones. There are three main reasons. First of all, the cryptographic construction of traceable ABA schemes is definitely more complicated than untraceable ones, which will need more computations and storage space. Secondly, if we take a closer look, we can find that parameters C_1, C_2 and C_3 in all schemes are used for traceability. Moreover, we may need more parameters to guarantee the property that an adversary cannot forge a valid signature on behalf of an unregistered user. The third reason relates to security. Intuitively, the more complicated a scheme is, the more difficult to guarantee its security requirements.

For the above reasons, we will use this whole chapter to discuss untraceable ABA schemes and how they can be cryptographically constructed based on traceable ABA schemes. To make the situation simpler and the untraceable ABA schemes easier to understand, we will modify Scheme 1 and Scheme 2 proposed in Chapter 4 to build two untraceable ABA schemes. The work we plan to do in this chapter is not to provide a general framework how to design and construct untraceable ABA schemes, because it demands more and deeper research and it can be considered as future research directions. We will use two specific examples to show the differences in cryptographic construction between traceable and untrace-

able ABA schemes, and how to achieve untraceable ABA schemes from traceable ones.

This chapter is organized as follows. First of all, we will review and compare related work on untraceable ABA schemes in Section 6.1. Next we describe in details how to construct untraceable ABA schemes based on Scheme 1 and Scheme 2. At last, we conclude the results of this chapter in Section 6.4.

6.1 Related Work

As explained in Section 4.1, researchers have already conducted some research on both traceable and untraceable ABA schemes [11, 103–108] as well as how to construct ABA schemes from attribute-based schemes [13]. To our best knowledge, however, there have not been any publications discussing the way how to transform traceable ABA schemes into untraceable ones or vise versa. Since we have already reviewed traceable ABA schemes in Section 4.1, we will only survey recent work on untraceable ABA schemes in this chapter.

As far as we know, the earliest untraceable ABA protocol was proposed in [46]. The protocol is based on bilinear groups [109–114] and a shared secret key will be established at the end of the protocol if the party to be authenticated possesses a set of attributes required by the other party. Its main focus is key sharing rather than authentication. Another ABA and key sharing protocol was proposed later in [115] with traceability. Cao etc. proposed an untraceable authentication scheme with attributes for online social networks in [47]. To protect users' privacy in eHealth network, Guo etc. [116] built a privacy-preserving ABA scheme without traceability. The authors divided privacy into four different levels to satisfy different privacy requirements in different scenarios, and different solutions were provided for all four privacy levels. The scheme proposed in [105] is an improved version of the scheme proposed in [116], aiming at adjusting to the adoption of mobiles in eHealth network. A proxy-based ABA scheme was proposed in [117] for the situation when the original signer was not available to sign a specific document and the scheme was untraceable as well.

A fuzzy ABA scheme was built based on an ABE scheme in [48]. Authors in [48] first described a fuzzy ABE scheme, based on which a fuzzy ABA scheme was built. Different from other attribute-based schemes, the scheme used Lagrange polynomial interpolation [118–120] instead of attribute trees to represent the possessing of a set of attributes. Therefore, the scheme can only present the relation

"possessing" rather than logical "AND" or "OR" among possessed attributes. The most significant difference between [105] and the untraceable ABA schemes to be proposed in this chapter is the authenticator. The authenticator in [105] can be considered as the central authority in the core ABA schemes described in Fig. 3.1 and knows the top secret, but authenticators in our schemes are normal users who only know the public parameters. This property can make our schemes more widely applicable.

6.2 Case Study 1: An Untraceable ABA Scheme Based on Scheme 1

In this section, we will build an untraceable ABA scheme (Scheme 5) based on Scheme 1 by removing the part dealing with traceability. Scheme 5 consists of algorithms Scheme5.spg, Scheme5.ukg, Scheme5.akg, Scheme5.uakg, Scheme5.ars, Scheme5.sg and Scheme5.so. Compared with the algorithms included in Scheme 1, the only difference is that Scheme 5 does not have the signature opening algorithm which is denoted by Scheme1.so. All algorithms of Scheme 5 can be defined in the same way as in Scheme 1, so we will omit this part. For more details, refer to Subsection 4.4.2.

6.2.1 Scheme Construction

Since Scheme 5 is built based on Scheme 1, they share a lot in common. The most significant difference between these two schemes lies in the signature generation and verification phase. Scheme 5 consists of system setup as well as signature generation and verification. We will use the structure and workflow of the core ABA schemes in Fig. 3.1 to describe Scheme 5. In the phase of system setup, the system is initiated by generating system parameters, users' keys and attribute keys, and it proceeds as follows.

1. **System parameter generation (Scheme5.spg)** This algorithm is carried out by the central authority. Assume k_0 is the security parameter. G_1, G_2 and G_3 are multiplicative groups of prime order p, $e : G_1 \times G_2 \to G_3$ is a bilinear map and φ is a computable isomorphism between G_1 and G_2. $H : \{0,1\}^* \to \mathbb{Z}_p^*$ is a one-way hash function. First of all, the central authority randomly selects a generator $g_2 \in G_2$ and $\gamma \in \mathbb{Z}_p^*$. Let $g_1 = \varphi(g_2)$ and $w = g_2^{\gamma}$. The system

private parameter set is $MSK =< \gamma >$ and the system public parameter set is $MPK =< G_1, G_2, G_3, e, H, g_1, g_2, w >$.

2. **User key generation (Scheme5.ukg)** This algorithm is carried out by the central authority. It takes γ as the input and outputs user U_i's secret key base $bsk_i =< A_i, x_i >$. Since it proceeds exactly as Scheme1.ukg, so refer to Subsection 4.4.3 for more details.

3. **Attribute key generation (Scheme5.akg)** This algorithm is carried out by the attribute authority. For each attribute $att_j \in \Psi$ $(1 \leq j \leq N_a, N_a = |\Psi|)$, the attribute authority randomly chooses a number $t_j \in \mathbb{Z}_p^*$ as its private key ask_j and computes its public key as $apk_j = g_1^{t_j}$. Let $w_j = w^{t_j}$. The public and private key sets related to Ψ can be denoted by $APK_\Psi =< w_1, \cdots, w_{N_a}, apk_1, \cdots, apk_{N_a} >$ and $ASK_\Psi =< t_1, \cdots, t_{N_a} >$ respectively.

4. **User attribute key generation (Scheme5.uakg)** This algorithm is carried out by the attribute authority. It proceeds the same as the algorithm Scheme1.uakg (refer to Subsection 4.4.3 for more details).

5. **Attribute tree generation (Scheme5.atg)** This algorithm proceeds the same as Scheme1.atg (refer to Subsection 4.4.3 for more details).

Signature generation and verification This is the second phase of Scheme 5 and it proceeds as follows.

1. **Verifier: attribute requirement selection (Scheme5.ars)** This algorithm proceeds the same as Scheme1.ars (refer to Subsection 4.4.3 for more details).

2. **Signer: signature generation (Scheme5.sg)** If the signer owns the required attributes selected by the verifier in algorithm Scheme5.ars, this algorithm proceeds as follows. Assume the attribute set to be used by the signer is Ψ' $(N_a' = |\Psi'|)$. First of all, the signer runs the algorithm $Simplify\_CTree$ as described in Subsection 4.2.2 to obtain the simplified attribute tree Γ'. Assume the leaf node set of Γ' is $\mathbb{L}' = \mathbb{L}'^{Att} + \mathbb{L}'^{Dum}$, where $\mathbb{L}'^{Att}$ is the attribute leaf set and $\mathbb{L}'^{Dum}$ is the dummy leaf set. Based on Γ', U_i computes $\triangle_j$ $(att_j \in \mathbb{L}'^{Att})$, $\triangle_{d_j}$ $(d_j \in \mathbb{L}'^{Dum})$, its root value rt' as described in Subsection 4.2.2 and $g_d = \prod_{d_k \in \mathbb{L}'^{Dum}} (g_2^{d_k})^{\triangle_{d_k}}$. Next U_i randomly chooses $\alpha, \varepsilon, r_\alpha, r_\varepsilon \in \mathbb{Z}_p^*$

and calculates

$$C = A_i w^{\varepsilon}, CT_j = T_{i,j} w_j^{\alpha}, R_{Att} = \frac{e(\prod_{att_j \in \Psi'} w_j^{\triangle_j}, g_2)^{r\alpha}}{e(w, rt'/g_d)^{r\varepsilon}},$$
$$c = H(M, C, R_{Att}) \in \mathbb{Z}_p^*, s_\alpha = r_\alpha + c\alpha, s_\varepsilon = r_\varepsilon + c\varepsilon.$$

Then U_i sends the signature $\delta =< M, c, C, CT_1, \cdots, CT_{N'_a}, R_{Att}, s_\alpha, s_\varepsilon, \Psi' >$ to the signer.

