ACCESS MANAGEMENT IN ELECTRONIC COMMERCE SYSTEM

by

Hua Wang

A thesis submitted to

The Department of Mathematics and Computing

University of Southern Queensland

for the degree of

Doctor of Philosophy

Statement

I hereby declare that the work presented in this dissertation is my own and is, to the best of my knowledge and belief, original except as acknowledged in the text. It has not previously been submitted either in whole or in part for a degree at this or any other university.

Hua Wang

Acknowledgement

I express my sincere gratitude and appreciation to my PhD supervisor Dr Yanchun Zhang of University of Southern Queensland (USQ) for his invaluable advice, teaching and encouragements during the realization of this research. It is his love for his students and dedication to research that made them available whenever I needed during all these years. If it was not for him I may not be getting my PhD.

I also express my gratitude and appreciation to my PhD co-supervisor Dr Jinli Cao of La Trobe University for all her valuable guidance, encouragement and support in every aspect during my PhD study.

I sincerely thank the Department of Mathematics and Computing, Faculty of science and High Degree Committee of the USQ for providing the excellent study environment and the financial support. I highly appreciate the great support from Professor Tony Roberts and Associate Professor Chris Harman during my study at the USQ. It is a great pleasure to study at the Department of Mathematics and Computing.

I thank Mrs Ruth Hilton for their support in many aspects. Thanks also goes to my friends at USQ and abroad for their friendship, valuable advice and encouragements that made my stay a memorable one.

I also acknowledge all the help from those who carefully read the dissertation

and made the English corrections.

Finally, my appreciation goes to my mother Jiqi, my wife Lili and daughter Haina for their love and affection. I could not be able to complete my PhD study without their encouragement and support.

Abstract

The definition of Electronic commerce is the use of electronic transmission mediums to engage in the exchange, including buying and selling, of products and services requiring transportation, either physically or digitally, from location to location [65]. Electronic commerce systems, including mobile e-commerce, are widely used since 1990 [15]. The number of world-wide Internet users tripled between 1993 and 1995 to 60 million, and by 2000 there were 250 million users. More than one hundred countries have Internet access. Electronic commerce, especial mobile e-commerce systems, allows their users to access a large set of traditional (for example, voice communications) and contemporary (for example, e-shop) services without being tethered to one particular physical location. With the increasing use of electronic service systems for security sensitive application (for example, e-shop) that can be expected in the future, the provision of secure services becomes more important. The dynamic mobile environment is incompatible with static security services. Electronic service access across multiple service domains, and the traditional access mechanisms rely on cross-domain authentication using roaming agreements starting home location. Cross-domain authentication involves many complicated authentication activities when the roam path is long. This limits future electronic commerce applications.

Normally, there are three participants in an electronic service. These are users, service providers, and services. Some services bind users and service providers as

well as services such as flight services; other services do not bind any participants, for instance by using cash in shopping services, everyone can use cash to buy anything in shops. Hence, depending on which parts are bound, there are different kinds of electronic services. However, there is no scheme to provide a solution for all kinds of electronic services. Users have to change service systems if they want to apply different kind of electronic services on the Internet. From the consumer's point of view, users often prefer to have a total solution for all kinds of service problems, some degree of anonymity with no unnecessary cross authentications and a clear statement of account when shopping over the Internet. There are some suggested solutions for electronic service systems [68, 69, 42, 52, 66, 61, 121], but the solutions are neither total solutions for all kinds of services nor have some degree of anonymity with a clear statement of account.

In our work, we build a bridge between existing technologies and electronic service theory such as e-payment, security and so on. We aim to provide a foundation for the improvement of technology to aid electronic service application. As validation, several technologies for electronic service system design have been enhanced and improved in this project. To fix the problems mentioned above, we extend our idea to a ticket based access service system.

The user in the above electronic service system has to pay when s/he obtains service. S/He can pay by traditional cash (physical cash), check, credit or electronic cash. The best way to pay money for goods or services on the Internet is using electronic cash [109]. Consumers, when shopping over the Internet, often prefer to have a high level of anonymity with important things and a low level with general one. The ideal system needs to provide some degree of anonymity for consumers so that they cannot be traced by banks. There are a number of proposals for electronic

cash systems [22, 30, 67, 26, 20, 81, 79]. All of them are either too large to manage or lack flexibility in providing anonymity. Therefore, they are not suitable solutions for electronic payment in the future.

We propose a secure, scalable anonymity and practical payment protocol for Internet purchases [119]. The protocol uses electronic cash for payment transactions. In this new protocol, from the viewpoint of banks, consumers can improve anonymity if they are worried about disclosure of their identities. An agent, namely anonymity provider agent provides a higher anonymous certificate and improves the security of the consumers. The agent will certify re-encrypted data after verifying the validity of the content from consumers, but with no private information of the consumers required. With this new method, each consumer can get the required anonymity level.

Electronic service systems involve various subsystems such as service systems, payment systems, and management systems. Users and service providers are widely distributed and use heterogeneous catalog systems. They are rapidly increasing in dynamic environments. The management of these service systems will be very complex. Whether systems are successful or not depends on the quality of their management. To simplify the management of e-commerce systems [90], we discuss role based access control management. We define roles and permissions in the subsystems. For example, there are roles TELLER, AUDITOR, MANAGER and permissions teller (account operation), audit operation, managerial decision in a bank system. Permissions are assigned to roles such as permission teller is assigned to role TELLER. People (users) employed in the bank are granted roles to perform associated duties. However, there are conflicts between various roles as well as between various permissions. These conflicts may cause serious security problems with

the bank system. For instance, if permissions teller and audit operation are assigned to a role, then a person with this role will have too much privilege to break the security of the bank system [28]. Therefore, the organizing of relationships between users and roles, roles and permissions currently requires further development.

Role based access control (RBAC) has been widely used in database management and operating systems. In 1993, the National Institute of Standards and Technology (NIST) developed prototype implementations, sponsored external research [38], and published formal RBAC models [39, 48]. Since then, many RBAC practical applications have been implemented [6, 40, 89], because RBAC has many advantages such as reducing administration cost and complexity.

However, there are some problems which may arise in RBAC management. One is related to authorization granting process. For example, when a role is granted to a user, this role may conflict with other roles of the user or together with this role; the user may have or derive a high level of authority. Another is related to authorization revocation. For instance, when a role is revoked from a user, the user may still have the role.

To solve these problems, we present an authorization granting algorithm, and weak revocation and strong revocation algorithms that are based on relational algebra. The algorithms check conflicts and therefore help allocate the roles and permissions without compromising the security in RBAC. We describe the applications of the new algorithms with an anonymity scalable payment scheme.

In summary, this thesis has made the following major contributions in electronic service systems:

- A ticket based global solution for electronic commerce systems;

A ticket based solution is designed for different kinds of e-services. Tickets provide a flexible mechanism and users can check charges at anytime.

- Untraceable electronic cash system;

  An untraceable e-cash system is developed, in which the bank involvement in the payment transaction between a user and a receiver is eliminated. Users remain anonymous, unless she/he spends a coin more than once.

- A self-scalable anonymity electronic payment system;

  In this payment system, from the viewpoint of banks, consumers can improve anonymity if they are worried about disclosure of their identities. Each consumer can get the required anonymity level.

- Using RBAC to manage electronic payment system;

  The basic structure of RBAC is reviewed. The challenge problems in the management of RBAC with electronic payment systems are analyzed and how to use RBAC to manage electronic payment system is proposed.

- The investigation of recovery algorithms for conflicting problems in user-role assignments and permission-role assignments.

  Formal authorization allocation algorithms for role-based access control have developed. The formal approaches are based on relational structure, and relational algebra and are used to check conflicting problems between roles and between permissions.

Publications Based on this Thesis

1. H. Wang, Y. Zhang, J. Cao and V. Varadharajan, Achieving Secure and Flexible M-Services Through Tickets, In B. Benatallah and Z. Maamar, editor, Special issue on M-Services. IEEE Transactions on Systems, Man, and Sybernetics. (Minor revisions)

2. H. Wang, Y. Zhang, J. Cao, Y. Kambayahsi, A Global Ticket-Based Access Scheme for Mobile Users, In Amjad Umar, editor, Special Issue on Object-Oriented Client/Server Internet Environments, Information Systems Frontiers, Vol. 5, No. 3, 2003.

3. H. Wang, J. Cao, Y. Zhang, An Electronic Payment Scheme and Its RBAC management, Concurrent Engineering: Research and Application Journal (To appear).

4. H. Wang, J. Cao, Y. Zhang, A Flexible Payment Scheme and Its Permission-Role Assignment, In Michael Oudshoorn, editor, Proceedings of the Twenty-Sixth Australasian Computer Science Conference (ACSC2003), Feb. 2-7, 2003, Adelaide, Australia. Vol. 25, No. 1, pages: 189-198.

5. H. Wang, J. Cao, Y. Zhang, Formal Authorization Approaches for Permission-Role Assignment Using Relational Algebra Operations, In Klaus Dieter-Schewe and Xiaofang Zhou, editor, Proceedings of the 14th Australasian Database

Conference (ADC2003), Feb. 2-7, 2003, Adelaide, Australia. Vol. 25, No.1, pages:125-134.

6. H. Wang, J. Cao, Y. Zhang, A Flexible Payment Scheme and Its User-Role Assignment, In Alvin Chan etc. editor, Book "Cooperative Internet Computing", pages: 107-128, published by Kluwer Academic Publisher, Dec. 2002.

7. H. Wang, J. Cao, Y. Kambayashi, Building a Consumer Anonymity Scalable Payment Protocol for the Internet Purchases, 12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems, Feb. 25-26, 2002, San Jose, USA.

8. H. Wang, J. Cao, Y. Zhang, Formal Authorization Allocation Approaches for Role-Based Access Control Based on Relational Algebra Operations, In T. Ling etc., Editor, Proceedings of the 3rd International Conference on Web Information Systems Engineering (WISE'2002), Dec. 3-6, 2002, Singapore, IEEE, pages: 301-312.

9. H. Wang, J. Cao, Y. Zhang, A Flexible Payment Scheme and Its Role-based User-Role Assignment, In A. Chan etc., Editor, Proceedings of the second International Workshop on Cooperative Internet Computing (CIC2002), August 18-19, 2002, Hong Kong, pages: 58-68.

10. H. Wang, J. Cao, Y. Zhang, Ticket-Based Service Access Scheme for Mobile Users, In Michael Oudshoorn, editor, Proceedings of the Twenty-Fifth Australian Computer Science Conference (ACSC2002), Jan. 28-Feb. 2, Melbourne, Australia. Vol. 4, No.1, pages: 285-292.

11. H. Wang, J. Cao, Y. Zhang, A Consumer Anonymity Scalable Payment Scheme With Role Based Access Control, In T. Ozsu, etc., editor, Proceedings of the 2nd International Conference on Web Information Systems Engineering (WISE'2001), IEEE Computer Society, pages: 53-62. Dec. 3-6, 2001, Kyoto, Japan.

12. H. Wang and Y. Zhang, Untraceable Off-line Electronic Cash Flow in E-Commerce, In Michael Oudshoorn, editor, Proceedings of the 24th Australian Computer Science Conference ACSC2001, IEEE Computer Society, pages: 191-198, Gold Coast, Australia. Feb. 2001.

13. H. Wang and Y. Zhang, A Protocol for Untraceable Electronic Cash, Proceedings of the First International Conference on Web-Age Information Management, Springer-Verlag, Vol. 1846, pages:189-197, Shanghai, China, 2000.

14. H. Wang, T. Duan. A Signature Scheme for Security of E-Commerce. International Conference for Information Security, Computer Engineering. Vol. 25, pages: 79-80, 1999.

15. H. Wang, H, Ming. A MultiSignature Scheme for Integrity of Database. The Fifth International Conference for Young Computer Science, International Academic Publisher, pages: 627-630, Nanjing, P. R. China, 1999.

Contents

List of Figures

List of Tables

xxx

Chapter 1

Introduction

This chapter presents the motivation and objectives of this thesis. There are three sections in the chapter. In the first section, three technology issues in e-commerce with their unsolved problems are introduced. The issues are ticket based access schemes, electronic payment and role-based access control management. The objectives and organization of this thesis are described in the second and third section respectively.

1.1 Overview and Motivation

1.1.1 Mobile service system

E-commerce is revolutionizing the way we work. Its impact is already being felt in consumer goods sales and will be much more widespread in the future. Mobile service systems, as an important subject, extend usages of e-commerce [2]. Mobile commerce is a subset of e-commerce which continues to see phenomenal growth, but so far most e-commerce involves wired infrastructures. Emerging wireless and mobile networks create new opportunities in e-commerce and increased growth. A *mobile service system* (or *M-commerce*) is defined as any type of transaction of an economic

value that is conducted through a terminal that uses a wireless telecommunications network for communication with an e-commerce infrastructure.

During the early 1980s, cellular telephone systems experienced rapid growth in Europe, particularly in the United Kingdom and Germany. Each country developed its own system, which was incompatible with everyone else's in equipment and operation. This was undesirable, because not only was the mobile equipment limited to operation within national boundaries, but there was also a very limited market for each type of equipment, so economies of scale and the consequent savings could not be realized [27].

In 1983, a group called the Global System Mobile (GSM) was setup for mobile unified standards. The first version of the GSM specifications were published in 1990. Commercial service was started in mid-1991, and by 1993 there were 36 GSM networks in 22 countries [97]. Over 200 GSM networks are operational in 110 countries around the world. In the beginning of 1994, there were 1.3 million subscribers worldwide [2], which has grown to more than 55 million by October 1997.

Mobile service systems are becoming extremely popular, which makes the provision of services to mobile users an attractive business area. Mobile service can be regarded as a special form of e-commerce, where users buy services instead of products from service providers via the network because services are much wider that products. From a technology perspective, mobile communication systems have been made possible by two factors: advances in wireless communications, and portable devices that are readily available on the market for decreasing costs. The first has resulted in a number of wireless access network technological telephones, such as, cellular telephone networks and wireless LANs (local area network). Examples of

the second are high performance laptop computers, handheld palmtop devices with considerable computing power, and portable sets with increasing functionality. From a business perspective, mobile service systems are possible because mobility is a big attraction for users. Mobile telephones, for instance, have gained more popularity among users than anyone predicted. In Europe, the number of mobile subscribers was 22 million in 1995 and was estimated to reach more than 110 million by 2000 [17]. In Finland and several other countries, the number of mobile subscriptions has already exceeded the figures for fixed telephone lines. The growing success and popularity of mobile communication systems makes the mobile services to be a benefit business area. Therefore, a great number and variety of service providers are expected to appear on the mobile market in the near future.

There are a number of proposals for mobile systems [68, 69, 42], though all of them lack flexibility in security management. The Global system for mobile communications [68], for example, provides mechanisms for user authentication as well as integrity and confidentiality, including protection of information exchanged between the mobile terminal and the fixed network. It provides, however, only limited privacy protection for users by hiding their real identities from eavesdroppers on the radio interface [69]. Another contemporary mobile communication system CDPD [42] provides similar security services, however, there are some other issues and problems in mobile commerce, which need to be addressed:

Global solution. Current solutions only solve particular service problems for mobile users. Users have to change the mobile service systems in order to do other business on the Internet. This is not convenient for users.

Clear charging. Mobile users wish to see a clear and continuously up-dated bill for provided services. Users do not like receiving an account statement only monthly

or bi-monthly, but like to be able to check it at anytime.

Trustworthiness. In most cases of buying services, we assume that mobile users trust service providers to bill their service usage correctly and not to misuse user and service usage related information. This kind of trust model is not adequate for future mobile communication systems. With the rapidly growing number of service providers, most of which are new on the market and unknown to the users, this assumption is no longer justified. This requires mechanisms that guarantee correct and indisputable billing and ensures anonymous service usage.

Scalability. Future mobile communication systems aim at offering access to any service, anywhere, at anytime. The mechanisms of current mobile communication systems are not sufficiently scalable to be able to fulfill this requirement. Traditional solutions for implementing user mobility rely on cross domain authentication and roaming agreements. A user, when visiting a foreign domain and accessing a service there, has to authenticate himself to the foreign service provider with the help of his home domain agent. This may involve a potentially time consuming authentication protocol over long distances. Furthermore, cross domain authentication requires the foreign service provider to trust the home domain agent of the user. Today, this trust is based on roaming agreements between various service providers. With the rapidly growing number of service providers, however, roaming agreements are becoming inefficient and no longer feasible. New mechanisms are needed that do not require contact with the home domain of the user when accessing services in a foreign domain, nor business agreements between domains.