3. **Signature verification (Scheme5.sv)** After receiving the signature δ from the signer, the verifier runs the algorithm *Simplify_CTree*, obtains the related attribute tree Γ' and computes the root value rt', $\triangle_j$ ($att_j \in \mathbb{L}'^{Att}$), $\triangle_{d_k}$ ($d_k \in \mathbb{L}'^{Dum}$) and $g_d = \prod_{d_k \in \mathbb{L}'^{Dum}} (g_2^{d_k})^{\triangle_{d_k}}$ as the verifier does in algorithm Scheme5.sg. Next the verifier computes

$$R'_{Att} = \frac{e(\prod_{att_j \in \Psi'} h_j^{\triangle_j}, g_2)^{s_\alpha}}{e(w, rt'/g_d)^{s_\varepsilon}} \left(\frac{e(C_4, rt'/g_d)}{e(\prod_{att_j \in \Psi'} CT_j^{\triangle_j}, g_2)} \right)^c.$$

Then the verifier calculates $c' = H(M, C, R'_{Att})$ and checks whether c' equals to c. If so, the verifier believes that the signer owns the required attributes and the authentication succeeds.

The correctness of the scheme holds if $R'_{Att} = R_{Att}$, which has already been proved in Theorem 1 in Subsection 4.4.4.

6.3 Case Study 2: An Untraceable ABA Scheme Based on Scheme 2

In this section, we propose another untraceable ABA scheme (Scheme 6) based on Scheme 2 proposed in Section 4.5. The new scheme keeps the property of one-time attribute tree generation but removes the property of traceability. Scheme 6 is similar to Scheme 2, but it is less complicated and its signature is shorter. Moreover, Scheme 6 consists of all algorithms in Scheme 2 except for the signature opening algorithm Scheme2.so, and these descriptions remain the same. Therefore, we will skip the algorithm descriptions and move directly to the descriptions of the cryptographic construction in the following subsection.

6.3.1 Scheme Construction

Since Scheme 6 is built based on Scheme 2, they have many common stages in cryptographic construction in the system setup phase but differ in the phase of signature generation and verification. The system setup phase includes four algorithms which are described as follows.

1. **System parameter generation (Scheme6.spg)** This algorithm is carried out by the central authority. Assume k_0 is the security parameter, G_1 and G_2 are two multiplicative groups of prime order p and with g_1 and g_2 as their generators. $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear group. DeLP and q-SDH is hard to solve in G_1 and DLP is hard to solve in both G_1 and G_2. $H_K : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ is a secure keyed one-way hash function where K is the key. Select $x_0 \in \mathbb{Z}_p^*$ as the top secret and compute $w_0 = g^{x_0}$. The system public key is $MPK =< G_1, G_2, g_1, g_2, w_0 >$ and the system private key is $MSK =< x_0 >$.

2. **Attribute key generation (Scheme6.akg)** This algorithm is the same as Scheme2.akg.

3. **User key generation (Scheme6.ukg)** This algorithm is the same as Scheme2.ukg.

4. **User attribute key generation (Scheme6.uakg)** This algorithm is the same as Scheme2.uakg.

Signature generation and verification This is the second phase of Scheme 6 and it proceeds as follows.

1. **Verifier: attribute tree generation (Scheme6.atg)** Assume the attribute set the verifier requires is Ψ' and the related attribute tree is Γ'. This algorithm runs the same as Scheme2.atg.

2. **Signer: signature generation (Scheme6.sg)** This algorithm proceeds the same as Scheme2.sg until the computation of K_s. Next the signer computes $c = H_{K_s}(M)$ and $\sigma =< M, c >$ and then sends σ to the verifier.

3. **Verifier: signature verification (Scheme2.sv)** After receiving σ, the verifier calculates $c' = H_{K_v}(M)$ and checks whether c' equals to c. If so, the verifier believes that the signer owns the required attributes and accepts the signature, otherwise rejects it.

The same as in Scheme 2, the verification of the signer's attributes is realized by the way how K_s is computed and the security of the keyed one-way hash function H_K in the above algorithm. To successfully generate a valid signature σ, an adversary has two choices, i.e., to generate the hash value c out of a different message M' without the knowledge of K_s, or compute K_s and then the MAC [121–123] value c. For the first choice, given the security of the keyed one-way hash function H_K, it is impossible to compute the correct value c without knowing the shared secret key K_s. For the second choice, since DLP is hard to solve in G_1, the adversary cannot compute α from g_1^{α}, and therefore it cannot compute K_v via $e(g_1, g_1^{x_0})^{\alpha}$. The only way left is to behave like the signer does. However, from the description of the algorithm Scheme6.sg or Scheme2.sg (described in Subsection 4.5.3), we can see that without the required attribute keys, it is impossible to recover K_s either. Therefore, it is impossible for an adversary to generate c or the same signature σ.

6.4 Conclusions

In this chapter, we have described how to build untraceable ABA schemes based on two traceable ABA schemes, i.e., Scheme 1 and Scheme 2 proposed in Chapter 4. The main idea is to remove the part responsible for traceability in the signature together with other supporting parameters. If we compare the cryptographic constructions described in Chapter 4 with those in this chapter, we can see that the parameters for traceability in Scheme 1 and Scheme 2 are C_1, C_2, C_3 and R_i ($1 \leq i \leq 5$), where R_i are used to ensure the unforgeability of the signature σ. In order to achieve untraceable attribute-based schemes, the most important part here is the recovery of R_{Att} based on $CT_{i,j}$ or the computation of K_s. The reason is that without the knowledge of required attributes, it is impossible for a user to compute either R_{Att} or K_s.

Chapter 7

Discussion, Conclusions and Future Work

This chapter first summarizes research contributions of this dissertation in Section 7.1. Next in Section 7.2, future work will be discussed, including attribute tree building, general framework of constructing ABA schemes, hierarchy, security models, traceability and revocation. The work in this subsection is based on [23].

7.1 Conclusions

This dissertation addresses several security and privacy issues in ABA schemes. As we have summarized in related work in Subsections 4.1, 5.1 and 6.1, there have already been many published research results about ABA schemes. Based on these results and our own, we first categorize ABA schemes according to two criteria such that we can gain a general understanding about ABA schemes and their properties. The first criterion is properties of ABA schemes. According to this criterion, we have divided ABA schemes into core ABA schemes, traceable and untraceable ABA schemes, dynamic and static ABA schemes and hierarchical ABA schemes. The core ABA schemes only consist of main parts and functions, while the others are extended versions of core ABA schemes by adding specific properties. The second criterion is the relation between attribute keys and attribute requirements. If the generation of attribute keys is bound to a specific attribute structure, the ABA schemes are considered as KP-ABA. However, if the generation of attribute keys is only based on a set of attributes and independent from attribute structures, these ABA schemes are CP-ABA. All ABA schemes proposed in this dissertation are CP-

ABA. In this dissertation, we have proposed six different ABA schemes according to the first categorization, and the details how they are constructed are included in Chapter 4, Chapter 5 and Chapter 6.