In the future, mobile service systems should provide global solutions for all kinds of mobile services and guarantee higher levels of security than current systems. This means that, as well as requiring confidentiality and the protection of the integrity of

message exchanged between the user and the service provider, and authentication of the user to the service provider, future systems should also require authentication of the service provider to the user and guarantee the user higher levels of privacy. Furthermore, clear billing has to be ensured.

A new approach is needed to address the above-mentioned problems. This approach is based on the Credential Centre, and a ticket-based mechanism for service access [119]. The main idea is illustrated in Figure 1.1.

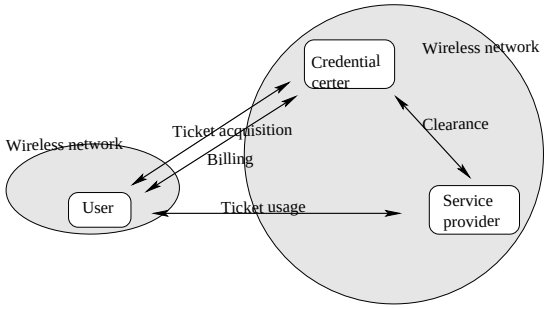

Figure 1.1: Ticket Model

In the first step, the Credential Centre issues tickets for the users. In the second step, a ticket-based mechanism is implemented allowing the user to remunerate the service providers. Tickets can provide a flexible and scalable mechanism for mobile access [115].

There are three participants (User, Service provider, and Credential Centre) and a protocol with several sub protocols (ticket acquisition, ticket usage, clearance, and billing) in the model. Each user is registered with the Credential Centre. The user obtains tickets by running the ticket acquisition protocol with the Credential Centre. These tickets are used to access services anonymously. In the ticket usage protocol, the user presents an appropriate ticket to the service provider, which can verify the

validity of the ticket and the legitimacy of the user to use it. While the user's private identity is not revealed in this protocol, the service provider authenticates itself to the user. If the verification of the ticket is successful, then the service provider provides the service to the user according to the conditions on the ticket. Based on the received tickets, the Credential Centre prepares a clear bill for each user.

We introduce a ticket based access scheme in Chapter 3. The main contributions of the ticket access scheme are:

1. It is a global ticket-based solution.
2. It is also scalable and users can check their account statement at anytime.
3. It is an anonymous and dynamic system, and new users and new service providers can join the system at anytime.

1.1.2 Electronic payment

After providing a service, the service provider will ask consumers to pay. There are a number of proposals for electronic payment. David Chaum [22] first proposed an on-line payment system that would guarantee that valid coins are received. This system provides some levels of anonymity against a collaboration of shops and banks. However, users have no flexible anonymity and banks have to keep a very large database for users and coins. Another on-line CyberCoin (http://www.cybercash.com) approach allows clients to make payments by signing fund transfer requests to merchants. The merchants submit the signed requests to the bank for authorization of the payments. The CyberCoin protocol, however, is not fully anonymous since it allows the issuing bank to track every purchase. Furthermore, the scalability of the CyberCoin protocol is questionable since it relies on the availability of a single on-

line bank. NetBill [30] extends the above payment mechanism by supporting goods atomicity and certified delivery. The drawbacks of NetBill protocol are the addition of extra messages and the significant increase in the amount of encryption used. In 1995, Chaum [23] proposed blind signatures to provide a fully anonymous coin-based payment system. This system has the disadvantages of centralized management of issuing and checking the double spending of coins. The most sophisticated protocol is the SET protocol [67], which was designed to facilitate credit card transactions over the Internet. SET security comes at a considerable computation and communication cost. SET, unlike other simpler on-line protocols, does not offer full anonymity, non-repudiation, nor certified delivery.

The systems mentioned above are on-line payment systems. The qualifier "on-line" means banks have to connect with service providers for verifications. They need sophisticated cryptographic functions for each coin, and require additional computational resources for the bank to validate the purchases. Forcing the bank to be on-line at payment is a very strict requirement. On-line payment systems protect the merchant and the bank against customer fraud, since every payment needs to be approved by the customer's bank. This increases the computation cost, proportional to the size of the database of spent coins. If a large number of people start using the system, the size of this database could become very large and unmanageable and furthermore keeping a database of every coin ever spent in the system is not a scalable solution [23]. Digicash [23] plans to use multiple banks each minting and managing their own currency with inter-bank clearing to handle the problems of scalability. It seems likely that the host bank machine has an internal scalable structure so that it can be set up not only for a 10,000 user bank, but also for a 1,000,000 user bank. Under these circumstances, the task of maintaining

and querying a database of spent coins is probably beyond today's state-of-the-art database systems [108, 109].

In an off-line protocol, the merchant verifies the payment using cryptographic techniques, and commits the payment to the payment authority later in an off-line batch process. Off-line payment systems were designed to lower the cost of transactions due to removing the delay in verifying batch processes. Off-line payment systems, however, suffer from the possibility of double spending, whereby the electronic currency might be duplicated and spent repeatedly.

The first off-line anonymous electronic cash was introduced by Chaum, Fiat and Naor [26]. The security of their scheme relied on some restricted assumptions such as a function $f(x, y)$ is similar to random oracle and g gives a one-to-one map from the second argument onto a special range. There is also no formal proof attempted. Although hardly practical, their system demonstrated how off-line e-cash can be constructed and laid the foundation for more secure and efficient schemes. In 1995, Chan, Frankel and Tsiounis [20] presented a provable secure off-line e-cash scheme that relied only on the security of RSA [86]. This scheme extended the work of Franklin and Yung [44] who aimed to achieve provable security without the use of general computation protocols. The anonymity of consumers is based on the security of RSA and it cannot be changed dynamically after the system is established. NetCents [81] proposed a lightweight, flexible and secure protocol for micropayments of electronic commerce over the Internet. This protocol is designed only to support purchases ranging in value from a fraction of a penny and up. In 2000, David Pointcheval [79] presented a payment scheme in which the consumer's identity can be found any time by a certification authority. So the privacy of a consumer cannot be protected.

A new off-line electronic cash scheme is needed, in which the anonymity of consumers is scalable and can be done by consumers themselves. Consumers can get the required anonymity without showing their identities to any third party.

We focus on efficient E-cash systems and consumer anonymity self-scalable payment schemes for the Electronic payment topic.

We have developed a new untraceable electronic cash scheme for transaction [108]. No banks are involved in payment transactions between users and receivers (for example shops). Users withdraw electronic "coins" from banks and use them to pay to receivers. The receivers subsequently deposit the coins back to the bank. In the process users remain anonymous, unless she/he spends a single coin more than once (double spend). The security of the system is based on DLA (Discrete Logarithm Assumption) and the cut-and-choose methodology.

We have designed a consumer anonymity self-scalable payment scheme [110]. This scheme includes two basic processes in system initialization (bank setup and consumer setup) and three main protocols: a new withdrawal protocol with which a user U withdraws electronic coins from a bank B while his account is debited; a new payment protocol with which U pays the coin to a shop S; and a new deposit protocol with which S deposits the coin to B and has his account credited. If a consumer wants to get a high level of anonymity after she/he has obtained a coin from the bank (withdrawal), she/he can contact an anonymity provider agent (AP).

Basic payment model: In the simplest form of a payment model, an e-cash system consists of three parts (a bank B, a consumer U and a shop S) and three main procedures as shown in Figure 1.2 (withdrawal, payment and deposit). In a coin's life-cycle, the consumer U first performs an account establishment protocol

to open an account with the bank B.

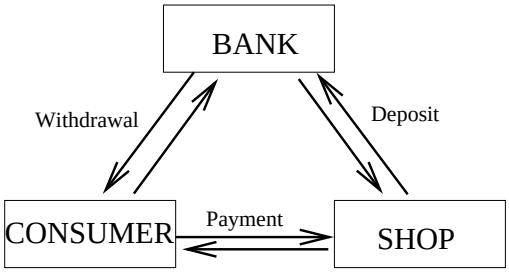

Figure 1.2: Basic Cash Transaction

Besides the basic participants, a third party named Anonymity Provider (AP) agent is involved in our scheme. The AP agent ensures the consumer's required level of anonymity and is not involved in a purchase process. The AP agent gives a certificate to the consumer when she/he needs a higher level of anonymity.

The new payment scheme has the following features:

1. Consumers can get a higher level of anonymity by themselves.
2. The identity of a consumer cannot be traced unless the consumer spends the same coin twice.
3. It is an off-line scheme with low communication and computation.
4. It can effectively prevent eavesdropping, tampering, impersonation and "perfect crime" [99].

1.1.3 Role based access control

To reduce administration cost and complexity and to improve the security of management, we analyze how to use Role based access control (RBAC) to manage the

new systems. Recently, role based access control (RBAC) has been widely used in databases system management and operating system products.

RBAC is described in terms of individual users being associated with roles as well as roles being associated with permissions (each permission is a pair of objects and operations). As such, a role is associated with users and permissions. A user in this model is a human being. A role is a job function or job title within the organization associated with its authority and responsibility.

A permission is an approval of a particular operation to be performed on one or more objects. The relationship between roles and permissions is shown in Figure 1.3, and arrows indicate a many to many relationship (that is , a permission can be associated with one or more roles, and a role can be associated with one or more permissions). The RBAC security model has two components: MC_0 and MC_1. Model component MC_0, called the RBAC authorization database model, defines the RBAC security properties for authorization of static roles. Static properties of a RBAC authorization database include role hierarchy, inheritance, cardinality, and static separation of duty. MC_1, called the RBAC activation model, defines the RBAC security properties for dynamic activation of roles. Dynamic properties include role activation, permission execution, dynamic separation of duties, and object access. In particular, the RBAC model supports the specification of:

1. User/role associations, that is , the constraints specifying user authorizations to perform roles;

2. Role hierarchies, for example, the constraints specifying which role may inherit all of the permissions of another role;

3. Duty separation constraints; these are role/role associations indicating conflict

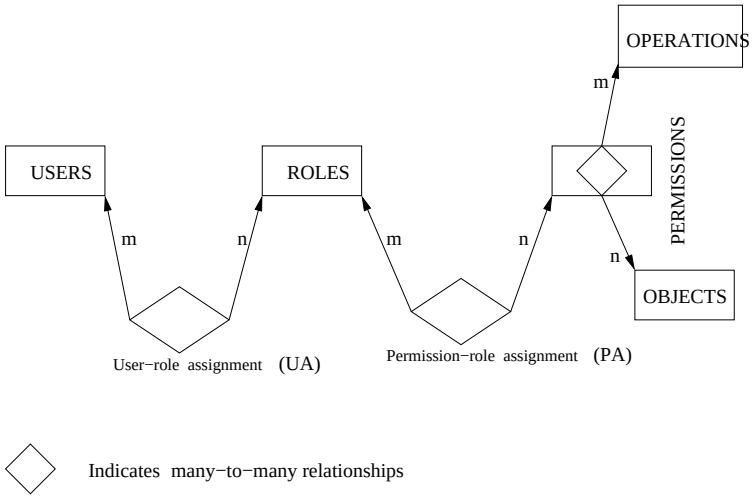

Figure 1.3: Basic RBAC relationship

of interest:

(a) Static separated duty (SSD); a constraint specifying that a user cannot be authorized for two different roles,

(b) Dynamic separated duty (DSD); a constraint specifying that a user can be authorized for two different roles but cannot act simultaneously in both.

4. Cardinality; the maximum number of users allowed, that is , how many users can be authorized for any particular role (role cardinality), for example, only one manager.

Properties 1 and 4 depend on how a system is implemented, and can be decided after the system has been designed. However, properties 2 and 3 have to be decided when a system is designed. This is because permissions of different roles may be in conflict compromising the security of the system.

There has been little research done on the usage of RABC in electronic service systems. Methods of use RBAC to manage electronic payment system is another

challenge in this project [91]. We have analyzed RBAC and its management for electronic service systems in chapters 6, 7 and Chapter 8.

1.2 Objectives of the Thesis

With advances in computer networks, in processor speed, and in databases, and with advances in note counterfeiting technology and with both individuals' and businesses' desire for remote and more convenient financial transactions, some forms of electronic cash are likely to become widespread within 5 to 10 years. Although unconditionally anonymous electronic cash systems have been proposed in the literature, financial institutions are unwilling to be a completely anonymous system. Their reasons for opposing complete untraceability have to do with the containment of user fraud and the desire to restrict new kinds of crime that unrestricted, and spendable electronic cash could facilitate. Because of the necessary concern over crime control, they have previously proposed systems with little or no protection for the users' privacy. On the other hand, consumers, when shopping over the Internet, prefer to have some degree of anonymity so that, for example, they cannot be traced by banks. There are a number of proposals for electronic cash systems [22, 30, 67, 26, 20, 81, 79]. All of these solutions are either too large to manage or lack flexibility in providing anonymity. Therefore, they are not suitable solutions for electronic payment in the future.

Electronic service systems allow users and service providers to be widely distributed and to use heterogeneous catalog systems. It may have conflicts within operations which may cause negative influences with the service systems. The management of these service systems is complex. Whether or not systems are successful depends on the quality of their management.

To investigate these problems, I will focus on three major tasks in my PhD research:

1. A ticket based access solution;
2. Electronic payment systems;
3. Role based access control.

1.3 Organization of the Thesis

The thesis consists of nine chapters. Their precedence order is outlined and illustrated in Figure 1.4.

In Chapter 2, e-commerce issues and a framework for e-commerce are presented. There are three dimensions in the framework: the software technology dimension, the hardware support dimension, and the application dimension. From Chapter 3 to Chapter 8, we focus on the software technology plane. In Chapter 3, a ticket based access scheme for mobile e-commerce is introduced. An electronic cash scheme is analyzed in Chapter 4 while a consumer scalable anonymity electronic payment is designed in Chapter 5. RBAC and its applications are discussed in Chapter 6. There are consistency problems that may arise in user-role assignment and permission-role assignment with RBAC. In Chapter 7, we address the problem in user-role assignment by using formal authorization allocation (FAA) approaches. In Chapter 8, we develop formal authorization allocation (FAA) approaches to solve the problem in permission-role assignment. Finally, conclusions and future work are indicated in Chapter 9.

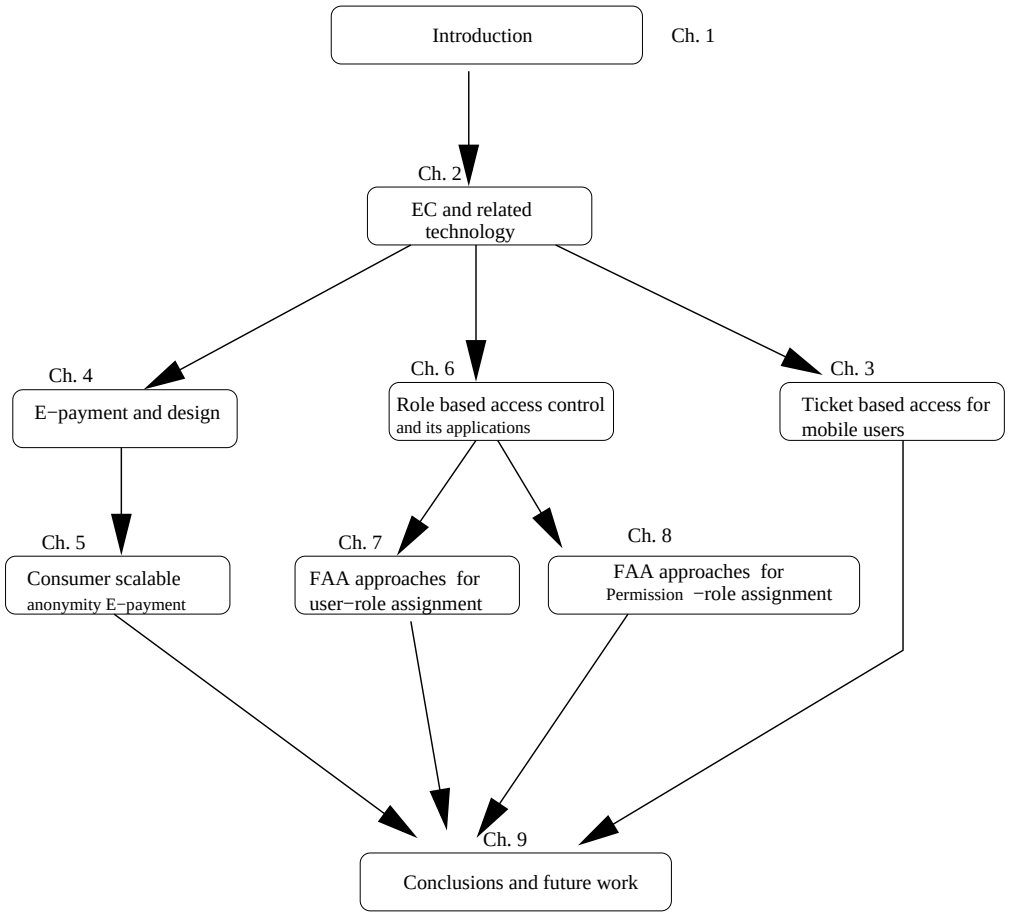

Figure 1.4: The structure of the thesis

Chapter 2

Electronic Commerce Items and Related Technology

This chapter presents e-commerce items and a framework for e-commerce. The items include trust and privacy, payment system, management and security and so on. The position and the importance of each topic in e-commerce are discussed extensively. Finally, a new framework for e-commerce is proposed. The framework has three dimensions which are software technology dimension, hardware dimension and e-commerce application dimension. The criteria of a good framework is also indicated. It has been demonstrated that the framework is a natural and efficient framework. Furthermore, all items in e-commerce can fit into the framework very well.