Scheme 1 proposed in Chapter 4 is the first example that demonstrates how to construct a CP-ABA scheme, and its dynamic of attribute tree generation is realized by the introducing of a big central attribute tree. More specifically, Scheme 1 utilizes a down-to-top approach to build attribute trees. A big central attribute tree is first built, based on which different attribute subtrees can be obtained by simplifying the big central attribute tree. In this way, the same attribute keys can be used for scenarios where different attributes are required without regenerating attribute keys as in KP-ABA schemes [13]. Therefore, this scheme can provide more flexibility and save system resources for regenerating and storing different attribute keys, so it can be implemented in a dynamic environment. However, the application of the big central attribute tree still costs a lot of computation and storage resources. In addition, this approach of attribute tree building has a limitation. Since all attribute subtrees are obtained by simplifying the central attribute tree, the logical relations between these attributes are restricted by the relations of attributes in the central attribute tree. As a result, in order to get an attribute tree where attributes are of a totally different relation, it is still unavoidable to generate new attribute trees and attribute keys.

These two issues are solved in Scheme 1, Scheme 2 by introducing a one-time attribute tree combined with a different approach of cryptographic construction for the signature generation and verification. Another innovation of Scheme 2 is that it reduces the signature size. In Scheme 1, parameters related to required attributes are part of the signature. However, these parameters are replaced by the usage a shared key and a secure keyed one-way hash function in Scheme 2, where this shared key is computed based on the required attributes.

The main focus of both Scheme 1 and Scheme 2 is the dynamic and flexibility of attribute trees and the property of traceability. In Scheme 3 and Scheme 4 proposed in Chapter 5, we introduced hierarchies. We have discussed two hierarchical situations, user-related and attribute-related. Scheme 3 is a U-HABA scheme. In this scheme, all attributes are organized by an attribute authority, while users are managed by different security levels of domain authorities. In this example, there are authorities from level 0 to level N, with the privilege from the highest to the lowest accordingly. There is only one authority of level 0, named $Auth_0$, and the remaining authorities are domain authorities. Domain authorities of a lower privilege

level are authorized by authorities of a higher level and users are authorized by different domain authorities. By applying this type of hierarchy, attributes are utilized together with users' privilege for authentication. In addition, this mechanism has also brought in the chance of specifying which group users should belong to, and thus it increases the flexibility of the scheme.

Scheme 4 is an A-HABA scheme, different from Scheme 3. All users in Scheme 4 are organized by a central authority while attributes are managed by domain authorities of different privileges. In this case, attribute authorities are divided into $N+1$ levels, from level 0 to level N with the highest and the lowest privileges. The authority of the highest privilege is denoted by $Auth_0$ and the rest are domain authorities. Domain authorities of a higher level can authorize domain authorities of a lower level as well as users' attributes, but $Auth_0$ can only authorize domain authorities of level 1. By implementing this type of attribute-related hierarchy, the authentication requirements can also consist of the privileges of attributes instead of merely attributes.

All ABA schemes from 1 to 4 discussed above are traceable. In order to gain some knowledge about how to construct untraceable ABA schemes and the relation between traceable and untraceable schemes, we have constructed two untraceable ABA schemes based on Scheme 1 and Scheme 2. The basic idea is as follows. Signatures in traceable ABA schemes consist of two parts, one for attribute authentication and the other for providing traceability. To obtain an untraceable ABA scheme from a traceable one, the main task is to remove the part for traceability in the signature together with related supporting parameters and processes in the scheme. Therefore, we cut the part for traceability in Scheme 1 and Scheme 2, more specifically, parameter C_i ($1 \leq i \leq 3$) and R_i ($1 \leq i \leq 5$). Meanwhile, we have kept the part for attribute authentication, which are $CT_{i,j}$, R_{Att} and related parameters in Scheme 1 and the algorithm of computing K_s in Scheme 2. The purpose of discussing these two cases is not to provide a general framework how to build untraceable ABA schemes from traceable ones, but to provide a guideline for the transformation between these two types of ABA schemes.

7.2 Discussion and Future Work

Except for the contributions summarized above, there are still a lot of interesting issues for future research. In this section, we will survey recent work in ABA schemes and point out several interesting research topics. Publications on ABA schemes can

be roughly divided into several fields, including system structures [11, 103, 124], cryptographic construction and security requirements [13, 116], policy specification [12] and so on. In this section, we mainly focus on ABA scheme construction. First of all, we review recent work on ABA construction, with focus on attribute tree building, cryptographic construction, security models, hierarchy, traceability and revocation, with analysis of their advantages and disadvantages. Secondly, we discuss some of open problems in these fields and propose potential directions how to solve them.

7.2.1 Attribute Tree Building

As mentioned earlier in Section 4.2, attribute trees can be built either from top to down [13] or from down to top [20]. Both approaches have been detailedly described in Section 4.2. The top to down approach can be applied to construct both KP-ABA and CP-ABA schemes. For example, all attribute trees built in [13] are from top to down and related schemes are KP-ABA, while all schemes proposed in this dissertation except for schemes 1 and 5 have implemented the top to down approach. On the contrary, to our best knowledge, the only down-to-top based approach in ABA schemes was proposed in [20] from which Scheme 1 originates.

A challenge for current attribute trees is that constrains are not designed to express logical relations like "$\leq$" and so on. For example, if a system divides 24 hours into three intervals, 0:01 to 8:00, 8:01 to 16:00 and 16:01 to 24:00, it needs three attribute elements to express each. As a result, the system needs to generate more attribute keys and use more resources. In [12], this issue is dealing with by using attribute predicates, but it is an approach based on a specification language instead of using attribute trees.

7.2.2 General Framework of Constructing ABA Schemes

In Subsection 3.1.3, we divided ABA schemes into two types, i.e., static and dynamic. This definition was first proposed in [125] to describe different group signatures [126–128]. The main difference is whether the system has the ability to add members into the group at any point of time. If all members in the group are decided during an ABA system's setup, it is static. Otherwise it is dynamic. In most recent work, however, "dynamic" is used to describe systems where users are involved in the generation of their secret keys to prevent key escrow, and it is realized by a

"join in" protocol. Compared with dynamic schemes, all keys are generated by the authority in static ABA schemes and thus there is no need for a "join in" protocol.

Despite of some research on ABA scheme construction, to the best of our knowledge, there is only one publication [13] in which a general framework to construct static ABA schemes is presented. The general framework proposed in [13] can be used to construct ABA schemes from group signature schemes, for example, similar to the work done by Boneh and Shacham [129] and BMW group signature scheme [128]. We will not explain the details of the general framework but only the general idea. To construct an ABA scheme, two necessary parts are needed: an attribute tree and a signature or encryption scheme on which an ABA scheme can be built. To simplify explanations, we call these signature or encryption "base schemes". Attribute trees are used to represent attribute requirements and to authenticate whether the user owns required attributes. The base scheme is used to generate signatures for anonymous authentication. As long as the base scheme is fully anonymous and traceable, the output ABA scheme is also fully anonymous and traceable. To achieve only anonymity, base schemes can be group signatures [130–134], ring signatures [135–142], ABS [14, 143, 144] and ABE [39] schemes. However, ring signatures and ABE schemes usually do not provide traceability and thus can only be utilized to construct untraceable ABA schemes rather than traceable ones. The way how to construct ABA schemes based on either group signatures, ring signatures or ABE schemes are based on pairings. As far as we know, there is only one ABS scheme [145] that is constructed without pairings. The scheme proposed in [145] is obtained by applying Fiat-Shamir transform [146] to the attribute-related identification (ABID) proposed by the authors [145]. Therefore, it gains an advantage in efficiency without pairings. Currently, there are three open issues in the area of general framework of ABA scheme construction.

1. In the general framework to construct static ABA schemes, only anonymity and traceability are discussed. How to achieve other security requirements and what requirements a base scheme should satisfy to build an ABA scheme with different security requirements are still not solved.