2.1 Introduction

E-commerce can be defined as the buying and selling of information, products, and services via computer network [58]. E-commerce systems can be of significant value as a milestone for new customer access management strategies. This is because e-commerce systems:

1. directly connect buyers and sellers;
2. support fully digital information exchange;
3. have no time and place limits;
4. support interactivity and therefore can dynamically adapt to customer behavior;
5. can be updated in real-time, therefore always up-to-date.

Next, I describe various items in e-commerce.

2.2 Items in Electronic Commerce

In this section, some items with technological details on the way to electronic commerce are reviewed. These items are important to understanding e-commerce.

2.2.1 Trust and privacy

Users are seriously concerned with privacy and trust, which may lead to a backlash against providers using such systems, or customers avoiding the use of these systems [29]. Some companies require customers to provide their information on their demographics information, buying patterns or product needs. Unfortunately, this data is critical in many cases since consumers do not know what will be done with their private data. There are two ways of handling these concerns, either customers can be made aware of the benefits of volunteering this data, or material incentives can be offered to customers to attract them.

The concerns with trust can be categorized as follows:

1. Personal data, whether personal data is secure or not during electronic transactions;

2. Service processes, whether service processes can be trusted or not during electronic transactions with organizations. The forms of service process involve:

    a. money paid and received;

    b. goods and services offered and acquired; and

    c. assurances that a refund is available for unsatisfactory goods and services;

3. Privacy of personal data, whether data storage of personal data can be trusted or not.

4. Subsequent behavior, whether has subsequent behavior with customers data to other companies (that is, sale customers data or share the data with sub companies and so on.)

The concerns with privacy includes:

1. Cookies, websites send cookies (files) to consumers who visit the websites and also get some information of consumers;

2. Internet Privacy: Cyberspace Invades Personal Space [29];

3. Spam, consumers receive a lot of useless messages.

No consumers like their private data to be reviewed. Electronic services have to protect the privacy of users, and then users trust and be interested in the e-commerce system.

2.2.2 Electronic payment systems

Electronic business can only be successful if financial exchanges between buyers and sellers can occur in a simple, universally accepted, safe and cheap way. E-commerce assumes that the participants pay for the services they receive. But there has been a marked reluctance among net-users to actually part with their money, particularly for digital goods and services. Various systems have been proposed, some of them based on traditional mechanisms (for example, credit cards accounts) while others rely on new designs, such as electronic money. The key here is to find a few widely accepted mechanisms, which can be used by most participants. Two later chapters address the questions of how to design a secure electronic payment system and how to provide a scalable anonymity payment scheme for consumers.

2.2.3 Security

Security is a key enabling factor in e-commerce. With the industry moving toward a consensus on providing service on wireless commerce, the next major challenge for enterprisers and service providers is securing resources from unauthorized access and preventing fraud. As companies allow customers to execute wireless transactions and business partners with wireless access to share information and resources via an Intranet or extranet, security becomes a chief concern. There are three levels for e-commerce security [64, 71, 37]. The following Figure 2.1 illustrates users access services in e-commerce. Three levels are involved in the figure. They are network connection level, management level, and transaction level.

Network connection level

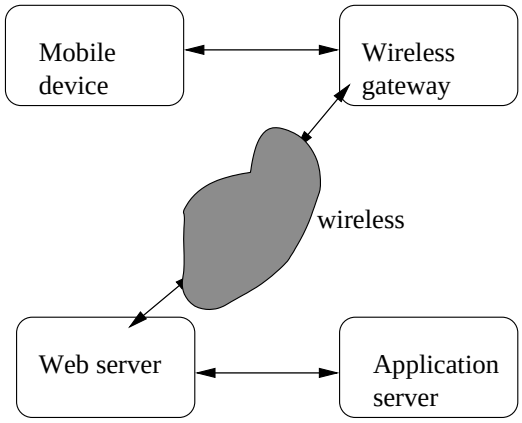

Figure 2.1: Users access service model

In a wired network, firewalls are used to provide the first level of security between the user and the Web server; in the wireless world, mobile gateways manage access to a Web server, provide encryption through the Wireless Transport Layer Security (WTLS) specification and authenticate users to enable a secure connection between the wireless device and the application server. WTLS, which is a version of TLS/SSL for wireless communications, provides secure service such as data integrity, authentication, and denial-of-service protection. Service providers in e-commerce environments must address the same challenges involved in securing wired environments and then users and service providers can trust each other. It means that users are controlled to access both the network and individual resources (applications, content and transactions).

Management level

When users log in the wireless network, a management system is needed to control which resources users are authorized to access and which transactions they can execute. This management system must also audit a user's actions. This requires an extensible infrastructure that can integrate with complementary security and e-commerce technologies. For example, it should support multiple authentication

methods including Personal Identification Numbers (PINs), passwords and Public Key Infrastructure (PKI). A security infrastructure should provide an administration model to support the much higher volume of users associated with wireless applications. Ideally, a wireless access management system should use different roles to protect individual resources and control user access. Systems that use access control lists (ACLs) would require that an administrator manually move the user to a new ACL for every change in status, which is an inefficient and expensive approach. The role based access control (RBAC) provides a new method to easily manage a complex system. We detailed introduce RBAC in the later chapters.

Transactions level.

Consumers are all comfortable with conducting transactions face-to-face; there is a physical exchange of goods and payment using a trusted mechanism such as cash, cheque or card. The comfort derives in part from familiarity. Many people are also familiar with, and trust, mail order transactions by post or over the phone, even though there are many opportunities for failure. In an electronic world trust is a more abstract concept:

1. Trust the information in a website?

2. Trust credit card information to the site?

3. Trust goods will be delivered?

These require that sensitive data must be secured throughout the transmission, and all transactions must be confirmed. It is necessary to authenticate a user's identity, authorize the transaction, log the transaction details, and generate a digital receipt. By logging all wireless user activity from accessing portfolio information and

applying for loans – organizations can ensure that all transactions are binding and provide customers with detailed transaction reports.

With the high progress of wireless technology, M-commerce has been becoming an important subset of e-commerce. The issues mentioned above are also existed in M-commerce.

2.3 A framework for electronic commerce

E-Commerce concerns with using a handheld terminal and mobile terminal such as a mobile phone, connecting with wired and wireless networks, and conducting transactions. Because there are many different kind of e-commerce Services (e-shop, e-bank, M-service, and so on), it is necessary to build a framework to organize them so that some conceptual structures can be discovered and new services may be compared meaningfully with existing ones along some uniform dimensions. In order to build such a framework, this section propose three dimensions. This integrated three-dimensional framework help users to understand the development status of current e-commerce services and further help designers, developers, and researcher to strategize and effectively implement new e-commerce applications. The three dimensions are:

1. E-commerce hardware dimension;
2. E-commerce software technology dimension;
3. E-commerce application dimension.

The first dimension of the framework is based on the hardware that is used within e-commerce such as communication server, computer terminal, mobile phone, lap

computer, and so on. In the second dimension, we provide various kinds of software technologies in e-commerce such as financial transactions, e-payment, management, and so on. In the third dimension, we categorize different e-commerce applications according to their value added features.

A framework for the field of e-commerce was introduced by Jeffrey F. etc [56]. There are four infrastructures in the framework; they are Technology, Capital, Media and Public Policy infrastructure. The authors specified the technology infrastructure is the foundation of a system. The hardware backbone of computers, routers, servers, modems, and other network technologies provides half of the technology issues. The other half includes the software and communication standards that run on the hardware, including the core protocol for the web. On the other hand, authors specified a Media infrastructure with communication technology. These two infrastructures are confusing by the communication technology. The main reason of this confused point is because the authors did not analyze the relationship between different infrastructures. Another framework proposed by Varshney and Vetter [105], shows a user plane with four levels and a developer-provider plane with three. The framework has several functional layers. Consumers, providers, designers, and so on can find individual layers in the framework. However, there are some shortcomings in this framework. For instance, there is a tight relationship between the user plane and the developer and provider plane, the authors do not address it well in their framework. Another disadvantage is that the authors put all mobile commerce applications into some classes among the application layer without addressing any relationship among them. Many services involved in this layer have different characteristics and relationships between them.

Criteria for Choosing a Framework

We discuss the criteria for choosing a framework in the context of e-commerce. A framework and its dimensions is a subjective work, there is no exact measure for the quality of a framework along with its dimensions. However, there are some criteria for choosing a framework's dimensions.

One criterion is that a framework should be natural, which means that the framework is easy to understand even to the novice. The next criterion is usefulness, which means that the framework either help customers in e-commerce to strategize e-commerce applications, or provide a model structure for researchers. The third criterion is that the framework should be sufficiently consistent in categorizing e-commerce applications. This means, all existing and possible future e-commerce applications can be included and fit well into the framework.

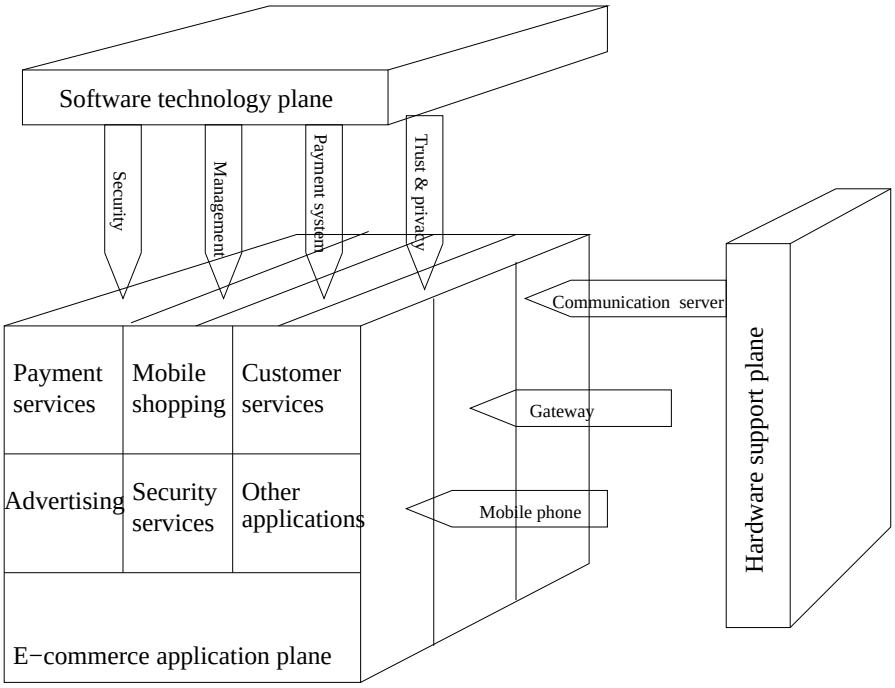

Figure 2.2: A framework of e-commerce

Framework and Dimensions

The framework proposed in Figure 2.2 has three dimensions: software technology dimension; hardware dimension and applications dimension. In this framework, the software technology dimension is faced with an outside layer that supports end-users and other third parties who make use of e-commerce services. These parties are not necessary players in the e-commerce arena. This is because many people such as researchers and technology supporters are not involved in e-commerce.

The hardware dimension is also supported by an outside layer. This layer refers to the hardware environment. Productions, for example, communication servers, mobile computers, gateway, and mobile firewall and so on. act as the cornerstone of e-commerce and are the basic requirements to implement an e-commerce.

The e-commerce application dimension can be considered as an application of the first dimension and the second dimension. Therefore we have positioned it on another dimension.

Dimension One: Software Technology in E-Commerce

There are many software technologies in e-commerce and new skills that are still being developed. Consumers and suppliers believe that good technologies can bring them benefit and can be used to build a trusted e-service. Therefore, the software dimension includes the basic and important technology requirements such as payment system, management, and so on in e-commerce. This thesis focus on some topics on this dimension.

Table 2.1 provides some examples of software technologies.

Software technologies	Description	Examples
Communication	It is used for connection between terminal users and servers	ray technology Secondary dimensiont communication
E-service	It provides access for the user on the mobile network, such as billing, helpdesk,	Google, yahoo
Management	It offers how to manage a e-service system and who can access what information in the system	MAC, RBAC
Software platform	Organize a platform to integrate different communications and services for e-commerce	ExpressQ 3.0 by Nettech Systems
Financial transaction	Handles the financial transactions	Banks and BEA
Middleware	Middleware is a layer of software that is used by application developers to connect their E-applications with different networks and operating systems without introducing awareness in the applications	Snapshot in Aberdeen

Table 2.1: Software Technology in E-Commerce

Dimension 2: Hardware Support in E-Commerce

This dimension has several levels from the client side to the server side. We have various hardware productions in e-commerce. For example, mobile phones or handheld computers are used in consumer side while computer servers provide service context in service provider's side. Gateways, between terminals and servers, are used to connect to each other. Table 2.2 shows some hardware in e-commerce.

Layers in dimension	Descriptions and example	Example
Terminals	mobile devices and terminal provides the client side functions in e-commerce applications.	Mobile phone Laptops
Computer server	It provides various information of service and users can choose what service they need	Nokia Mobile Server SDK 1.0, Yahoo Chat Service.
Gateway	It is used to connect from consumers to servers	InfoSync,

Table 2.2: Hardware in e-commerce

Dimension 3: E-Commerce Application

E-commerce application	Descriptions	Examples
e-shop	e-commerce extends ability to make transactions across time location and creates new, business opportunities.	Amazon.com Yahoo
Financial Services	It offers financial transactions and is a key issue in e-commerce environment	ebay e-pay
Security services	A terminal can function as a security device for gaining access to e-commerce	PKI systems Wireless PKI systems
Customer Services	It can be more economically to provide services	AvantGo, ebay
Advertising	It introduces production for companies and individuals	AdsOnWheel, AvantGo

Table 2.3: E-commerce application

This dimension is based on the unique characteristics of e-commerce that combine the advantages of electronic communications with existing e-commerce services. And these characteristics can also be looked upon as the key drivers for the increasing expanded e-commerce market [122, 127]. Table 2.3 lists these categories and characteristics with examples. This layer can be divided into several minor subsets according to different market segments: C2C, B2B, and B2C layer, and so on.

M-commerce can be defined as mobile variance of e-commerce, most of these categories have their counterpart in a wireless world.

From what we have discussed above, we can find that all of these three dimensions are good in the first criteria. The framework is natural and easy to understand. As to the criteria of consistency and fitness issue, we have shown that it can strategize e-commerce application and all existing and future applications are included in the framework and fit well in these three dimensions.

2.4 Conclusions

With the accelerating progress of content presentation standards and continuing advances in data transmission speeds, e-commerce is assured by both consumers and business. An overview of e-commerce has been presented in this chapter. Various kinds of e-commerce issues such as mobile devices, e-payment, security and management, and so on are discussed. Finally, a new framework of e-commerce is also presented. Some technology topics in the software dimension such as payment system and management are detail analyzed in the rest of the thesis.

Chapter 3

A ticket-based access scheme for mobile users

This chapter presents a ticket-based access model for e-commerce, specially for mobile services. A ticket is a piece of information that represents the rights of a user to access a service. The model supports efficient authentication of users, services and service providers over different domains. Tickets are used to verify correctness of the requested service as well as to direct billing information to the appropriate user. The service providers can avoid roaming to multiple service domains, only contacting a Credential Centre to certify the user's ticket since tickets carry all authorization information needed for the requested services. The user can preserve anonymity and read a clear record of charges in the Credential Centre at anytime. Furthermore, the identity of misbehaving users can be revealed by a Trusted Centre.

The information in this chapter is based on a published paper [119].

3.1 Introduction

With recent advances in wireless computing and communication, mobile services are becoming an important factor in business. As a result, the security and privacy issues

in mobile systems have become more critical. The static security access control is incompatible with dynamic mobile environments. Mobile service access across multiple service domains, and the traditional access mechanisms rely on cross-domain authentication using roaming agreements starting home location. Cross-domain authentication involves many complicated authentication activities when the roam path is long. This limits the future mobile applications. Normally, there can be three participants in a mobile service. These are users, service providers, and services. Some services bind users and service providers as well as services such as flight services, other services do not bind any participants, for instance by using cash in shopping services, everyone can use cash to buy anything in shops. Hence, depending on which parts are bound, there are different kinds of mobile services. However, there is no scheme to provide a solution for all kinds of mobile services. Users have to change mobile service systems if they want to do different kind of mobile services on the Internet. From the consumer's point of view, there is often a preference for a total solution for all kinds of mobile services, some degree of anonymity such as no more cross authentication, and a clear statement of account when shopping over the Internet.

In the future, mobile service systems should provide a global solution for all kinds of services and guarantee higher levels of security than current systems. It means that as well as being a global solution, protecting the integrity of the message exchanged between the user and the service provider, and authenticating the user to the service provider, future systems should also require authentication of the service provider to the user. Furthermore, clear billing has to be ensured.

In this chapter, a new approach to address the above-mentioned issues is proposed. This approach is based on a Trusted Centre, a Credential Centre and a

ticket-based mechanism for service access. The main idea is illustrated in Figure 3.1.

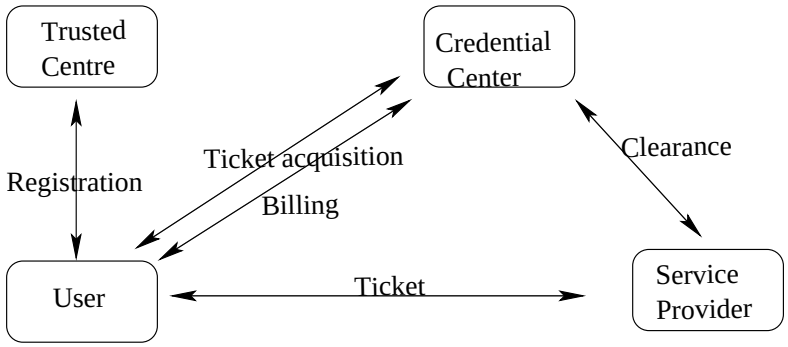

Figure 3.1: Ticket Model Structure

In this model, users, service providers and services are registered with the Trusted Centre. The Credential Centre issues tickets to its users. A ticket is a piece of information that represents the rights of a user to access a service provided by a service provider. Users can use these tickets to access services anonymously. When requesting a service, the user is required to hand over an appropriate ticket. After checking the ticket, the service provider provides the requested service to the user and reports to the Credential Centre. Later, the user can see a clear charging bill in the Credential Centre.

This chapter is organized as follows: in section 2, the basic ticket model and ticket types are introduced. There are eight different kinds of tickets that are divided into two groups, group_1 and group_2. A single signature scheme for ticket group_1 and its ticket usage are presented in section 3 while a multi-signature scheme for ticket group_2 and its usage are discussed in section 4. Security analysis and related work are described in section 5. How the scheme can survive the lossy wireless environments and can work for wireless service providers are also analyzed in this

section. Finally the conclusions are presented in section 6.

3.2 Basic ticket model

There are four participants (the user, the service provider, the Trusted Centre and the Credential Centre) and a protocol with several sub-protocols (ticket acquisition, ticket usage, clearance, and billing) in the ticket model. The user obtains tickets by running the ticket acquisition protocol. These tickets can be used to access services. The user presents an appropriate ticket to the service provider, which can verify the validity of the ticket. If the verification of the ticket is successful, then the service provider provides the service to the user according to the conditions on the ticket. Based on the received tickets, the Credential Centre prepares a charging bill for each user. The exact forms of the clearance (payment to the service provider) and billing (payment to the Credential Centre) protocols are not specified in our model. Readers may refer to [109] for details.

There are several advantages in using tickets for accessing services [17]:

Flexibility. Users can choose services as they need and buy an appropriate ticket that matches their personal requirements. They do not have to enter into long term contractual relationships with service providers.

Scalability. The information in tickets is used by a service provider to decide whether the service should be provided or not. Therefore, it is not necessary to run long distance protocols to perform authentication.

Privacy. Users only have to show tickets, they do not need to reveal their real identities. No private information is available to service providers.

Transferability. In real life, not all tickets can be transferred. It is not convenient for users to limit the wide use of tickets. In our ticket-based service access mechanism, a ticket can be lent to other users even though it is bound with the user. This means the ticket buyer and the ticket user do not have to be the same.

In addition to these advantages, some security problems such as duplication, forgery, and modification must be solved in order to implement a ticket system [82].

Duplication. There are two types of duplication that need to be considered. The first type is that users either use or sell a ticket many times (similar to double spending in electronic cash systems). The second type is an eavesdropper who listens to someone else acquiring a ticket and makes a copy for himself.

Forgery. The illegal construct of a valid ticket, which can be used for accessing to services.

Modification. Users must not modify tickets. This is to prevent users from accessing resources for which the tickets have not permitted, for example, a ticket that allows travelling by a bus, should not be modifiable to allow travelling by a flight.

A ticket may bind a given user, a given service, and a given service provider together. For example, a movie ticket, which usually does not specify who can use it (that is the user) or a travel card, which may not restrict the means of transport (that is the service). Based on this observation, there are eight types of tickets. These are illustrated in Table 3.1, where $'\Theta'$ means that the corresponding entity, user, service provider or service is bound by the ticket, while $'-'$ means that it is not.

A ticket of type t_0, for instance, does not restrict the service for which it can be used, the service provider which accepts it, or the user who can use it. This is much like cash in real life. The other extreme is a ticket of type t_7, which can only be used by a given user, for a given service, provided by a given service provider. An example of this type is a flight ticket.

Types	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
user	-	-	-	-	Θ	Θ	Θ	Θ
provider	-	-	Θ	Θ	-	-	Θ	Θ
service	-	Θ	-	Θ	-	Θ	-	Θ

Table 3.1: Ticket types

As mentioned in Table 1, tickets t_1, t_2 and t_4 have only one entity bounded and tickets t_3, t_5, t_6 and t_7 have two or three entities bounded. The tickets can be divided into two groups, one is ticket group_1 including tickets t_1, t_2, t_4, and another one is ticket group_2 including t_3, t_5, t_6, t_7. We design different mechanisms related to each ticket group. Users are anonymous in purchasing since no private message needs to be shown to service providers. Use of a ticket-based system can avoid roaming multiple service domains. A simple case is a single signature. This case can be used in tickets with only one bound entity (users, service providers or services). As a signer, the bound entity uses a signature to authenticate a ticket. To cope with the cases of two or more bound entities, it is extended to $v(v = 2, 3)$ Signers (multi-signature). This means that a user can get a service if all v entities agree. The v Signers case can also associate with the other services provided by many cooperative providers since the number v is not limited to 2 or 3. A Credential_role in the Credential Centre is set up to issue tickets and control the user's charging bill, and a Trusted_role in the Trusted Centre is also set up to judge conflicts. Each

user's statement of account can be seen clearly in the Credential Centre.

Through the usage of tickets, the problems of lack of Trust and Scalability are also addressed as follows:

Trust. Users can anonymously access services by using tickets. They need neither to reveal their identities, nor to fully trust service providers to handle user and service usage related information. On the other hand, the information of service providers is bound in tickets, thus, the user can assure that the service is provided by the selected service provider. Therefore, users and service providers can trust each other. Service providers can verify the validity of the tickets and check if their legitimate users used them. If necessary, anonymity can be revoked and the Trusted Centre can trace users who behave in a malicious way.

Scalability. The service providers only need to verify the ticket. Users do not require long distance protocols but connect to the Credential Centre. They acquire the ticket from the Credential Centre before roaming into the foreign domain.

For example, a user with a ticket for accessing information in a website can visit the information. The user does not need to show his/her identity to the website but the ticket. There is no long distance authorization protocol between the user and the website since the ticket includes all required authorization message.

In the remaining sections, we discuss how the protocol works for various kinds of tickets. We are not interested in ticket t_0 since it does not bind any entities and electronic cash can be instead of it.

3.3 Single signature scheme for ticket group_1

To facilitate discussions, some well-known primitive cryptographic terminologies, which are used in the remaining of the chapter, are reviewed.

Hash function, $h(x)$ is a hash function. For a given Y it is computationally hard to find a x such that $h(x) = Y$, where x might be a vector.

Hash functions have been used in computer science for a long time. A hash function is a function, mathematical or otherwise, that takes a variable-length input string and converts it to a fixed-length (generally smaller) output string. Regardless of the nature of the function, an adversary can always select values at random from the domain of the function in the hope that they hash on the same value. Counter-intuitively, it can be shown that if the range of a hash function is of size n, a guessing algorithm does not need to perform 2^{n-1} iterations (on average) in order to find a collision, but rather only $O(2^{n/2})$. Currently, a range of 160 bits is considered to be sufficient for most applications.

Hash functions are a major building block for several cryptographic protocols, including pseudorandom generators [8, 9, 14], digital signatures [18], and message authentication [106].

RSA, is a public key cryptosystem that offers both encryptions and digital signatures (authentication) [86]. RSA works as follows: taking two large primes p and q, and computing their product $n = pq$; n is called the modulus. Choosing a number e, less than n and relatively prime to $(p-1)(q-1)$. Finding another number d such that $(ed-1)$ is divisible by $(p-1)(q-1)$. The public key is the pair (n, e), the private key is d. The factors p and q may be kept with the private key or destroyed.

It is currently difficult to obtain the private key d from the public key (n, e). RSA is often used in modern environments [31], especially on the Internet, since an individual needs not send any private secret key to others when they want to contact him.

Multi-signatures, are multiple signatures signed on the same document. There are two ways to implement multi-signature. One is that each person signs separately, the other is that the message is signed simultaneously [102]. A multi-signature is the enhancement of a single signature.

This section introduces a single signature scheme for tickets t_1, t_2, t_4. There are four roles in the single signature scheme, Signer, Verifier, Credential_role and Trusted_role. Depending on tickets, the Signer can be a user, service or service provider that signs a signature as a ticket. The Verifier might be a user or service provider that verifies the signature of the Signer. The Credential_role in the Credential Centre issues tickets. It provides information for the Verifier to check the signature. Whether the signature is valid or not depends on the information. The Trusted_role is a judge to solve the conflict between users, service providers and services. This is because only the Trusted_role has the secret key of the system and can trace users and service providers. Each Signer has a different but fixed identity I, which is validated once the Signer is registered in the Trusted Centre and does not include any private message of the Signer. Ticket t_4, for instance, is bound to a user only. A user can follow this scheme to sign a signature as a ticket, the service provider verifies it and then sends some information to the Credential_role and asks for payment. Tickets t_1, t_2 are similar to ticket t_4, the signers are service provider and service separately but not users.

The outline of the process in the scheme is shown in Figure 3.2. In the system

initialization, the Trusted_role sends the private messages (r, S) to the Signer when the Signer I is set up, where r, S are computed by the Trusted_role, r is used in the first verification by the Credential_role and S is used as the first signature key by the Signer. In the second step, the Credential_role verifies if the data (I, r, D) sent by the Signer are valid or not, where D is used in the ticket verification. The data (I, D) are put on a public directory in the Credential Centre if the data are valid. At this time, the Signer can do a signing message job.

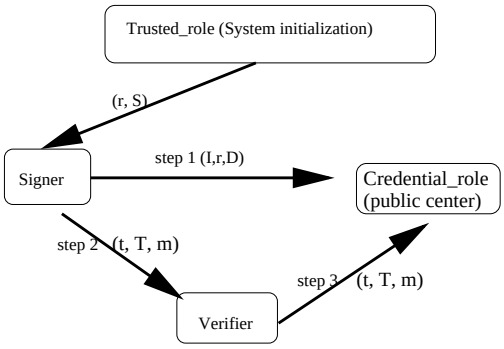

Figure 3.2: Single signature scheme for ticket group_1

While the Signer signs a message m, the Signer sends the signed message (t, T, m) as a ticket to the Verifier, and the latter checks if it is true or not, where t and T are computed by the Signer and m may include some service information and conditions etc. The data (I, D) in the Credential Centre are needed. The Verifier cannot verify the message when the data (I, D) in the Credential Centre are not correct. Then the Credential_role can control the usage of the ticket, and even find who the Signer is if it contacts the Trusted_role. In the final step, the Verifier sends a message which including the ticket to the Credential Centre while the ticket is true. The latter updates the data (I, D) that is used to issue a charging bill. The data (I, D) is changed while the ticket is used and the ticket is invalid if the verifier cannot get

the correct data (I, D). Thus, the ticket cannot be used twice and the user can see a clear statement.

3.3.1 Initialization of the system

There are two components in a signature scheme, one is the Signer played by consumers (users), service providers, or services; the other is the Verifier played by consumers or service providers. As a ticket, a signature is valid only if its verification is passed.

The Trusted_role computes a public composite modulus $n = pq$ where factors are strong primes. The Trusted_role chooses also prime exponents e and d such that:

$$e * d = 1 \ (mod\, \phi(n)).$$

Where $\phi(n) = (p-1)(q-1)$. The pair (n, e) are made public, and d is kept secret by the Trusted Centre as the system key. The Trusted_role computes when the Signer with identity I signs up:

$$r = k^e \ (mod\, n), \quad S = k * I \ (mod\, n)$$

where $k \in_R Z_n$ ($a \in_R A$ means that the element a is selected randomly from the set A with uniform distribution). Then

$$S^e = r * I^e \ (mod\, n).$$

Let $D = S^e \ (mod\, n)$. The Trusted_role sends (r, S) with PGP cryptographic technology to the Signer whose public identity is I. S is used as the first signature key to issue a ticket. Obviously, it is hard to compute S from D without system key d under the RSA assumption.

The Signer with the public key I sends (I, r, D) to the Credential_role, and the latter verifies the following equation:

$$D = r * I^e \ (mod \ \ n).$$

The data (I, r, D) are valid when the equation is successful, in which r and D are computed by the Trusted_role; otherwise the (I, r, D) is invalid. The Credential_role publishes in a public directory the pair (I, D) for the Signer with the public key I. The initialization processes of the system are shown in Figure 3.3.

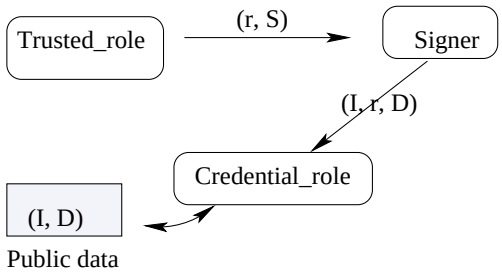

Figure 3.3: Initialization for group_1

3.3.2 The single signature scheme

The Verifier can access the public values n, e and the public pair (I, D) registered in the Credential Centre. The data D in the Credential Centre must be right, otherwise the signed message (the ticket) cannot be verified by the Verifier.

To express the general process of the single signature scheme, it is assumed that messages $m_1, m_2, ..., m_{l-1}$ $(l \geq 1)$ have already been signed by the Signer I. The messages $m_1, m_2, ..., m_{l-1}$ $(l \geq 1)$ can indicate different service requirements that are included in tickets. A user can get a valid ticket if the signature is right. The corresponding public key (I, D_{l-1}) $(D_0 = D)$ of the Signer is now registered in the

public directory of the Credential Centre. The message m_l for the next service is signed by the Signer using the secret key $S_{l-1}(S_0 = S)$. The Signer and the Verifier perform the following steps.

Input: (I, D_{l-1}, e, n),

Signer:

1. Picks $r_{l-1} \in_R Z_n$ and computes: $T_{l-1} = r_{l-1}^e (mod\, n)$.

2. Computes: $S_l = S_{l-1} * m_l \ (mod \ n)$, S_l is used as the secret key by the Signer I in the next signing operation.

3. Computes the Hashing value $d_{l-1} = h(T_{l-1}, m_l) \ (mod \ n)$.

4. Computes the final witness $t_{l-1} = r_{l-1} * (S_{l-1} * m_l)^{-d_{l-1}} \ (mod \ n)$.

Note: A ticket is the signature (t_{l-1}, T_{l-1}, m_l). The ticket is sent to the Credential Centre to make a record, it also needs to be sent to a service provider when the user wants to go shopping.

Credential_role:

The Credential_role computes D_l for the ticket, where

$$D_l = D_{l-1} * m_l^e \ (mod \ n) = S_l^e \ (mod \ n).$$

D_l is published in the Credential Centre. It is used to verify the ticket by the Verifier and used to issue another ticket.