2. There is no general framework to build a dynamic ABA scheme from base schemes.

3. From the summary of recent work in Table 7.1, we can see that most schemes are static. As explained before, the "join in" protocol in dynamic schemes

can prevent group members and authorities from impersonating honest users and is thus more privacy preserving. Therefore, more dynamic ABA schemes rather than static ones are preferable considering security and privacy reasons.

7.2.3 Security Models

ABA schemes are built on bilinear groups [11] and their security is based on hard problems in groups or bilinear groups, such as Diffie-Hellman problem (DH) [3], DLP [19] and q-SDH [3]. Moreover, the security requirements of ABA scheme are usually proved under some models, such as random oracle model [15], generic model [20] and standard model [21]. The dilemma in cryptographic application is that there is always a gap between theory and practice, and the same for ABA schemes. The proving of their security requirements is based on these hard problems, cryptographic assumptions and security models. However, using too many assumptions constrains the usability of these schemes. Some schemes are proved secure in random oracles, but their security is still questionable once they are implemented in real systems [38]. Some schemes are even unpractical because of too many assumptions and too complicated computations. Security requirements of most ABA schemes are proved under random oracle model. Compared with random model, the gap between assumptions in generic and standard model and implementations is comparatively more narrow and thus more preferable models to design more practical ABA schemes.

Considering the cryptographic constructions and assumptions used in ABA schemes, attribute-related authorization, ABS and ABE share something in common. In the following, we summarize and compare some recent work to gain an understanding of the usage of security models. The results are shown in Table 1. Before that some explanations are needed for better understanding.

1. Four ABA schemes from [13] are analyzed, from Chapters 5.3.3, 5.4.3, 5.5.3 and 6.4.1 respectively, and we denote them by [13]-1, [13]-2, [13]-3 and [13]-4 accordingly.

2. Two schemes are proposed in [15] and we denote them by [15]-1 and [15]-2 respectively.

3. “CR” is for collision resistant, “D” is for dynamic and “S” is for static.

Table 7.1: Security Comparisons

	Anonymous	Unforgeable	Unlinkable	Traceable	CR	D/S	Model
[13].1	Y	Y	Y	Y	Y	S	RM
[13].2	Y	Y	Y	Y	Y	S	RM
[13].3	Y	Y	Y	Y	Y	D	RM
[13].4	Y	Y	Y	Y	Y	S	RM
[24]	Y	-	-	-	Y	S	RM&GM
[143]	Y	Y	-	N	-	S	SM
[144]	Y	Y	Y	-	-	S	GM
[25]	Y	Y	-	Y	Y	D	RM
[116]	Y	-	Y	-	Y	S	-
[15].1	Y	Y	-	N	-	S	RM
[15].2	Y	Y	-	N	-	S	SM
[143]	Y	Y	-	N	-	S	SM
Scheme 1	Y	Y	Y	Y	Y	S	RM
Scheme 2	Y	Y	Y	Y	Y	S	RM
Scheme 3	Y	Y	Y	Y	Y	S	RM
Scheme 4	Y	Y	Y	Y	Y	S	RM
Scheme 5	Y	-	Y	N	N	S	RM
Scheme 6	Y	-	Y	N	N	S	RM

4. SM, RM and GM are short for standard model, random oracle model and generic model respectively.

5. "-" means no information in the referenced paper.

7.2.4 Hierarchy

Hierarchies have been discussed in Chapter 5 and been divided into the types of delegation [87, 88], hierarchical structure of attributes [90], a hierarchical structure of users [91] and hierarchical security class [77]. Scheme 3 and Scheme 4 proposed in this dissertation are an example of user-related and attribute-related hierarchies respectively.

As discussed in Subsection 7.2.2, since ABA schemes can be built on base schemes such as ring signatures, group signatures or ABE schemes, these base schemes have some similarities with ABA schemes in the organization of hierarchies. First of all, there should be a group of signers in ABA, group or ring signature schemes, no matter the group is real or abstract. Signers in a group or ring signature

scheme generate signatures by a group based private key. However, signers in ABA schemes sign by their own attribute private keys, but the verification is based on the group based public key. This is how they can hide their identities in a signature in ABA, group or ring signature schemes. If tracing is required, there should be a group manager or an authority that has the identity information of all the signers in the group. Besides, identity information should be contained in the signature, so that it can be used by the opener to reveal the signer's identity. However, the identity information cannot be extracted by a verifier.

There are mainly four advantages for using hierarchical structures [24, 45]. First of all, authorities are structured in more fine-grained trust levels, which are closer to applications in real life. Secondly, the property of different trust levels or privileges allows more fine-grained requirements from the verifier. Thirdly, the decentralized structure distributes the workload of the system so that a specific authority will not become the system bottleneck. Finally, the compromise of one non-top authority will not lead to the compromise of the whole system. However, the work on HABA is quite limited, and how to build ABA schemes which combine different hierarchies together is still unsolved.

7.2.5 Traceability

The main purpose of ABA schemes is to achieve anonymity, which promises anonymous authentication, or more specifically, the verifier cannot get any identifying information about the signer during authentication. However, for some systems, traceability is of great importance. When disputes happen and the signers' identities are treated as legitimate evidence, tracing is useful. Because of the property of anonymity in ABA schemes, the verifier itself cannot trace signers' identities, so it has to ask an authority (the "opener") to reveal signers' identities. Whether the system has traceability or not greatly depends on the way how signatures are generated and what is included in the signature. For a traceable signature, it should have at least two parts, one for proving that the signer has the required attributes and one for identity tracing.

Based on whether the ABA schemes are traceable or not, we can divide most of the recent work into two parts. In general, ABA schemes based on group signatures are usually traceable. A typical example is the schemes proposed in [13], where some schemes are based on results from [129, 147]. If ABA schemes are based on ring signatures [148], it is quite possible that they are not traceable, such as

the scheme proposed in [149]. As for using ABE as base schemes, it differs. For example, both Scheme 2 and Scheme 6 are based on ABE schemes, but Scheme 2 is traceable and Scheme 6 is untraceable. However, an ABA scheme with tracing ability usually has longer signatures compared with those untraceable systems. The signature sizes in schemes 2, 3, 4 proposed in this dissertation are comparatively short because of the use of keyed one-way hash function with the key computed based on the required attributes. More approaches about how to design shorter or even constant size signature [143] for a traceable ABA schemes should be an active research area.

7.2.6 Revocation

Revocation is an important feature of ABA systems. There has not been much work focused on ABA revocation, but it is well studied in group signatures [129], identity-based encryption (IBE) [29] and ABE [16, 49, 76] schemes. We will discuss revocation methods that can be used in ABA schemes.

Generally, there are two types of methods for revocation, a revocation list with revoked users or expiration time related attribute keys. The scheme proposed in [129] uses a revocation list, which contains a token representing each revoked user. During system setup, each user should be registered in an authority's database and be known only to the authority. Once they are revoked, they can be found in the database and then added into the revocation list. For this method, the revocation list should be public and available all the time. Thus the biggest disadvantage for this method is that verifiers should have access to the Internet or at least to the server where the revocation list is stored.

The second method is to use expiration time. When signers' attribute keys are generated, they are usually combined with a time expiration date [144]. Compared with the first approach, verifiers do not have to have access to the revocation list, and what they have to do is to check whether the signature is generated by expired keys or not. There are two main drawbacks for this approach. The first one is that when the keys are expired, they need to be updated even though the users are still legitimate, which consumes extra system resources. The other drawback is that it is impossible to revoke either a certain user or an attribute key before they become expired, unless the system has an extra revocation mechanism.

What has been discussed above is categorized by the methods of revocation. If we consider the coarse- or fine-grained revocation approaches, then they can be di-

vided into user-leveled [150] and attribute-leveled [76] revocation as stated in [144]. If the revocation is user-leveled, then once a user is revoked, all his or her attribute keys are revoked. However, if only an attribute of a user is revoked, the other attributes of the user are still usable. In an ABA system, the users are supposed to be dynamic, new members joining and old members leaving or being revoked. We have already discussed two methods based on revocation lists and expiration time, which can be either user- or attribute-leveled. Revocation list based mechanism cannot work in an off-line environment while expiration time mechanism is not flexible enough to revoke either a signer or an attribute at arbitrary point. There is still a need for methods that combine user-leveled with attribute-leveled revocation together to provide more efficiency and flexibility.