Verifier:

5. The Verifier gets (t_{l-1}, T_{l-1}, m_l) and knows (I, D_{l-1}), then checks that:

$$d_{l-1} = h(t_{l-1}^e * D_{l-1}^{d_{l-1}} * m_l^{e d_{l-1}} \ (mod \ n), m_l) \ (mod \ n).$$

It is easy to see that if the Signer follows the protocol, the equation is valid. Indeed:

$$\begin{aligned} d_{l-1} &= h(T_{l-1}, m_l)\ (mod\ \ n). \\ T_{l-1} &= r_{l-1}^{e}\ (mod\ \ n) \\ &= (t_{l-1} * (S_{l-1} * m_l)^{d_{l-1}})^{e}\ (mod\ \ n) \\ &= (t_{l-1}^{e} * D_{l-1}^{d_{l-1}} * m_l^{e d_{l-1}})\ (mod\ \ n). \end{aligned}$$

Using this protocol the Verifier is convinced with overwhelming probability that the Signer knows the secret key S_{l-1}. This S_{l-1} is used but not revealed at the end of the protocol.

6. The Verifier sends the ticket to the Credential_role. The latter updates (I, D_{l-1}) in the public director and takes a record. The ticket (t_{l-1}, T_{l-1}, m_l) cannot be used twice since it has been marked by the Credential_role. ◇

These steps are shown in Figure 3.4.

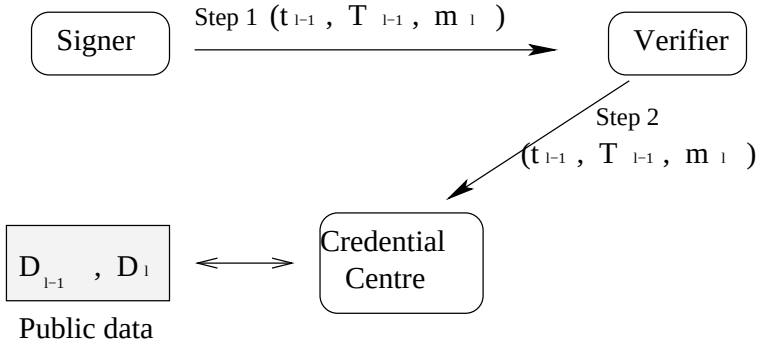

Figure 3.4: Single signature scheme

Remark: The Verifier must use the public data D_{l-1} in the Credential Centre when it checks whether the signed message is true or not. The signed message is unavailable if the data D_{l-1} are changed, then the Credential_role can revoke the anonymity of the Signer.

3.3.3 The usage of tickets in ticket group_1

Tickets are pieces of messages, which can be signatures, and the Credential_role can remember them. Ticket t_4, for instance, is the signature of a user and can be bought by the user. The following analysis is only of ticket t_4 since the signature for tickets t_1, t_2 are similar to that of t_4.

We suppose that users, service providers and services are registered in the Trusted Centre. A ticket is obtained by a user who requests the service in the ticket. When requiring a service, the user goes to the Credential Centre for a ticket. The Credential_role sends a message m_l including the service information, current time, user's requirement etc to the user. For instance, $m_l = \{$Get a bread, 10/10/2004, deliver$\}$. As a Signer, the user signs the message and makes a ticket (t_{l-1}, T_{l-1}, m_l). The ticket (t_{l-1}, T_{l-1}, m_l) can be used to obtain a service from a service provider. As a Verifier, the service provider verifies if the ticket is valid or not, using the data (I, D_{l-1}) in the Credential Centre. Neither the service provider nor the Credential_role knows who the user is. Only the Trusted_role can trace the user from the public key I. When the ticket (t_{l-1}, T_{l-1}, m_l) is used the Credential_role makes a record for the data D_{l-1}. The record is used to prevent the ticket from being duplicated and to issue a charging bill. Then users can see the charging bill at any time. This is what consumers expect when they do business on the Internet. Finally, the Credential_role can send a bill to the user.

In this mechanism presented here, a user can issue many tickets which can be used whenever, this is because whether a ticket is valid or not depends on the data in the Credential Centre only. The data $D_0, D_1, ..., D_{l-1}, D_l, ...$ are published in the public directory. Thus there is no order of tickets. The user can also lend the

ticket to others. He/She gives only the ticket (t_{l-1}, T_{l-1}, m_l) to others. This is very convenient for users. Furthermore, most computing in this scheme is done by the terminal side (the user or the service provider); this can reduce the resource of the mobile service system. The following two approaches can use to avoid the public directory from becoming too big to manage.

1) Delete the verified data in time, this can make the directory to be small,

2) A distributed solution of directories can be setup in the implementation of the scheme.

However, this scheme only suits the ticket in ticket group_1. The problems of tickets t_3, t_5, t_6, t_7 cannot be solved in the scheme of this section. A multi-signature scheme to solve these problems is explained in the next section.

3.4 Multi-signature scheme for ticket group_2

We extend the single signature scheme to a multi-signature scheme for tickets t_3, t_5, t_6, t_7. The number of signers is not limited to two or three, but v signers. This means that the scheme can also be used when services are provided by many cooperative providers.

This is, in brief, the process of the multi-signature scheme. Instead of the public key I of a signer in the last section, we use ID_i $(i = 1, 2, ...v)$ as a public keys for signers U_i since there are more than one signers in a multi-signature. In the system initialization, the Trusted_role computes and secretly sends the messages (r_i, S_i) to signers U_i in the group when the Signers are set up. This step is same as the first step in the last section. In the second step, the Credential_role verifies if the data (ID_i, r_i, D_i) sent by the Signers are valid or not. A vector $(ID_1, ID_2, ..., ID_v, g_1)$, as

the group public key, is put in the Credential Centre, where g_1 is computed by the Credential_role and is used in the first ticket verification, then the group can sign.

In the signature process, the Credential_role gets v pairs of data (t_{il}, T_{il}) from the Signers with identity $ID_i (1 <= i <= v)$ when a message m is signed, where (t_{il}, T_{il}) are computed by the Signer ID_i. In the next step, the Credential_role sends the signed message $(t_l = \prod_{il=1}^{v} t_{il} \ (mod \ n), T_l = \prod_{il=1}^{v} T_{il} \ (mod \ n), m)$ to the Signer as a ticket, where n is a public integer defined in the system initialization. The ticket is sent to the Verifier and the Verifier checks if it is true or not. The Verifier may not verify if the data g_1 in the Credential Centre is not correct, and the signed message is invalid. Therefore the Credential Centre can revoke the anonymity of the Signers. In the final step, the Verifier sends the ticket to the Credential Centre and then the Credential_role can make a record for the ticket. This process is shown in Figure 3.5.

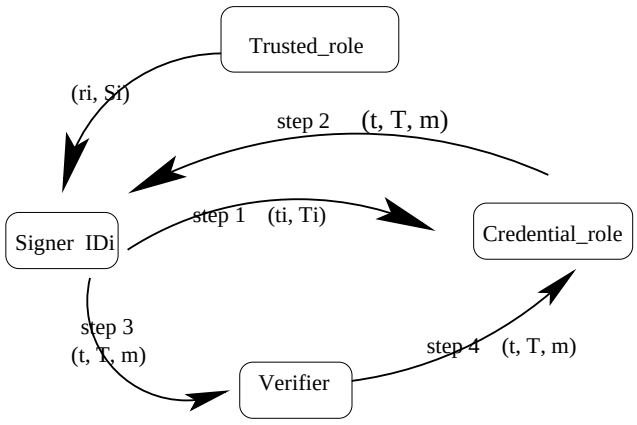

Figure 3.5: Multi-signature scheme for ticket group_2

Suppose there are v Signers $U_1, U_2, ..., U_v$ in the signature system to sign a message simultaneously, for tickets t_3, t_5, t_6, t_7, two or three signers are enough. The scheme can also cope with some other cases for example some services provided by many providers. Ticket t_6, for instance, is bound to the user and the service

provider. Then the ticket includes the agreement between these two components. Signers are needed to change in order to suit different kinds of tickets.

3.4.1 Initialization of the system

Similar to the previous section, the pair (n, e) are made public, and d is kept secret by the Trusted Centre as the system key. The Signer U_i of the system has a public key ID_i which is produced by the Trusted Centre when the signer joins the system. The Trusted_role computes:

$$r_i = k_i^e \ (mod\, n), \quad S_i = k_i * ID_i \ (mod\, n)$$

$k_i \in_R Z_n$, then $S_i^e = r_i * ID_i^e \ (mod\, n)$. Let $D_i = S_i^e \ (mod\, n)$, the Trusted_role secretly sends (r_i, S_i) to the Signer with the public key ID_i. S_i is used by U_i as the first signature key. It is hard to compute S_i from ID_i without the system key d under the RSA assumption.

The Signer U_i sends (ID_i, r_i, D_i) to the Credential_role, and the latter verifies the following equation:

$$D_i = r_i * ID_i^e \ (mod \ \ n) \tag{3.1}$$

The data (ID_i, r_i, D_i) are valid if the equation (1) is successful, which means all v Signers agree to issue a ticket. Otherwise the data (ID_i, r_i, D_i) are invalid. While the equation is successful for $i = 1, 2, ..., v$, the Credential_role computes a system public key:

$$g_1 = \prod_{i=1}^{v} D_i \ (mod \ \ n) = \prod_{i=1}^{v} S_i^e \ (mod \ \ n).$$

The Credential_role registers in a public directory a vector $(ID_1, ID_2, ..., ID_v, g_1)$ for Signers $U_1, U_2, ..., U_v$. The data g_1 is used and changed when a valid signature is done. The processes are shown in Figure 3.6.

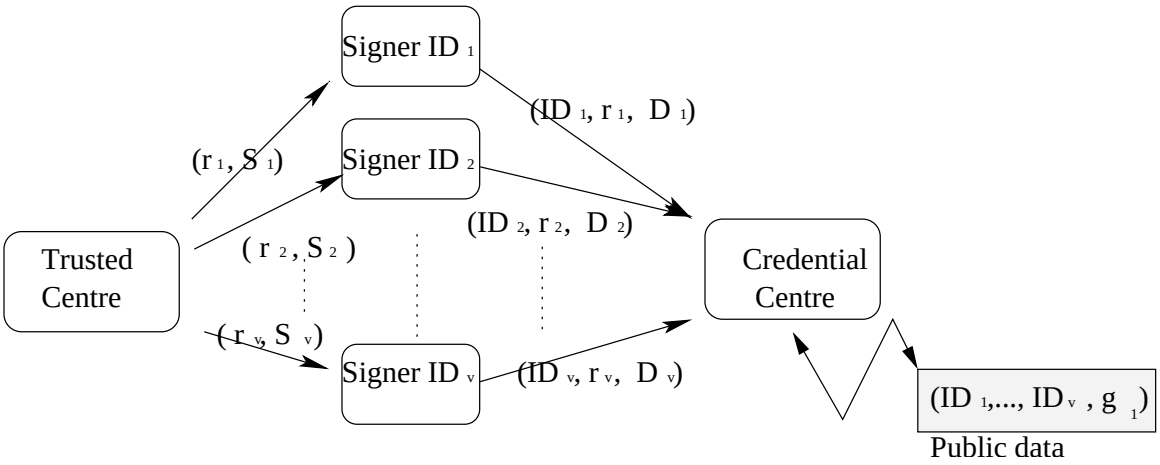

Figure 3.6: Initialization of Multi-signature scheme

3.4.2 The Multi-signature scheme

When the Verifier accesses the system public key n, e and the public vector $(ID_1, ID_2, ..., ID_v, g_1)$ in the Credential Centre, the data g_1 must be correct, otherwise the signature is unavailable since the Verifier cannot verify the signed message.

Assuming that a message $m_l (l = 1, 2, 3, ...)$ including service information, users requirements etc is signed by the Signers $U_1, U_2, ..., U_v$. $S_{i\,l-1}$, the secret key of Signer U_i is changed when the message m_l has been signed $((i = 1, 2, ..., v)$ and $S_{i0} = S_i)$. This means $S_{i\,l-1}$ is a once-a-time secret key and it improves the security of the system. z is a public prime number which is known to v Signers and it is used in the new multi-signature scheme. The processes of the multi-signature scheme are below.

Input:(ID_i, D_i, e, n),

Signer U_i:

Step 1.

1.1 Picks $r_{il} \in_R Z_n$ and computes: $T_{il} = r_{il}^e (mod\, n)$.

1.2 Computes: $S_{il} = S_{i\,l-1} * m_l \ (mod\, n)$.

S_{il} is used as the secret key by U_i in the next signing operation.

1.3 Computes: $t_{il} = r_{il} * (S_{i\,l-1} * m_l)^z \ (mod \ n)$.

1.4 Sends the pair (t_{il}, T_{il}) to the Credential_role.

The Credential_role is not able to get the secret key $S_{i\,l-1}$ from the data (t_{il}, T_{il}).

The Credential_role can now produce a ticket.

Credential_role:

Step 2. The Credential_role computes:

$$g_{l+1} = g_l * m_l^{ve} \ (mod \ n).$$

and

$$t_l = \prod_{il=1}^{v} t_{il} \ (mod \ n), \quad T_l = \prod_{il=1}^{v} T_{il} \ (mod \ n)$$

g_{l+1} is published in the public directory, it is required to issue another ticket. (t_l, T_l, m_l) is a ticket which is used for asking services.

It should be noted, for instance a ticket t_6, both the user and the service provider are Signers, however, the ticket (t_l, T_l, m_l) is only sent by the Credential_role to the user. The user sends the ticket to a service provider to ask for a purchase. The service provider, as a verifier, verifies the ticket. The verifier follows the next steps when the ticket is received.

Verifier:

Step 3. The Verifier knows the public data $(ID_1, ID_2, ..., ID_v, g_l)$ in the Credential

Centre and data (t_l, T_l, m_l), checks that:

$$T_l = t_l^e * g_l^{-z} * m_l^{-zve} \ (mod \ n) \tag{3.2}$$

It is easy to see that if the Signer and the Credential_role follow the steps, the equation (2) is valid. Indeed,

$$\begin{aligned} T_l &= \prod_{il=1}^{v} T_{il} \ (mod \ n) \\ &= \prod_{il=1}^{v} t_{il}^e * (S_{i\ l-1} * m_l)^{-ze} \ (mod \ n) \\ &= t_1^e * g_1^{-z} * m_l^{-zve} \ (mod \ n). \end{aligned}$$

Step 4. The Verifier sends the ticket to the Credential Centre. The latter updates the data g_l and prepare a charging bill for the user.

Remark: The signed message in the multi-signature scheme is invalid if the data g_l is changed. Then the Credential_role can revoke the ability to sign messages of the Signers.

3.4.3 Usage of tickets in ticket group_2

The usage of tickets in ticket group_2, ticket t_6, for instance, binds a user and service providers and it should be an agreement between the user and the service providers. The usages of other tickets are similar to that of the ticket t_6. So only the ticket t_6 is analyzed and the other tickets are omitted.

When a user requires a ticket t_6 from the Credential Centre, the Credential_role sends the user's requirement to the service providers. The Credential_role issues a public key for the user and the service providers if the service providers agree to provide the service. The Credential_role sends a message including the service information, current time, requirement and agreements of the service providers and so on to the user and the service providers. As Signers, the user and the service

providers use their secret key to sign this message, and then return the data (t_{il}, T_{il}) to the Credential Centre. The Credential_role makes a ticket (t_l, T_l, m_l) and sends it to the user. The ticket (t_l, T_l, m_l) is acceptable to the service provider. As a Verifier, the service provider uses the public data $(ID_1, ..., ID_v, g_l)$ in the Credential Centre to verify if the ticket is valid or not. Neither the service provider nor the Credential_role knows who the user is. Only the Trusted Centre can trace the user's identity from the public key ID_i. After the data g_l is updated, the user can see a clear charging bill in the Credential Centre. Finally, the Credential_role can send a bill to the user. This can be shown in Figure 3.7.

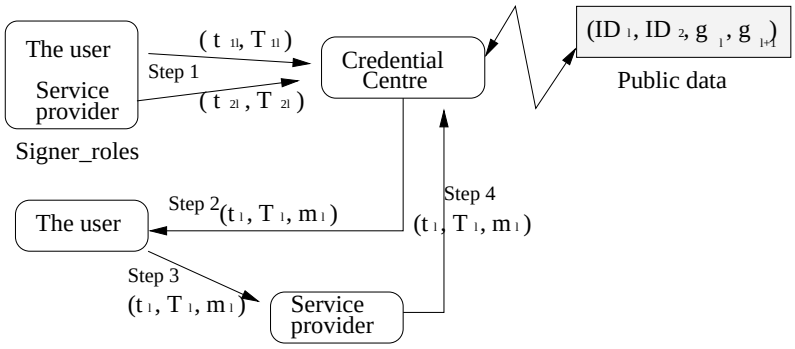

Figure 3.7: Usage of ticket t_6

As the tickets in the group_1, tickets in group_2 have no fixed order, this means no ticket should be used early or late. This is because the data for a ticket verification are $g_1, ..., g_l, g_{l+1}$ in the public directory. In addition, the data g_l is changed and marked while the ticket (t_l, T_l, m_l) is used. Therefore, a ticket cannot be used twice. The following two approaches can use to avoid the public directory from becoming too big to manage.

1) Delete the verified data in time, this can make the directory to be small,

2) A distributed solution of directories can be setup in the implementation of

the scheme.

3.5 Security analysis and the related work

The multi-signature scheme is an extension of the single signature since they use the same system key d. Therefore, the global solution has two sub-schemes. Based on the two sub-schemes, the global solution has the following features:

1. It is anonymous for users.
2. The ticket can be lent to others.
3. The security of the system is greatly improved since the secret keys S_l and S_{il} are used only once.

In this section, we analyze its security and usage for various tickets.

3.5.1 Security threats

This subsection first analyzes threats to the system, including threats from the people who do not join the system, then shows how to solve the security problems of duplication, forgery and modification. There are four roles in the scheme. They are the Signer, the Verifier, the Credential_role and the Trusted_role.

Outside: knows the public data (I, D_l) and $(ID_1, ..., ID_v, g_l)$. It is hard to compute the secret key S_l from D and S_{il} from g_l without system key d under the RSA assumption.

Verifier: knows (I, D_l) and ticket (t_{l-1}, T_{l-1}, m_l) in the first sub-scheme and $(ID_1, ..., ID_v, g_l)$ and ticket (t_l, T_l, m_l) in the second sub-scheme. But no useful mes-

sage can be obtained from these public data. The Verifier knows no more information about the key than the outside.

Credential_role: can revoke the anonymity of the users since it can control the ability to sign messages by the Signers. It knows only as much as the Outside does, it cannot get the secret key either.

Signer: knows the secret key S_l of the ticket in the sub-scheme for group_1, but cannot use the secret key S_l and the ticket twice. Use, for a second time, of the same secret key S_l to produce another ticket implies a second verification. If the previous verifier was honest, the public data in the Credential Centre would be updated and the second ticket would be rejected. There is a similar cases for the Signers in the second sub-scheme.

Trusted_role: knows the system key d, and can get the signer's key S_l. So the Trusted Centre must be trusted. Here the Trusted_role can be a judge.

The secret keys S_l and S_{il} are not revealed at the end of the process and no secret information is revealed during the running of the system. They are only dependent on the Trusted_role, and does not depend on the Credential_role. The security is also improved since the secret keys are changed once a message is signed.

Duplication is prevented since using a ticket twice requires that the ticket be verified twice, the second verification cannot succeed as the data in the Credential Centre are changed after the first verification. In the multi-signature scheme, for instance, the Credential Centre issues tickets and sends them to users. The other four, even the Trusted_role, cannot forge a ticket because the messages of (t_{il}, T_{il}) are only sent to the Credential Centre that is not able to get the secret key S_{il-1} from the data. To protect from eavesdropping or sending the ticket to other users,

the cryptographic technology like PGP (http://www.pgp.com) can be used between users and the Credential Centre. The user cannot modify the service information since it is needed in the ticket verification.

There is no limitation with service providers in the scheme. Hence this scheme can be used by wireless service providers. The PKI technologies [53] could be used in the processing of the scheme. For example, in the initialization of the system for the ticket group_1, the Trusted_role may use PKI approach to secretly send (r, S) to a Signer.

The transferred data in current wireless environments is easily lost. The ticket scheme can preserve the integrity of exchanged data in the lossy wireless environments. It means either users cannot obtain services or the system can find the lost data. For instance, in ticket group_1, tickets need to be sent to the Credential Centre and the Verifier, tickets are invalid if data is lost in these two processes. When this occurs, users have to send tickets again until they are received. The Verifier sends tickets to the Credential_role. The system can find the lost data when they are missing, and users can still get services since tickets are valid through verifications. Users may use tickets twice since the data (I, D_{l-1}) in the Credential Centre are not updated in time. However, the system will double charge the users because it receives the same ticket twice.

3.5.2 Related work

Related work has been done on this topic of mobile communication security such as [52, 66, 61, 121]. Two similar approaches, using ticket access for the third generation mobile system (UMTS) were presented by Horn and Preneel in 1998, and Martin etc in 1998 [52, 66]. In these solutions, the users obtain tokens from the UMTS service

providers, who act as brokers. The tokens are then handed by the users to the value-added service providers as a proof of their credit worthiness. The settlements between the value-added service providers and the brokers are then accomplished off-line. The UMTS service providers collect the billing information from all the value-added service providers accessed by given users and integrate them in a single bill addressed to the users. These mechanisms are a significant improvement over the prevailing mechanisms of the second generation mobile systems. However, they have the weakness of not providing anonymity to the users.

Other similar approaches for ticket-based service access are described by Patel and Crowcroft in 1997 [82], and Buttyan and Hubaux in 1999 [17]. In [82], tickets are prepaid and can only be used with the service provider that issued them (according to the categorization described here, tickets are type t_7 and require a special Outlet model). Anonymity can be provided for all services for which it is deemed appropriate. Although [17] solves several problems, tickets are issued by customer care agents and cannot be transferred to others. These two methods only solve particular mobile access problems.

In the proposed ticket-based service access scheme, the users are anonymous since their private information is not revealed to service providers and the Credential Centre. It is a global solution for all kinds of mobile services and the tickets can be lent to others, which is very convenient and useful for mobile environment users. The users can see a clear record of charges in the Credential Centre and identify any problems in the bill. Furthermore, the scheme can save mobile system resources, since most computing is done by users or service providers.

3.6 Conclusion

Mobile communication systems are becoming extremely popular, making the provision of services to mobile users an attractive business area. This can be regarded as a special form of e-commerce, where users buy services instead of products from service providers via the network. Users prefer high security and clear bill charging.

In this chapter, a ticket-based service access scheme for mobile users is proposed. The scheme can also be used by non-mobile commerce. First, the Credential Centre issues tickets for the users. Second, a ticket-based mechanism is implemented allowing the user to remunerate the service providers. Tickets provide a flexible and scalable mechanism for mobile access and users can check charges at anytime. It is an anonymous and dynamic system, and new users and new service providers can join at anytime.

Chapter 4

Untraceable Off-line Electronic Cash Flow in E-Commerce

Electronic cash payment has been playing an important role in electronic-commerce. One of the desirable characteristics is its traceability, which can prevent money laundering and can find the destination of suspicious withdrawals.

In this chapter, a new scheme for untraceable electronic cash transaction processing is developed, in which the bank involvement in the payment transaction between a user and a receiver is eliminated. The user withdraws electronic "coins" from the bank and uses them to pay to a receiver. The receiver subsequently deposits the coins back to the bank. The user remains anonymous, unless she/he spends a single coin more than once (double spend). Comparisons with other's work are discussed.

The ideas in this chapter is based on a published paper [109].

4.1 Introduction

4.1.1 Electronic cash and its properties

Traditional cash is a bearer instrument that can be used spontaneously and instantaneously, to make payments from one user to another user without the involvement

of a bank. It is the preferred method for low and medium value purchases, and transactions. Cash payments also offer privacy for they are not normally traceable by a third party. Together these factors account for the wide acceptability of traditional cash.

But traditional cash has some shortcomings. First, cash must be created such that is hard to forge and cash must be transported from one place to another place. Cash must be stored safely. Bank notes can be easily destroyed or forged using sophisticated color copier machines. Cash is annoying to carry. It spreads germs, and it can be stolen. Another shortcoming of traditional cash is that it cannot be used for payments over the phone or the Internet.

Cheques and credit cards have reduced cash circulation through-out our society, but cheques and credit cards allow people to trace the user's privacy to a degree never imagined before.

Hence, a new "cash" is needed which can allow for authenticated but untraceable messages. For example, Alice can transfer "cash" to Bob. But newspaper reporter Eve does not know Alice's identity. Bob can then deposit that money into his account, and the bank has no idea who Alice is. But if Alice tries to buy cocaine with the same "cash" she sent to Bob, she will be detected by the bank. And if Bob tries to deposit the same "cash" into two different accounts, he will be detected, but Alice remains anonymous. It is called Electronic-cash (or E-cash) to differentiate it from digital money with an audit trail, such as cards. Electronic cash can make money laundering more difficult for a coin must run a full cycle from the bank during withdrawal to the same bank for deposit (on Internet). An interesting overview of these issues is available in [43].

The ideal electronic cash system should have the following properties [109]:

1. Anonymous;
2. Revocation;
3. Efficiency;
4. Crime prevention.

Firstly, electronic-cash should be anonymous for legitimate users. The bank cannot link to the legitimate users, but it can identify the double-spenders. Only legal users are anonymous, the anonymity of illegal users should be revoked.

E-cash system must be efficient. It means not only should tracing (anonymity revocation) be performed efficiently, but the added burden to the basic system should be minimal for all involved parties—trustees, banks, users and shops. In particular, trustees must be involved only when revocation is required and remain off-line otherwise. At last, a cash system must protect all users for their electronic money, sometimes motivating crimes are more serious than other mistakes.

In an on-line electronic-cash system, the banks have to be on-line during the payment to guarantee that the coins received by shops are valid. However, it is strictly required that banks to be on-line during payment. In an off-line electronic-cash system, the bank does not need to be involved during the payment processing. Although double-spent coins cannot be prevented from being used as payment, the identity of the double spender can be identified.

4.1.2 Off-line Electronic Cash Overview

Off-line anonymous electronic cash was first introduced by Chaum, Fiat and Naor [26]. Franklin and Yung [44] presented a provably secure scheme that was not based on general computation protocols. The security relied on the DLA and on the existence of a mutually trusted party. Although Cut-and-Choose techniques were used and efficiency was not a prime consideration, Franklin and Yung were the first to illustrate how off-line e-cash could be based on the DLA, as well as the first to construct a formal security model; variations of this security model have appeared in subsequent e-cash systems [20, 74].

In 1995 Chan, Frankel, and Tsiounis [20] presented a provably secure off-line e-cash scheme that relied only on the security of RSA. This Cut-and-Choose based scheme extended the work of Franklin and Yung [44] who aimed to achieve provable security without the use of general computation protocols. In 1998, T. Yiannis and M. Yung [126] showed that the decision Diffie-Hellman assumption implies the security of the original ElGamal encryption scheme (with messages from a subgroup) without modification and they also showed that the opposite direction holds, that is, the semantic security of the ElGamal encryption was actually equivalent to the decision Diffie-Hellman problem.

4.1.3 Outline of the chapter

In this chapter, we propose an untraceable, off-line electronic cash scheme which achieves provable security without the use of general computation protocols and without requiring a trusted third party. To illustrate the practicality of schemes that are not based on general computation protocols, and show how to derive an efficient variant based on the random-oracle model. This variant thus achieves provable

security based on DLA, Cut-and-Choose technique and the existence of random oracle like hash functions. Furthermore our untraceable electronic cash scheme is much more simple than [44]. One implication is that truly anonymous e-cash can be implemented very efficiently without sacrificing security in comparison to existing account-based or anonymous-like systems.

This chapter is organized as follows: in section 2, some basic definitions and the simple examples are reviewed. The basic model of electronic cash is presented in section 3. In section 4, a new off-line electronic cash scheme is designed and the security analysis of our scheme is given in section 5. A simple example is given in section 6 and the comparison with other scheme is present in section 7. Section 8 is the conclusion.

4.2 Some Basic Definitions

4.2.1 Random oracle model

A random oracle R is a mapping (function) from $\{0,1\}^* \rightarrow \{0,1\}^\infty$ chosen by selecting each bit of $R(x)$ uniformly and independently (random and unpredictable), for every x.

Random oracles are very powerful tools, allowing the construction of very efficient signatures.

A system model is called a random oracle model if its operations are under random oracles. In this model, the functions (random oracles) produce a random answer for each new query. Of course, if the same query is asked twice, identical answers are obtained. Random oracle models are commonly used in practice and in electronic cash in particular [10, 19, 47], especially in light of a construction by

Bellare and Rogaway [20] showing instantiations of random oracles based on efficient hash function, such as MD5 [87].

For example, suppose $h' : \{0,1\}^{256} \rightarrow \{0,1\}^{64}$ is a hash function, $h''(x) = h'(x) \oplus C$, where C is a random chosen 64 bit constant and $\oplus$ denotes bitwise exclusive or. Defining $h_1(x) = h''(x[0])||h''(x[1])||h''(x[2])||...$, where $|x| = 224$ and $[i]$ is the encoding of i such that $x[i]$ has 256 bits, where $||$ denote concatenation. We define $h : \{0,1\}^* \rightarrow \{0,1\}^\infty$ as follows: for any input x, encoding x by x' consisting of x, the bit "1" and "0" to make $|x'|$ a multiple of 224 bits (the "1" and "0" are depended on the encoding). Now let $x' = x'_1||...||x'_n$, where $|x'_i| = 224$ and define $h(x) = h_1(x'_1) \oplus ... \oplus h_1(x'_n)$. Then $h(x)$ is a random oracle for its output is random and unpredictable.

4.2.2 Cut-and-Choose technique

Cut-and-Choose technique is a basic method in integer theory. We can use mathematic method to express Cut-and-Choose technique. Suppose a set $A = \{1, 2, ..., 2k\}$.

1. Alice cuts the set A into two parts

$$A_1 = \{j_1, ...j_k\},\ A_2 = A - A_1$$

   the size of A_1 is same as that of A_2.

2. Bob randomly chooses A_1, or A_2.

3. Alice gets the remain part .

The Cut-and-Choose technique works for no way but Alice can guess which part Bob chooses. Alice has a 50 percent chance of guessing which part Bob chooses

in each round of the protocol, so she has a 50 percent chance of right guess. Her chance to be right in two rounds is 25 percent, and the chance of her to be right all n times is 2^{-n}. After 16 rounds, the right rate of Alice' guessing is 1 in 65536. So Alice cannot get anything but guessing. It means Alice gets nothing and it is called zero-knowledge.

Michael Rabin was the first person to use the Cut-and-Choose technique in cryptography [84].

The first e-cash systems employed a Cut-and-Choose technique: at withdrawal the user presents $2n$ (where n is the security parameter) "terms"; the bank "cuts-and-chooses" n, for which the user reveals the inner structure. The bank verifies their correctness and blindly signs the remaining n. At payment a similar Cut-and-Choose technique is employed for the shop to verify a "hint" on the user's identity, such that upon double-spending two hints identify the user. The Cut-and-Choose technique is a tool for a zero-knowledge proof of correctness of the coin, thus preserving user anonymity. Security is guaranteed with probability overwhelming in n, but a scheme's communication, computation and storage requirements are multiplied by a factor of n.

4.2.3 DLA

The source of DLA is the discrete logarithm problem.

The discrete logarithm problem is as follows: given an element g in a group G of order t, and another element y of G , the problem is to find x, where $0 < x < t - 1$, such that y is the result of composing g with itself x times. In some groups there exist elements that can generate all the elements of G by exponentiation (that is, applying the group operation repeatedly) with all the integers from 0 to $t-1$. When

this occurs, the element is called a generator and the group is called cyclic. Rivest [87] has analyzed the expected time to solve the discrete logarithm problem both in terms of computing power and cost.

Discrete Logarithm Assumption (DLA) is an assumption that the discrete logarithm problem is believed to be difficult and also to be the hard direction of a one-way function. For this reason, it has been used for the basis of several public-key cryptosystems, including the famous ElGamal system.

4.2.4 Blind signature

Blind signature schemes, first introduced by Chaum [26] allow a person to get a message signed by another party without revealing any information about the message to the other party.

Suppose Alice has a message m that she wishes to have it signed by Bob, and she does not want Bob to learn anything about m. Let (n, e) be Bob's public key and d be his private key. Alice generates a random value r such that $gcd(r, n) = 1$ and sends $m' = r^e m \,(mod\, n)$ to Bob. The value m is "blinded" by the random value r, and hence Bob can derive no useful information from m. Bob returns the signed value, $s' = (m')^d = (r^e m)^d \,(mod\, n)$ to Alice. Since $s' = r m^d \,(mod\, n)$, Alice can obtain the true signature s of m by computing $s = s' r^{-1} \,(mod\, n)$.

A probabilistic polynomial time (p.p.t) Turing machine M is a Turing machine which can flip coins as an additional primitive step, and on input string x runs for at most a polynomial in $|x|$ steps. $M(x, y)$ denotes the outcome of M on input x when internal coin tosses are y.