REFERENCES

[1] N. Li, L. Zhang, P. Xu, L. Wang, J. Zheng, and Y. Guo, "Research on pricing model of cloud storage," in *2013 IEEE Ninth World Congress on Services (SERVICES)*, June 2013, pp. 412–419.

[2] D. Slamanig and C. Hanser, "On cloud storage and the cloud of clouds approach," in *2012 International Conference for Internet Technology And Secured Transactions*, Dec 2012, pp. 649–655.

[3] S. Rahaman and M. Farhatullah, "PccP: A model for preserving cloud computing privacy," in *2012 International Conference on Data Science Engineering (ICDSE)*, July 2012, pp. 166–170.

[4] S. Hamouda, "Security and privacy in cloud computing," in *2012 International Conference on Cloud Computing Technologies, Applications and Management (ICCCTAM)*, Dec 2012, pp. 241–245.

[5] C. M. Yu, C. Y. Chen, and H. C. Chao, "Privacy-preserving multikeyword similarity search over outsourced cloud data," *Systems Journal, IEEE*, vol. PP, no. 99, pp. 1–10, 2015.

[6] Z. Li, C. Wilson, Z. Jiang, Y. Liu, B. Zhao, C. Jin, Z.-L. Zhang, and Y. Dai, "Efficient batched synchronization in dropbox-like cloud storage services," in *Middleware 2013*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8275, pp. 307–327.

[7] M. Privat and R. Warner, "Talking to services: iCloud and dropbox," in *Pro iOS Persistence*. Apress, 2014, pp. 259–286.

[8] C. H. Vincent, F. David, K. Rick, S. Adam, S. Kenneth, and S. Karen, "NIST special publication 800-162: Guide to attribute based AC (ABAC) definition and considerations," 2014. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-162

[9] M. Benantar, "Foundations of security and ac in computing," in *AC Systems*. Springer US, 2006, pp. 1–39.

[10] H. Vincent C., F. F. David, and D. R. Kuhn, "Assessment of ac systems," 2006.

[11] C. Schlger, M. Sojer, B. Muschall, and G. Pernul, "Attribute-based authentication and authorisation infrastructures for e-commerce providers," in *E-Commerce and Web Technologies*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4082, pp. 132–141.

[12] J. Camenisch, M. Dubovitskaya, A. Lehmann, G. Neven, C. Paquin, and F.-S. Preiss, "Concepts and languages for privacy-preserving attribute-based authentication," in *Policies and Research in Identity Management*, ser. IFIP Advances in Information and Communication Technology. Springer Berlin Heidelberg, 2013, vol. 396, pp. 34–52.

[13] D. D. Khader, "Attribute-based authentication scheme," Ph.D. dissertation, University of Bath, 2009.

[14] H. Maji, M. Prabhakaran, and M. Rosulek, "Attribute-based signatures," in *Topics in Cryptology CT-RSA 2011*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6558, pp. 376–392.

[15] J. Li, M. H. Au, W. Susilo, D. Xie, and K. Ren, "Attribute-based signature and its applications," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. ACM, 2010, pp. 60–69.

[16] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. ACM, 2010, pp. 261–270.

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

[18] M. De Soete, "Pki," in *Encyclopedia of Cryptography and Security*. Springer US, 2011, pp. 935–936.

[19] "Pki trust relationships," in *Encyclopedia of Cryptography and Security*. Springer US, 2011, pp. 939–939.

[20] H. Yang and A. Oleshchuk, Vladimir, "A dynamic attribute-based authentication scheme," in *Codes, Cryptology, and Information Security*. Springer International Publishing, 2015, vol. 9084, pp. 106–118.

[21] H. Yang and V. A. Oleshchuk, "An efficient traceable attribute-based authentication scheme with one-time attribute trees," in *Secure IT Systems*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9417, pp. 123–135.

[22] ——, "Traceable hierarchical attribute-based authentication for the cloud," in *2015 IEEE Conference on Communications and Network Security (CNS)*, Sept 2015, pp. 685–689.

[23] H. Yang and A. Oleshchuk, Vladimir, "A survey on the security and privacy of attribute-based authentication schemes," *International Journal of Computing*, vol. 14, no. 2, 2015.

[24] X. Liu, Y. Xia, S. Jiang, F. Xia, and Y. Wang, "Hierarchical attribute-based access control with authentication for outsourced data in cloud computing," in *2013 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, July 2013, pp. 477–484.

[25] K. Emura, A. Miyaji, and K. Omote, "A dynamic attribute-based group signature scheme and its application in an anonymous survey for the collection of attribute statistics," in *2009 International Conference on Availability, Reliability and Security (ARES '09)*, March 2009, pp. 487–492.

[26] K. N. Saravana, L. G. Rajya, and B. Balamurugan, "Enhanced attribute based encryption for cloud computing," *Procedia Computer Science*, vol. 46, pp. 689 – 696, 2015.

[27] M. Surya and N. N. Anithadevi, "Single sign on mechanism using attribute based encryption in distributed computer networks," *Procedia Computer Science*, vol. 47, pp. 441 – 451, 2015.

[28] N. Attrapadung, J. Herranz, F. Laguillaumie, B. Libert, E. de Panafieu, and C. Rafols, "Attribute-based encryption schemes with constant-size ciphertexts," *Theoretical Computer Science*, vol. 422, pp. 15 – 38, 2012.

[29] S. Arita, "Flexible attribute-based encryption," in *Information and Communications Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7618, pp. 471–478.

[30] T. Naruse, M. Mohri, and Y. Shiraishi, "Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating," *Human-centric Computing and Information Sciences*, vol. 5, no. 1, 2015.

[31] K. Kaushik, V. Varadharajan, and R. Nallusamy, "Multi-user attribute based searchable encryption," in *2013 IEEE 14th International Conference on Mobile Data Management (MDM)*, vol. 2, June 2013, pp. 200–205.

[32] R. Manjusha and R. Ramachandran, "Comparative study of attribute based encryption techniques in cloud computing," in *2014 International Conference on Embedded Systems (ICES)*, July 2014, pp. 116–120.

[33] X. Liu, H. Zhu, J. Ma, J. Ma, and S. Ma, "Key-policy weighted attribute based encryption for fine-grained access control," in *2014 IEEE International Conference on Communications Workshops (ICC)*, June 2014, pp. 694–699.

[34] H. Wang, B. Yang, and Y. Wang, "Server aided ciphertext-policy attribute-based encryption," in *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, March 2015, pp. 440–444.

[35] Y. Shi, Q. Zheng, J. Liu, and Z. Han, "Directly revocable key-policy attribute-based encryption with verifiable ciphertext delegation," *Information Sciences*, vol. 295, pp. 221 – 231, 2015.

[36] N. Attrapadung and S. Yamada, "Duality in ABE: Converting attribute based encryption for dual predicate and dual policy via computational encodings," in *Topics in Cryptology - CT-RSA 2015*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 9048, pp. 87–105.

[37] Y. Wang, K. Chen, Y. Long, and Z. Liu, "Accountable authority key policy attribute-based encryption," *Science China Information Sciences*, vol. 55, no. 7, pp. 1631–1638, 2012.

[38] C. Wang and Y. Liu, "A secure and efficient key-policy attribute based key encryption scheme," in *2009 1st International Conference on Information Science and Engineering (ICISE)*, Dec 2009, pp. 1601–1604.

[39] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM Conference on Computer and Communications Security*, ser. CCS '06. New York, NY, USA: ACM, 2006, pp. 89–98.

[40] Z. Zhou, D. Huang, and Z. Wang, "Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption," *IEEE Transactions on Computers*, vol. 64, no. 1, pp. 126–138, Jan 2015.