4.3 Basic model

Electronic cash (in particular off-line untraceable electronic cash) has sparked wide interest among cryptographers ([43, 124, 87, 125, 74], etc.). In its simplest form, an e-cash system consists of three parts (a bank B, a user U and a shop S) and three main procedures as shown in Figure 4.1 (withdrawal, payment and deposit). In a coin's life-cycle, the user U first performs an account establishment protocol to open an account with the bank B. To obtain a coin U performs a withdrawal protocol with B and during a purchase U spends a coin by participating in a payment protocol with the shop S. To deposit a coin, S performs a deposit protocol with the bank B.

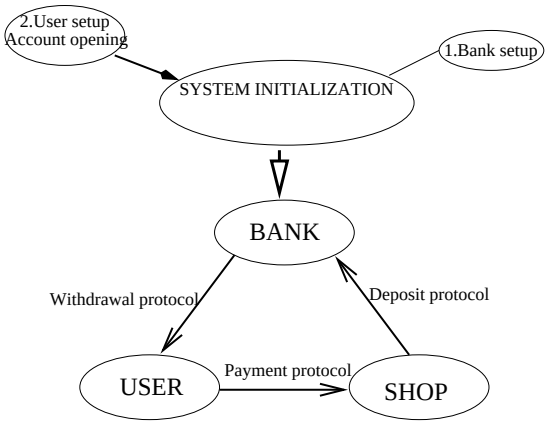

Figure 4.1: Basic off-line electronic cash system

Users and shops maintain an account with the bank, while

1. U withdraws electronic coins from his account, by performing a withdrawal protocol with the bank B over an authenticated channel.

2. U spends a coin by participating in a payment protocol with a shop S over an anonymous channel, and

3. S performs a deposit protocol with the bank B, to deposit the user's coin into

his account.

The system is *off-line* if during payment the shop S does not communicate with the bank B. It is *untraceable* if there is no p.p.t. TM (probabilistic polynomial-time Turing Machine) M access to all bank's views of withdrawal, payment and deposit protocols, can decide a coin's origin. It is *anonymous* if the bank B, in collaboration with the shop S, cannot trace the coin to the user. However, in the absence of tamper-proof hardware, electronic coins can be copied and spent multiple times by the user U. This has been traditionally referred to as double-spending. In anonymous on-line e-cash, double-spending is prevented by having the bank check if the coin has been deposited before. In off-line anonymous e-cash, however, this solution is not possible; instead, as proposed by Chaum, Fiat and Naor [26], the system guarantees that if a coin is double-spent the user's identity is revealed with overwhelming probability.

There are also three additional proceedings such as the bank setup, the shop setup, and the user setup (account opening). They describe the system initialization, namely creation and posting of public keys and opening of bank accounts. Although they are certainly parts of a complete system, these are often omitted as their functionalities can be easily inferred from the description of the three main procedures. For clarity we only describe the bank setup and the user setup (because the shop setup is as similar as user setup) for our new scheme in the next section.

4.4 New off-Line Untraceable Electronic Cash Scheme

In this section, we propose a new off-line untraceable electronic cash scheme.

Our scheme includes two basic processes in system initialization (bank setup and user setup) and three main protocols: a new withdrawal protocol with which U withdraws electronic coins from B while his account is debited, a new payment protocol with which U pays the coin to S, and a new deposit protocol with which S deposits the coin to B and has his account credited.

4.4.1 System Initialization

We only describe the bank setup and the user setup based on Discrete Logarithm Assumption and random-oracle model here and omit the detail of the shop setup (because the shop setup is similar to the user setup).

Bank's setup: (performed once by B)

Primes p and q are chosen such that $|p-1| = \delta + k$ for a specified constant δ, and $p = \gamma q+1$, for a specified small integer γ. Then a unique subgroup G_q of prime order q of the multiplicative group Z_p and generators g, g_1, g_2 of G_q are defined. Secret key $x_B \in_R Z_q$ for a denomination is created, where $a \in_R A$ means that the element a is selected randomly from the set A with uniform distribution. Hash function H from a family of collision intractable (or, ideally, according to [43], correlation-free one way) hash function is also defined. B publishes p, q, g, g_1, g_2, H and its public keys $h = g^{x_B} (\mathrm{mod}\ p)$, $h_1 = g_1^{x_B} (\mathrm{mod}\ p)$, $h_2 = g_2^{x_B} (\mathrm{mod}\ p)$.

The secret key x_B is safety under the DLA. The Hash function is used in withdrawal process.

User's setup (account opening): (performed for each user U)

The bank B associates the user U with $I = g_1^{u_1} (\mathrm{mod}\ p)$ where $u_1 \in G_q$ is generated by U and $g_1^{u_1} g_2 \neq 1 (\mathrm{mod}\ p)$. U computes $z = h_1^{u_1} h_2 = (I g_2)^{x_B} (\mathrm{mod}\ p)$.

In system initialization, the communication complexity is $O(l)$ for the user only sends its account I of length l bits to the bank, and the computation complexity is $O(1)$.

After the user's account and the shop's account opening, we can describe the new untraceable electronic cash scheme.

4.4.2 New Untraceable Electronic Cash Scheme

We now describe the new off-line untraceable electronic cash scheme which includes three protocols: withdrawal protocol, payment protocol and deposit protocol.

Withdrawal: (over an authenticated channel between B and U)
The withdrawal creates a "restrictively blind" signature $B_i\,(i = 1, ..., k)$ of I and using Cut-and-Choose technology. U puts a signature as $(Ig_2)^s$ where s is a random number (chosen by U and kept secret).

1. The user chooses $a_i, c_i, 1 \leq i \leq k$, independently and uniformly at random form the residues (mod p).

2. The user forms and sends to the bank k blinded candidates $B_i = H(x_i, y_i)\ (mod\, p)$, $1 \leq i \leq k$, where

$$x_i = g^{a_i}(mod\ p),\ \ y_i = g_1^{a_i \oplus (I||c_i)}(mod\ p).$$

3. The bank chooses a random subset of $k/2$ blinded candidate indices

$$R = \{i_j\},\ 1 \leq i_j \leq k,\ 1 \leq j \leq k/2$$

and transmits it to the user.

4. The user transmits $A = (Ig_2)^s(mod\ p)$ and $z^{'} = z^s(mod\ p)$ to bank.

5. The user displays a_i, c_i values for all i in R, and the bank checks them. To simplify notation we assume that $R = \{k/2+1, k/2+2, ..., k\}$.

6. The bank verifies: $A^{x_B} = z^{'}(mod\ p)$ and gives the user the electronic coin C,

$$C = \prod_{i \notin R} B_i = \prod_{1 \le i \le k/2} B_i (mod\ p).$$

We use the Hash function in step 2 and the Cut-and-Choose technique in step 3, step 5 and step 6. The basic safety in withdrawal is protected by Hash function, and the deep safety is kept by the Cut-and-Choose technique. Indeed, since Cut-and-Choose technique is zero-knowledge proof, then nothing can be inferred about the coin. At the final step, the output of the coin is random and unpredictable. It is a random oracle model and secure in withdrawal.

In withdrawal process, the communication complexity is $O(k)$ for the user sends $B_i, 1 \le i \le k$ to the bank and the bank sends R which length is $k/2$ to the user, the computation complexity is $O(q^{k/2})$, since x_i, y_i, B_i, A, Z', C must be computed. $C = \prod_{1 \le i \le k/2} B_i (mod\ p)$ is the main computation.

Payment: (performed between the user and the shop over an anonymous channel) At payment time the user supplies information to the receiver (which is later forwarded to the bank) so that if a coin is double-spent the user is identified. The detailed payment is as below. (the user and the shop agree on date/time):

1. The user sends C to the shop.

2. The shop chooses a random binary string $z_1, z_2, ..., z_{k/2}$, and sends to the user.

3. The user responds as follows, for all $1 \le i \le k/2$:

   a. If $z_i = 1$, then sends to the shop: a_i, y_i

   b. If $z_i = 0$, then sends the shop: $x_i, a_i \oplus (I||c_i), c_i$

4. The shop verifies that C is right since the user's responses can fit C.

The user gives some data to the shop according its random binary string $z_1, z_2, ..., z_{k/2}$. The random binary strings from different shops are different with high probability. Two different shops will send to the user complementary binary values for at least one bit z_i for which B_i was of the proper form. The user's account I can be obtained from $a_i, y_i, x_i, a_j \oplus (I||c_j), c_j$ when $i = j$. The different strings are obtained if the user uses the same coin C twice, then the user has a high probability of being traced.

In payment, the communication complexity is $O(k+l)$ for the shop sends $z_i, 1 \leq i \leq k/2$ to the user and the user sends responds $a_i, y_i, x_i, a_i \oplus (I||c_i), c_i$ to the shop. The computation complexity is $O(1)$ for only the shop verifies the form C.

Deposit:(The receiver deposits a coin to a bank)
After some delay for the system is off-line, the shop sends to B the payment transcript, consist of $C, a_i, y_i, x_i, a_j \oplus (I||c_j), c_j$ and the date/ time of the transaction. The bank verifies their correctness and credits his account.

In deposit, the communication complexity is $O(k + l)$ for the shop sends user's responds $a_i, y_i, x_i, a_i \oplus (I||c_i), c_i$ to the bank. The computation complexity is $O(1)$, since only the bank verifies whether C was used before or not.

Remark The receiver (shop) deposits the coin in its account provided by the bank with a transcript of the payment. If the user uses the same coin C twice, then the user has a high probability of being traced: with high probability, two different receivers send complementary binary values for at least one bit z_i for which B_i was of the proper form. The bank can easily search its records to ensure that C has not been used before. If the user uses C twice, then with high probability, the bank has

both $a_i, a_i \oplus (I||c_i)$ and c_i with same i. Thus, the bank can isolate the user and trace the payment to the user's account I.

In our new scheme, the communication complexity is $O(k+l)$ and the computation complexity is $O(q^{k/2})$ where k is the security parameter and l is the size of user's identity I.

The system initialization in figure 1 includes the security random oracle model and how to get the bank setup and the user setup. It is important for the withdrawal, payment and deposit in our new scheme. The security of our new scheme is also based on the system initialization.

We have shown how to derive an efficient scheme based on the random-oracle model. It achieves provable security based on DLA and the existence of random oracle like hash functions. Based on this system initialization, three new protocols with Cut-and-Choose methodology are designed. It is much more secure due to the Cut-and-Choose methodology and random-oracle model.

4.5 Security Analysis

An off-line E-cash scheme is secure [44] if the following requirements are satisfied:

1. *Unreusable*: If any user uses the same coin twice, the identity of the user's can be computed.

2. *Unexpandable*: With n withdrawal proceedings, no p.p.t. (Probabilistic polynomial time) Turing Machine can compute $(n+1)$th distinct and valid coin.

3. *Unforgeable*: With any numbers of the customer's withdrawal, payment and deposit, no p.p.t. Turing Machine can compute a single valid coin.

4. *Untraceable*: With any numbers of the customer's valid withdrawal, payment and deposit protocols, no p.p.t. Turing Machine can compute a legal user's identity.

We employ Discrete Logarithm methods in our new scheme; these methods have been suggested in many of the recent e-cash schemes to bind identities (an unavoidable issue in off-line e-cash). These methods were started in [78] and continued by others [20, 74] as well as in [125]. The security of our scheme is based on the hardness of Discrete Logarithms [126] and the Cut-and-Choose technology. The Cut-and-Choose technology is based on zero-knowledge proof, and the scheme assumes that the hash function used is perfect (that is random oracle).

We have analyzed the untraceability and unreusebility before. To prevent the cooperation of the bank and some others frame the user as a multiple spender in the scheme, we use digital signature Z^s for s is known only by the user. To prevent unexpanding, we use the Discrete Logarithm methods and Cut-and-Choose technology.

A possible problem with the scheme is a collusion between a user U and the second shopkeeper. After having user transactions with two receivers which send the same information to the bank, the bank knows that with high probability one of them is lying, and the bank can decide the first purchase is right by the date/time in the payment but cannot trace the coin to the user's account.

To prevent the bank frame the user as a multiple spender in the scheme, we use digital signature Z^s for s is known only by the user. The user is protected against frame-up only computationally, not unconditionally.

4.6 A simple example

We give a simple example to explain how our scheme works in this section.

Bank setup

Suppose $(p, q, \gamma, k) = (47, 23, 2, 4)$, then $G_q = \{0, 1, 2, ..., 22\}$ is a subgroup of order 23. $g = 2, g_1 = 3, g_2 = 5$ are the generators of G_q. Bank's secret key $x_B = 4$ and hash function $H(x, y) = 3^x * 5^y \, (mod\, 47)$. Bank publishes $H(x, y)$ and $\{p, q, g, g_1, g_2, h, h_1, h_2\} = \{47, 23, 2, 3, 5, 16, 34, 14\}$.

User setup (opening an account)

Every user has a secret key. We assume the secret of a user is $u_1 = 7$ and the user sends $I = g_1^{u_1} = 32 \, (mod\, 47)$ to the bank. The user computes

$$z = h_1^{u_1} * h_2 = 18 \, (mod\, 47).$$

The user performs the following steps when she/he does shopping.

1. **Withdrawal**

   The user chooses a one-time secret key $s = 3$ and

   (a) Chooses

   $$\{a_1, a_2, a_3, a_4, c_1, c_2, c_3, c_4\} = \{1, 2, 3, 4, 11, 12, 13, 14\}$$

   (b) The user computes (We omit module 47):

   $$\{x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4\} = \{2, 4, 8, 16, 32, 7, 7, 32\}$$

   and sends B_i to the Bank:

   $$\{B_1, B_2, B_3, B_4\} = \{16, 45, 26, 36\}.$$

(c) The Bank chooses $R = \{3, 4\}$ (suppose) and sends it to the user.

(d) The user transmits $A = (Ig_2)^s = 44\,(mod\,47)$ and $z' = z^s = 4\,(mod\,47)$ to the Bank.

(e) The user displays $(a_3, a_4, c_3, c_4) = (3, 4, 13, 14)$ to the Bank, and The Bank checks the correctness of the B_3, B_4.

(f) The Bank verifies $A^{x_B} = 44^4 = 34 = z'\,(mod\,47)$ and gives the user the coin C:

$$C = B_1 * B_2 = 16 * 45 = 15\,(mod\,47).$$

2. **Payment**

The user can use the coin in shop as follows. If the user uses the coin only once, she is legal. But when she uses the coin twice she will be identified.

(a) The user sends $c = 15$ to a shop (The user needs not to display I).

(b) The shop chooses a random binary string to the user, suppose it is $\{z_1, z_2\} = \{1, 0\}$.

(c) The user responds to the shop $(a_1, y_1) = (1, 32)$ for $z_1 = 1$ and $(x_2, c_2, a_2 \oplus (I||c_2)) = (4, 12, 526)$ for $z_2 = 0$.

(d) The shop sends the responds $(a_1, y_1, x_2, c_2, a_2 \oplus (I||c_2))$ to the bank and the bank checks if the responds are correct with $\{B_1, B_2\} = \{16, 45\}$.

3. **Deposit and owner tracing**

The bank puts the money into the shop's account when the checking of the coin C is correct. The shop can also see that the money in his account is added. If the user uses the coin twice, the bank gets $a_i, a_j \oplus (I||c_j)$ and c_j with $i = j$, then the user's identity I can be found.

4.7 Comparisons

In this section, we compare our new scheme with of the proposed approaches in [44, 83]. The communication complexity and computation complexity of our protocols are better than that in [44, 83].

We first recall the basic main processing stages of M. Franklin and M.Yung [44].

1. $R \rightarrow A : Z_1^* = \rho_1^{e_A} h(z_1) \, mod \, N_A, ..., Z_{2k}^* = \rho_{2k}^{e_A} h(z_{2k}) \, mod \, N_A$, where
   (a) $\rho_i \in_R Z_{N_A}^*$ for all $1 \leq i \leq 2k$.
   (b) $z_i = e'(s, r_i)$ for all $1 \leq i \leq 2k$, where each r_i is uniformly random over the appropriate range, and where e' is a public and easily computable function.
   (c) h is a publicly known collision-free hash function.
2. In round two, the following messages are sent:
   (a) $R \rightarrow A : [r_i, \rho_i : i \in S]$.
   (b) $A \rightarrow R : C' = \prod_{j \notin S} (z_j^*)^{d_A} \, (mod \ N_A)$, assuming that the messages received so far are consistent (otherwise A terminates the protocol); that is:

$$z_j^* \rho_j^{-e_A} = h(e'(s, r_j)) \quad j \in S.$$

3. R finds $[r, C]$ where
   (a) $r = [r_j : j \notin S]$
   (b) $C = C' \prod_{j \notin S} \rho_j^{-1} \, (mod \ N_A)$
   (c) The public and easily computable function e is defined to be

$$e(s, r) = [e'(s, r[1]), ..., e'(s, r[k])].$$

Where e_A, d_A is A's encryption and decryption key, respectively. e_A is public and d_A is secret key. N_A is public.