[41] J. Xu, Q. Wen, W. Li, and Z. Jin, "Circuit ciphertext-policy attribute-based hybrid encryption with verifiable delegation in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, no. 99, pp. 1–1, 2015.

[42] X. Liang, Z. Cao, H. Lin, and D. Xing, "Provably secure and efficient bounded ciphertext policy attribute based encryption," in *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ser. ASIACCS '09. New York, NY, USA: ACM, 2009, pp. 343–352.

[43] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '11. New York, NY, USA: ACM, 2011, pp. 386–390.

[44] M. Horvth, "Attribute-based encryption optimized for cloud computing," in *SOFSEM 2015: Theory and Practice of Computer Science*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2015, vol. 8939, pp. 566–577.

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

[46] M. Gorantla, C. Boyd, and J. Gonzlez Nieto, "Attribute-based authenticated key exchange," in *Information Security and Privacy*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6168, pp. 300–317.

[47] D. Cao, B. Zhao, X. Wang, J. Su, and Y. Chen, "Authenticating with attributes in online social networks," in *2011 14th International Conference on Network-Based Information Systems (NBiS)*, Sept 2011, pp. 607–611.

[48] S. Zhu, L. Zhan, H. Qiang, D. Fu, W. Sun, and Y. Tang, "A fuzzy attribute-based authentication scheme on the basis of lagrange polynomial interpolation," in *Human Centered Computing*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 8944, pp. 685–692.

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

[50] Y. Zhang, M. Wu, L. Wu, and Y. Li, "Attribute-based access control security model in service-oriented computing," in *Proceedings of the 2012 International Conference on Cybernetics and Informatics*, ser. Lecture Notes in Electrical Engineering. Springer New York, 2014, vol. 163, pp. 1473–1479.

[51] S. Ruj, "Attribute based access control in clouds: A survey," in *2014 International Conference on Signal Processing and Communications (SPCOM)*, July 2014, pp. 1–6.

[52] H. Shen, "A semantic-aware attribute-based access control model for web services," in *Algorithms and Architectures for Parallel Processing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5574, pp. 693–703.

[53] A. Armando, R. Carbone, E. G. Chekole, and S. Ranise, "Attribute based access control for APIs in spring security," in *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '14. ACM, 2014, pp. 85–88.

[54] S. Ding, Y. Zhao, and Y. Liu, "Efficient traceable attribute-based signature," in *2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Sept 2014, pp. 582–589.

[55] P. P. Chandar, D. Mutkuraman, and M. Rathinrai, "Hierarchical attribute based proxy re-encryption access control in cloud computing," in *2014 In-*

ternational Conference on Circuit, Power and Computing Technologies (ICCPCT), March 2014, pp. 1565–1570.

[56] E. Caprin and Y. Zhang, "Negotiation based framework for attribute-based access control policy evaluation," in *Proceedings of the 7th International Conference on Security of Information and Networks*, ser. SIN '14. ACM, 2014, pp. 122–127.

[57] L. D. Kudryavtsev and M. Samarin, "Lagrange interpolation formula," Last modified in 2011. [Online]. Available: http://www.encyclopediaofmath.org/index.php?title=Lagrange_interpolation_formula&oldid=17497

[58] B. Archer and E. W. Weisstein, "Lagrange interpolation formula," Last accessed in July, 2015. [Online]. Available: http://mathworld.wolfram.com/LagrangeInterpolatingPolynomial.html

[59] M. M. Jahani Yekta, "Equivalence of the lagrange interpolator for uniformly sampled signals and the scaled binomially windowed shifted sinc function," *Digit. Signal Process.*, vol. 19, no. 5, pp. 838–842, sep 2009.

[60] T. Okamoto, "Cryptography based on bilinear maps," in *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3857, pp. 35–50.

[61] R. Sahu and S. Padhye, "Efficient ID-based signature scheme from bilinear map," in *Advances in Parallel Distributed Computing*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2011, vol. 203, pp. 301–306.

[62] Y. Qin, D. Feng, and X. Zhen, "An anonymous property-based attestation protocol from bilinear maps," in *2009 International Conference on Computational Science and Engineering (CSE '09)*, vol. 2, Aug 2009, pp. 732–738.

[63] M. Bellare, "A note on negligible functions," *Journal of Cryptology*, vol. 15, no. 4, pp. 271–284, 2002.

[64] O. Goldreich, *The Foundations of Cryptography: Volume 1, Basic Tools (1st Edition)*. Cambridge, United Kingdom: Cambridge University Press, 2007.

[65] D. M. Gordon, "Discrete logarithm problem," in *Encyclopedia of Cryptography and Security*. Springer US, 2005, pp. 164–168.

[66] K. Nyberg and R. A. Rueppel, "Message recovery for signature schemes based on the discrete logarithm problem," *Designs, Codes and Cryptography*, vol. 7, no. 1-2, pp. 61–81, 1996.

[67] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Advances in Cryptology - EUROCRYPT 2004*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3027, pp. 56–73.

[68] D. Jao and K. Yoshida, "Boneh-boyen signatures and the strong diffie-hellman problem," in *Pairing-Based Cryptography Pairing 2009*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5671, pp. 1–16.

[69] R. S. Winternitz, "Producing a one-way hash function from DES," in *Advances in Cryptology*. Springer US, 1984, pp. 203–207.

[70] J. Katz, "The random oracle model," in *Digital Signatures*. Springer US, 2010, pp. 135–142.

[71] L. Granboulan, "Short signatures in the random oracle model," in *Advances in Cryptology ASIACRYPT 2002*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2501, pp. 364–378.

[72] R. Hady and A. Agranovsky, "Crypto miracles with random oracle," in *IEEE-2001 Siberian Workshop of Students and Young Researches (SIBCOM-2001) Modern Communication Technologies*, 2001, pp. 20–22.

[73] R. Canetti, O. Goldreich, and S. Halevi, "The random oracle methodology, revisited," *J. ACM*, vol. 51, no. 4, pp. 557–594, Jul. 2004.

[74] M. Bellare and O. Goldreich, "Proving computational ability," in *Studies in Complexity and Cryptography: Miscellanea on the Interplay between Randomness and Computation*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6650, pp. 6–12.

[75] S. Bakhtiari, R. Safavi-Naini, and J. Pieprzyk, "Keyed hash functions," in *Cryptography: Policy and Algorithms*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1996, vol. 1029, pp. 201–214.

[76] T. Naruse, M. Mohri, and Y. Shiraishi, "Attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating," in *Future Information Technology*, ser. Lecture Notes in Electrical Engineering. Springer Berlin Heidelberg, 2014, vol. 276, pp. 119–125.

[77] D. Xu, F. Luo, L. Gao, and Z. Tang, "Fine-grained document sharing using attribute-based encryption in cloud servers," in *2013 Third International Conference on Innovative Computing Technology (INTECH)*, Aug 2013, pp. 65–70.

[78] J. Lai, R. H. Deng, Y. Li, and J. Weng, "Fully secure key-policy attribute-based encryption with constant-size ciphertexts and fast decryption," in *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ser. ASIA CCS '14. New York, NY, USA: ACM, 2014, pp. 239–248.

[79] D. Servos and S. Osborn, "HGABAC: Towards a formal model of hierarchical attribute-based access control," in *Foundations and Practice of Security*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 8930, pp. 187–204.

[80] R. Bobba, H. Khurana, and M. Prabhakaran, "Attribute-sets: A practically motivated enhancement to attribute-based encryption," in *Computer Security ESORICS 2009*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5789, pp. 587–604.

[81] F. Han, J. Qin, H. Zhao, and J. Hu, "A general transformation from KP-ABE to searchable encryption," in *Cyberspace Safety and Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7672, pp. 165–178.

[82] S. Yu, K. Ren, W. Lou, and J. Li, "Defending against key abuse attacks in KP-ABE enabled broadcast systems," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer Berlin Heidelberg, 2009, vol. 19, pp. 311–329.

[83] Z. Liu, Z. Cao, and D. Wong, "Fully collusion-resistant traceable key-policy attribute-based encryption with sub-linear size ciphertexts," in *Information Security and Cryptology*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2015, vol. 8957, pp. 403–423.

[84] M. Padhya and D. Jinwala, "A novel approach for searchable CP-ABE with hidden ciphertext-policy," in *Information Systems Security*, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, vol. 8880, pp. 167–184.

[85] A. Xiong, C. Xu, and Q. Gan, "A CP-ABE scheme with system attributes revocation in cloud storage," in *Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2014 11th International Computer Conference on*, Dec 2014, pp. 331–335.

[86] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, May 2007, pp. 321–334.

[87] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology EUROCRYPT 2010*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6110, pp. 62–91.

[88] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proceedings of the 17th ACM Conference on Computer and Communications Security*, ser. CCS '10. New York, NY, USA: ACM, 2010, pp. 735–737.

[89] M. Asim, T. Ignatenko, M. Petkovic, D. Trivellato, and N. Zannone, "Enforcing access control in virtual organizations using hierarchical attribute-based encryption," in *2012 Seventh International Conference on Availability, Reliability and Security (ARES)*, Aug 2012, pp. 212–217.

[90] J. Li, Q. Wang, C. Wang, and K. Ren, "Enhancing attribute-based encryption with attribute hierarchy," *Mobile Networks and Applications*, vol. 16, no. 5, pp. 553–561, 2011.

[91] J. Liu, Z. Wan, and M. Gu, "Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing," in *Information Security Practice and Experience*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6672, pp. 98–107.

[92] N. Li, "Discretionary access control," in *Encyclopedia of Cryptography and Security*. Springer US, 2011, pp. 353–356.

[93] S. Dranger, R. Sloan, and J. Solworth, "The complexity of discretionary access control," in *Advances in Information and Computer Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4266, pp. 405–420.

[94] D. Slamanig, "Dynamic accumulator based discretionary access control for outsourced storage with unlinkable access," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7397, pp. 215–222.

[95] M. Benantar, "Mandatory-access-control model," in *AC Systems*. Springer US, 2006, pp. 129–146.

[96] L. Tian, X. Rong, and T. Liu, "Design and implementation of linux file mandatory access control," in *Network Computing and Information Security*, ser. Communications in Computer and Information Science, J. Lei, F. Wang, M. Li, and Y. Luo, Eds. Springer Berlin Heidelberg, 2012, vol. 345, pp. 15–22.

[97] J. Jafarian, M. Amini, and R. Jalili, "A dynamic mandatory access control model," in *Advances in Computer Science and Engineering*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2009, vol. 6, pp. 862–866.

[98] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based AC," *ACM Trans. Inf. Syst. Secur.*, vol. 4, no. 3, pp. 224–274, Aug. 2001.

[99] S. Osborn, "Role-based access control," in *Security, Privacy, and Trust in Modern Data Management*, ser. Data-Centric Systems and Applications. Springer Berlin Heidelberg, 2007, pp. 55–70.

[100] D. Ferraiolo and R. Kuhn, "Role based AC," in *Proceddings of 15th NIST-NCSC National Computer Security Conference*, 1992, pp. 554–563.

[101] V. Alturi and D. Ferraiolo, "Role-based access control," in *Encyclopedia of Cryptography and Security*. Springer US, 2011, pp. 1053–1055.

[102] Y. Zhang and J. Joshi, "Role based access control," in *Encyclopedia of Database Systems*. Springer US, 2009, pp. 2447–2452.

[103] M. Covington, M. Sastry, and D. Manohar, "Attribute-based authentication model for dynamic mobile environments," in *Security in Pervasive Computing*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3934, pp. 227–242.

[104] S. Ruj, M. Stojmenovic, and A. Nayak, "Privacy preserving access control with authentication for securing data in clouds," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, ser. CCGRID '12. IEEE Computer Society, 2012, pp. 556–563.

[105] L. Guo, C. Zhang, J. Sun, and Y. Fang, "A privacy-preserving attribute-based authentication system for mobile health networks," *IEEE Transactions on Mobile Computing*, vol. 13, no. 9, pp. 1927–1941, Sept 2014.

[106] H. Wang and Y. Ren, "An attribute-based anonymous authentication scheme," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT)*, Sept 2013, pp. 569–573.

[107] S. Dolev, L. Krzywiecki, N. Panwar, and M. Segal, "Dynamic attribute based vehicle authentication," in *2014 IEEE 13th International Symposium on Network Computing and Applications (NCA)*, Aug 2014, pp. 1–8.

[108] H. a Park, D. H. Lee, and J. Zhan, "Attribute-based access control using combined authentication technologies," in *IEEE International Conference on Granular Computing (GrC 2008)*, Aug 2008, pp. 518–523.

[109] Z. Wang, Q. Wu, D. Ye, and H. Chen, "Practical identity-based aggregate signature from bilinear maps," *Journal of Shanghai Jiaotong University (Science)*, vol. 13, no. 6, pp. 684–687, 2008.

[110] M. Gagn, S. Narayan, and R. Safavi-Naini, "Threshold attribute-based signcryption," in *Security and Cryptography for Networks*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6280, pp. 154–171.

[111] T. Pandit and R. Barua, "Efficient fully secure attribute-based encryption schemes for general access structures," in *Provable Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7496, pp. 193–214.

[112] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM, 2013, pp. 463–474.

[113] L. Nwosu, A. Thompson, B. Alese, and O. Obe, "An attribute-based signature using rivest shamir adleman scheme," in *2014 9th International Conference for Internet Technology and Secured Transactions (ICITST)*, Dec 2014, pp. 380–386.

[114] D. Khader, "Attribute based search in encrypted data: ABSE," in *Proceedings of the 2014 ACM Workshop on Information Sharing & Collaborative Security*, ser. WISCS '14. New York, NY, USA: ACM, 2014, pp. 31–40.

[115] F. Zeng, C. Xu, X. Zhang, and J. Liu, "Strongly secure attribute-based authenticated key exchange with traceability," in *Web Information Systems and Mining*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2012, vol. 7529, pp. 231–238.

[116] L. Guo, C. Zhang, J. Sun, and Y. Fang, "PAAS: A privacy-preserving attribute-based authentication system for ehealth networks," in *2012 IEEE 32nd International Conference on Distributed Computing Systems (ICDCS)*, June 2012, pp. 224–233.

[117] W. Hongbin and R. Yan, "An attribute-based anonymous authentication scheme," in *2013 Fourth International Conference on Emerging Intelligent Data and Web Technologies (EIDWT)*, Sept 2013, pp. 569–573.

[118] C. Solares, E. Vieira, and R. Mnguez, "Functional networks and the lagrange polynomial interpolation," in *Intelligent Data Engineering and Automated*

Learning IDEAL 2006, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 4224, pp. 394–401.

[119] X. Song, H. Deng, and Z. Xiong, "Using piecewise hashing and lagrange interpolation polynomial to preserve electronic evidence," in *Advances in Information Technology and Education*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2011, vol. 201, pp. 472–480.

[120] H. Lin, D. Pan, X. Zhao, and Z. Qiu, "A rapid and efficient pre-deployment key scheme for secure data transmissions in sensor networks using lagrange interpolation polynomial," in *2008 International Conference on Information Security and Assurance (ISA 2008)*, April 2008, pp. 261–265.

[121] M. Blanton, "Message authentication codes," in *Encyclopedia of Database Systems*. Springer US, 2009, pp. 1715–1716.

[122] F. Yang, C. Zhong, and D. Lu, "Cross message authentication code based on multi-core computing technology," in *Advances in Future Computer and Control Systems*, ser. Advances in Intelligent and Soft Computing. Springer Berlin Heidelberg, 2012, vol. 159, pp. 181–186.

[123] R. Alosaimy, K. Alghathbar, A. Hafez, and M. Eldefrawy, "NMACA approach used to build a secure message authentication code," in *Security Technology, Disaster Recovery and Business Continuity*, ser. Communications in Computer and Information Science. Springer Berlin Heidelberg, 2010, vol. 122, pp. 290–298.

[124] T. Priebe, W. Dobmeier, C. Schlger, and N. Kamprath, "Supporting attribute-based access control in authorization and authentication infrastructures with ontologies," *Journal of Software*, vol. 2, no. 1, 2007.

[125] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups," in *Advances in Cryptology CRYPTO'97*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1997, vol. 1294, pp. 410–424.

[126] A. Giuseppe, C. Jan, H. Susan, and d. M. Breno, "Practical group signatures without random oracles," 2006.

[127] D. Chaum and E. van Heyst, "Group signatures," in *Advances in Cryptology EUROCRYPT 91*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 1991, vol. 547, pp. 257–265.

[128] M. Bellare, D. Micciancio, and B. Warinschi, "Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions," in *Advances in Cryptology EUROCRYPT 2003*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2656, pp. 614–629.

[129] D. Boneh and H. Shacham, "Group signatures with verifier-local revocation," in *Proceedings of the 11th ACM Conference on Computer and Communications Security*, ser. CCS '04. New York, NY, USA: ACM, 2004, pp. 168–177.

[130] G. Bleumer, "Group signatures," in *Encyclopedia of Cryptography and Security*, H. van Tilborg and S. Jajodia, Eds. Springer US, 2011, pp. 526–528.

[131] S. Zhou and D. Lin, "Unlinkable randomizable signature and its application in group signature," in *Information Security and Cryptology*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, vol. 4990, pp. 328–342.

[132] S. Mohanty, B. Majhi, and V. Iyer, "A strong designated verifiable group signature scheme," in *2013 International Multi-Conference on Automation, Computing, Communication, Control and Compressed Sensing (iMac4s)*, March 2013, pp. 518–523.

[133] B. Libert, T. Peters, and M. Yung, "Scalable group signatures with revocation," in *Proceedings of the 31st Annual International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 609–627.

[134] B. Libert and M. Yung, "Dynamic fully forward-secure group signatures," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '10. New York, NY, USA: ACM, 2010, pp. 70–81.

[135] F. Zhang and K. Kim, "ID-based blind signature and ring signature from pairings," in *Advances in Cryptology ASIACRYPT 2002*, ser. Lecture Notes

in Computer Science. Springer Berlin Heidelberg, 2002, vol. 2501, pp. 533–547.

[136] J. Herranz and G. Sez, "Forking lemmas for ring signature schemes," in *Progress in Cryptology - INDOCRYPT 2003*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2003, vol. 2904, pp. 266–279.

[137] J. Herranz and G. Sez, "New identity-based ring signature schemes," in *Information and Communications Security*, ser. Lecture Notes in Computer Science, J. Lopez, S. Qing, and E. Okamoto, Eds. Springer Berlin Heidelberg, 2004, vol. 3269, pp. 27–39.

[138] J. Lee and J. Chang, "Strong designated verifier ring signature scheme," in *Innovations and Advanced Techniques in Computer and Information Sciences and Engineering*. Springer Netherlands, 2007, pp. 543–547.

[139] J. Ren and L. Harn, "Generalized ring signatures," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 3, pp. 155–163, July 2008.

[140] X. Zhou, "Study on ring signature and its application," in *2009 Chinese Control and Decision Conference (CCDC'09)*, June 2009, pp. 1388–1393.

[141] T. H. Yuen, J. K. Liu, M. H. Au, W. Susilo, and J. Zhou, "Threshold ring signature without random oracles," in *Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS'11. New York, NY, USA: ACM, 2011, pp. 261–267.

[142] S. S. M. Chow, V. K. Wei, J. K. Liu, and T. H. Yuen, "Ring signatures without random oracles," in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS'06. New York, NY, USA: ACM, 2006, pp. 297–302.

[143] A. Ge, C. G. Ma, and Z. F. Zhang, "Attribute-based signature scheme with constant size signature in the standard model," *Information Security, IET*, vol. 6, no. 2, pp. 47–54, June 2012.

[144] Y. Lian, L. Xu, and X. Huang, "Attribute-based signatures with efficient revocation," in *2013 5th International Conference on Intelligent Networking and Collaborative Systems (INCoS)*, 2013, pp. 573–577.

[145] H. Anada, S. Arita, and K. Sakurai, "Attribute-based signatures without pairings via the fiat-shamir paradigm," in *Proceedings of the 2nd ACM Workshop on ASIA Public-key Cryptography*, ser. ASIAPKC '14. New York, NY, USA: ACM, 2014, pp. 49–58.

[146] M. Abdalla, J. An, M. Bellare, and C. Namprempre, "From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security," in *Advances in Cryptology EUROCRYPT 2002*, ser. Lecture Notes in Computer Science, L. Knudsen, Ed. Springer Berlin Heidelberg, 2002, vol. 2332, pp. 418–433.

[147] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology CRYPTO 2004*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2004, vol. 3152, pp. 41–55.

[148] R. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret: Theory and applications of ring signatures," in *Theoretical Computer Science*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2006, vol. 3895, pp. 164–186.

[149] W. Wenqiang and C. Shaozhen, "Attribute-based ring signature scheme with constant-size signature," *Information Security, IET*, vol. 4, no. 2, pp. 104–110, June 2010.

[150] J. Ye, W. Zhang, S. Wu, Y. Gao, and J. Qiu, "Attribute-based fine-grained access control with user revocation," in *Information and Communication Technology*, ser. Lecture Notes in Computer Science, Linawati, M. Mahendra, E. Neuhold, A. Tjoa, and I. You, Eds. Springer Berlin Heidelberg, 2014, vol. 8407, pp. 586–595.

Appendices A

List of Publications

The purpose of this preface is to record all the articles published, accepted and submitted by the author of this dissertation. The followings are the papers included in this dissertation:

1. **Huihui Yang**, Vladimir A. Oleshchuk and Andreas Prinz, "Verifying Group Authentication Protocols by Scyther (accepted)", *the Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA)*, Vol. 7, 2016.

2. Mohamed Abomhara and **Huihui Yang**, "Attribute-Based Authenticated Access for Secure Sharing of Healthcare Records in Collaborative Environments", *The Eighth International Conference on eHealth, Telemedicine, and Social Medicine (eTELEMED 2016)*, pp. 138-144, 2016.

3. **Huihui Yang**, Andreas Prinz and Vladimir A. Oleshchuk, "Formal Analysis and Model Checking of a Group Authentication Protocol by Scyther", *24th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing (PDP)*, pp. 553-557, 2016.

4. **Huihui Yang** and Vladimir A. Oleshchuk, "Traceable Hierarchical Attribute-based Authentication for the Cloud", *2015 IEEE Conference on Communications and Network Security (CNS)*, pp. 685-689, 2015.

5. **Huihui Yang** and Vladimir A. Oleshchuk, "An Efficient Traceable ABA Schemes with One-time Attribute Trees", *20th Nordic Conference, NordSec 2015*, Vol. 9417, pp. 123-135, LNCS, 2015.

6. **Huihui Yang** and Vladimir A. Oleshchuk, “A Survey on the Security and Privacy of Attribute-based Authentication Schemes”, *International Journal of Computing*, Vol. 14, Issue 2, 2015.

7. **Huihui Yang** and Vladimir A. Oleshchuk, “A Dynamic Attribute-Based Authentication Scheme”, *International Conference in “Codes, Cryptology and Information Security”*, Vol. 9085, pp. 106-118, LNCS, 2015.

8. **Huihui Yang** and Vladimir A. Oleshchuk, “An improvement of the batch-authentication and key agreement framework for P2P-based online social networks”, *2014 International Conference on Privacy and Security in Mobile Systems (PRISMS)*, pp. 1-4, 2014.

9. **Huihui Yang**, Lei Jiao and Vladimir A. Oleshchuk, “A General Framework for Group Authentication and Key Exchange Protocols”, *6th International symposium on foundations & Practice in security FPS’ 2013*, Vol. 8352, pp. 31-45, LNCS, 2014.