In [44], the communication complexity is $O(k^2 l)$ bits where k is a security parameter and l is the size of a signed bit; the computation complexity is $O(n^{k/2})$. In our case, the communication complexity is $O(k + l)$. The computation complexity is $O(q^{k/2})$. As general, $q < n$.

We now recall the main processing steps in payment of G. Maitland and C. Boyd [83].

1. A customer retrieves the previously calculated values $T_1, T_2, T_3, d_1, d_2, d_3$ and d_4.

2. The customer uses the values $T_1, T_2, T_3, d_1, d_2, d_3, d_4$ and the message msg to complete the challenge and response phases.

   (a) Challenge Phase: Calculate

   $$c = H(g \parallel h \parallel y \parallel a_0 \parallel a \parallel T_1 \parallel T_2 \parallel T_3 \parallel d_1 \parallel d_2 \parallel d_3 \parallel d_4 \parallel msg)$$

   (b) Response Phase: Compute

   $$s_1 = r_1 - c(e_i - 2^{\gamma_1}), s_2 = r_2 - c(x_i - 2^{\lambda_1}),$$

   $$s_3 = r_3 - ce_i w, s_4 = r_4 - cw$$

   The resulting group signature is $(c, s_1, s_2, s_3, s_4, T_1, T_2, T_3)$.

3. The merchant verifies the group signature of the payment transcript msg as follows:

(a) Compute:

$$d_1' = a_0^c T_1^{s_1 - c2^{\gamma_1}} / (a^{s_2 - c2^{lambda}} y^{s_3}) mod\, n$$

$$d_2' = T_2^{s_1 - c2^{\gamma_1}} / g^{s_3}\, mod\, n$$

$$d_3' = T_2^c g^{s_4}\, mod\, n$$

$$d_4' = T_3^c g^{s_1 - c2^{\gamma_1}} h^{s_4}\, mod\, n$$

$$c' = H(g \parallel h \parallel y \parallel a_0 \parallel a \parallel T_1 \parallel T_2 \parallel T_3 \parallel d_1' \parallel d_2' \parallel d_3' \parallel d_4' \parallel msg)$$

(b) Accept the group signature if and only if $c = c'$ and

$$s_1 \in \pm\{0,1\}\epsilon(\gamma_2 + k) + 1$$

$$s_2 \in \pm\{0,1\}\epsilon(\lambda_2 + k) + 1$$

$$s_3 \in \pm\{0,1\}\epsilon(\gamma_1 + 2\iota_p + k + 1) + 1$$

$$s_4 \in \pm\{0,1\}\epsilon(2\iota_p + k) + 1$$

In the payment, the communication complexity is $O(kl)$ bits and the computation complexity is $O(n^k)$.

	communication complexity	Computation complexity
Scheme in [44]	$O(k^2l)$	$O(n^{k/2})$
Scheme in [83]	$O(kl)$	$O(n^k)$
New scheme	$O(k+l)$	$O(q^{k/2})$

Table 4.1: Comparisons of complexity

4.8 Conclusion

In this chapter an untraceable electronic cash scheme is designed which is an off-line scheme and without using of general computation protocols and without the requirement of a trusted party. We have shown how to derive an efficient cash scheme based on the variants in the random-oracle model. The variants thus achieve provable security based on DLA and the existence of random oracle like hash function. The security of the system is based on DLA and the Cut-and-Choose methodology. We give a simple example to explain our new untraceable scheme and compare our scheme with other e-cash schemes.

Chapter 5

Building a consumer scalable anonymity payment protocol for Internet purchases

We have developed an efficient electronic cash scheme in the last chapter. This chapter describe a consumer scalable anonymity payment protocol. The protocol uses electronic cash for payment transactions. In this new protocol, from the viewpoint of banks, consumers can improve anonymity if they are worried about disclosure of their identities. An agent provides a higher anonymous certificate and improves the security of the consumers. The agent certifies re-encrypted data after verifying the validity of the content from consumers, but with no private information of the consumers required. With this new method, each consumer can get the required anonymity level, depending on the available time, computation and cost.

We also analyze how to prevent a consumer from spending a coin more than once and how to use the proposed protocol for Internet purchases. After comparing with another scheme and discussing the properties of the new payment protocol, the new method is proved that it is more efficient and can prevent from eavesdropping, tampering and "perfect crime" effectively. It is promising for electronic trades through the Internet.

The information in this chapter has been published in [110].

5.1 Introduction

Recent advances in the Internet and WWW have enabled rapid development in e-commerce. More and more businesses begin to develop or adopt e-commerce systems to support their selling/business activities. While this brings convenience for both consumers and vendors, many consumers have concerns about security and their private information when purchasing over the Internet, especially with electronic payment or e-cash payment. Consumers often prefer to have some degree of anonymity when shopping over the Internet.

There are a number of proposals for on-line electronic cash systems [22, 30, 23, 67, 26, 79, 81]. All of them lack flexibility in anonymity. David Chaum [22] first proposed an on-line payment system that guaranteed receiving valid coins. This system provides some levels of anonymity against a collaboration of shops and banks. However, users have no flexible anonymity and banks have to keep a very big database for users and coins. On-line payment systems force banks to be on-line at payment processes that is a very strict requirement. This increases the computation cost, proportional to the size of the database of spent coins. If a large number of people start using the system, the size of this database could become very large and unmanageable. Under the circumstances, the task of maintaining and querying a database of spent coins is probably beyond today's state-of-the-art database systems [110].

Off-line payment systems were designed to lower the cost of transactions due to the delay in verifying batch processes. Off-line payment systems, however, suffer from the potential of double spending, whereby the electronic currency might be

duplicated and spent repeatedly.

Moreover, as mentioned above, the on-line e-cash payments need more computing resources. Most of the previously designed off-line schemes are only for micropayments. They rely on the heuristic proofs of security and therefore do not formally prevent fraud and counterfeit money. Under these conditions, most on-line and off-line payment schemes do not provide efficient anonymity for consumers. Hence, a new payment scheme for the purchases over the Internet with untraceability, flexible anonymity and with low computation is very useful and very important.

In this chapter, we analyze electronic-payment models first, then propose a new off-line electronic cash scheme, in which the anonymity of consumers is scalable and can be done by consumers themselves. Consumers can get the required anonymity without showing their identities to any third party. Furthermore, the new method can prevent from eavesdropping, tampering, impersonation and "perfect crime" effectively. It is a more efficient electronic cash scheme by comparing with David Pointcheval [79]. This is truly anonymous for legal consumers and can trace consumers' identities for double spending.

The chapter is organized as follows. In the following section, some basic definitions and the simple examples are reviewed. The payment model and the anonymity provider agent are described in section 3. The design of a new off-line electronic cash scheme and its complexity are detailed in section 4 and the security analysis of the scheme is given in section 5. Comparing with David Pointcheval [79] is shown in section 6. An example and how to use the new e-cash for Internet purchases are given in section 7. Conclusions are included in section 8.

5.2 Some Basic Definitions

5.2.1 ElGamal encryption system

ElGamal encryption system [35] is a public key encryption scheme which provides semantic security. Let us briefly recall it.

1. The system needs a group G of order q, and a generator g. The secret key is an element $X \in Z_q = \{0, 1, ..., q-1\}$ and the public key is $Y = g^X$.

2. For any message $m \in G$, the ciphertext of m is $c = (g^r, Y^r m)$, for a random $r \in Z_q - \{0\}$.

3. For any ciphertext $c = (a, b)$, the message m can be retrieved by $m = b/a^X$.

5.2.2 Undeniable signature scheme and Schnorr signature scheme

The undeniable signature scheme, devised by Chaum and van Antwerpen [25], is a non-self-authenticating signature schemes, where signatures can only be verified with the signer's consent. However, if a signature is only verifiable with the aid of a signer, a dishonest signer may refuse to authenticate a genuine document. Undeniable signatures solve this problem by adding a new component called the disavowal protocol in addition to the normal components of signature and verification.

An undeniable proof scheme consists of the following algorithms:

1. The key generation algorithm K which outputs random pairs of secret and public keys (sk, pk).

2. The proof algorithm $P(sk, m)$ which inputs a message m, returns an "undeniable signature" S on m.

   However this proof "S" does not convince anybody by itself. To be convinced of the validity of the pair (m, S), relative to the public key pk, one has to interact with the owner of the secret key sk.

3. The confirmation process confirms (sk, pk, m, S), which is an interactive protocol between the signer and the verifier, where the prover (the signer) tries to convince the validity of the pair (m, S).

4. The disavowal process is an interactive protocol between the signer and the verifier, where the prover (the signer) tries to prove that the pair (m, S) is not valid (that is not produced by him).

Schnorr proposed an undeniable signature scheme in 1991 [96]. We simply recall it.

The system needs primes p and q such that q is divided by $(p-1)$, that is, $q\|(p-1)$, $g \in Z_p$ with order q, that is $g^q = 1(mod p), g \neq 1$. A consumer generates by himself a private key s which is a random number in Z_q. The corresponding public key v is the number $v = g^{-s}(mod p)$.
To sign message m with the private key s the consumer performs the following steps: 1. Computes $x = g^r(mod p)$, where $r \in Z_q$ is a random number. 2. Computes $e = H(x, m)$, where H is a hash function. 3. Computes $y = r + se(mod p)$ and output the signature (e, y).
To verify the signature (e, y) for message m with the public key v a verifier computes $\overline{x} = g^y v^e (mod p)$ and checks $e = h(\overline{x}, m)$.

Table 5.1: Schnorr signature scheme

There are three exponentiations in the Schnorr signature scheme, one is from the signer and other two from the verifier.

5.3 Basic model and new payment model

We show the basic payment model and then discuss the new payment model in this section.

5.3.1 Basic payment model

Electronic cash has sparked wide interest among cryptographers ([87, 125, 74], etc.). In its simplest form, an e-cash system consists of three parts (a bank B, a consumer U and a shop S) and three main procedures as shown in Figure 1.2 (withdrawal, payment and deposit). In a coin's life-cycle, the consumer U first performs an account establishment protocol to open an account with the bank B.

The consumers and the shops maintain an account with the bank, while

1. U withdraws electronic coins from his account, by performing a withdrawal protocol with the bank B over an authenticated channel.
2. U spends a coin by participating in a payment protocol with a shop S over an anonymous channel, and
3. S performs a deposit protocol with the bank B, to deposit the consumer's coin into his account.

The system is *off-line* if the shop S does not communicate with the bank B during payment. It is *untraceable* if there is no p.p.t. TM (probabilistic polynomial-time Turing Machine) that can identify a coin's origin even if one has all the information of withdrawal, payment and deposit transactions. It is *anonymous* if the bank B, in collaboration with the shop S, cannot trace the coin to the consumer. However, in the absence of tamper-proof hardware, electronic coins can be copied and spent

multiple times by the consumer U. This has been traditionally referred to as double-spending. In on-line e-cash, double-spending is prevented by having the bank check if the coin has been deposited before. In off-line e-cash, however, this solution is not possible; instead, as proposed by Chaum, Fiat and Naor [26], the system guarantees that if a coin is double-spent the consumer's identity is revealed with overwhelming probability.

There are also three additional processes such as the bank setup, the shop setup, and the consumer setup (account opening). They describe the system initialization, namely creation and posting of public keys and opening of bank accounts. Although they are certainly parts of a complete system, these are often omitted as their functionalities can be easily inferred from the description of the three main procedures. For clarity we only describe the bank setup and the consumer setup (because the shop setup is as similar as the consumer setup) for the new scheme in the next section.

Besides the basic participants, a third party named Anonymity Provider (AP) agent is involved in the scheme. The AP agent helps the consumer to get the required anonymity but is not involved in the purchase process. The new model can be shown in Figure 5.1. The AP agent gives a certificate to the consumer who needs a higher level of anonymity.

5.3.2 Anonymity Provider Agent

Here we explain what is an AP agent. Assuming a consumer owns a valid coin $c = \varphi(pk_B, pk_u, y)$ with its certificate $Cert_c$, which guarantees correct withdrawal

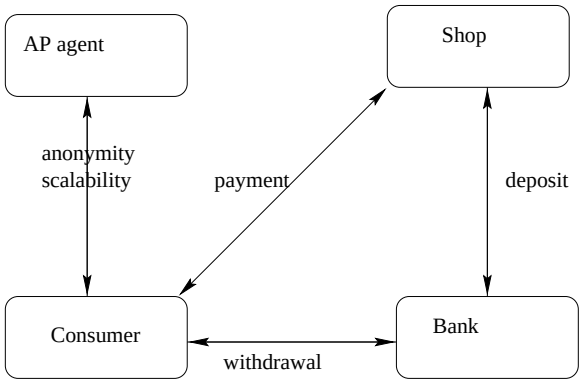

Figure 5.1: New electronic cash model

from the bank. Where $\varphi(pk_B, pk_u, y)$ is a function on the public keys of the bank, the user and a variable y, that is (pk_B, pk_u, y). A coin's validity depends on its certificate. Therefore the bank can revoke the anonymity of the consumer who spends a coin twice. After the following processes with the AP agent, the consumer owns a new valid coin, $c' = \varphi(pk_B, pk_u, y + t)$ with its certificate $Cert_{c'}$.

1. The consumer re-encrypts the coin c into $c' = \varphi(pk_B, pk_u, y + t)$.

2. The consumer provides an undeniable signature S, using c as a public key associated with the secret key sk_u of the user, of the equivalence between c and c'. This equivalence is guaranteed by the variable t.

3. The consumer confirms the validity of this signature S to the AP agent.

4. The AP agent certifies the new coin c' and sends $Cert_{c'}$ to the consumer.

Indeed, after steps 2 and 3, the AP is convinced that the conversion has been performed by the owner of the coin c; c' is equivalent to c. The owner of c is not able to deny S (the relation between c and c'). The AP agent should be an electronic notarized participant in the system. It verifies the information of consumers, but does not need to know any private information.

5.3.3 Proof of ownership of a coin

This subsection shows how users prove the ownership of a coin. Let us assume that Y is the public key of the bank, and $I = g^{x_u}$ the identity of a consumer. $H(x, y)$ is a hash function. A coin is the encryption of I: $c = (a = g^r, b = Y^r I^s)$ which is afterwards certified by the bank, where r, s are random numbers. With the certificate of the bank, one knows that the encryption is valid. Therefore, in order to prove his ownership, the consumer has just to convince of his knowledge of (x_u, r, s) such that $b = Y^r I^s$. This can be expressed as follows.

1. Consumers choose random $k \in Z_p$, then compute $t = Y^k g^s (mod\, p)$ and $e = H(m, t)$ where m is a mixed message of c, current time etc,
2. Then compute $u = k - re(mod\, p)$, $v = s - x_u e(mod\, p)$, and $t_1 = g^{(s-1)x_u e}(mod\, p)$,
3. The signature finally consists of (e, u, v, t_1),
4. In order to verify it, one has just to compute $t' = Y^u g^v b^e$ and check whether $t' = tt_1$ and $e = H(m, t'/t_1)$.

Table 5.2: Proof of validity of a coin $c = Y^r I^s$

We like to note that the message m includes the coin c, the certificate $Cert_c$, the current time and so on. The coin c can not be used again by the shop because the variable of the current time has been changed when the owner of the shop wants to use the coin.

In the proof process, there are six exponentiations, three are from the consumer and other three from the verifier.

Then, a scrambled coin is simply got by multiplying both parts of the old one by respective bases, g and Y, put at a same random exponent ρ :

$$c' = (a' = g^\rho a, b' = Y^\rho b) = (g^{r+\rho}, Y^{r+\rho} I^s).$$

Then, if the owner of the old coin has certified the message $m' = h^{\rho}$, equivalence of both coins can be proven with the proof of equivalence of three discrete logarithms:

$$log_h m' = log_g(a'/a) = log_Y(b'/b)$$

where h is a public variable.

5.4 Self-scalable anonymity payment scheme

In this section, we propose an anonymity self-scalable payment scheme. The new payment scheme has two main features, the first is that a consumer can have a higher level of anonymity by himself, the second is that the identity of a consumer can not be traced unless the consumer spends the same coin twice.

Our scheme includes two basic processes in system initialization (bank setup and consumer setup) and three main protocols: a new withdrawal protocol with which U withdraws electronic coins from B while his account is debited, a new payment protocol with which U pays the coin to S, and a new deposit protocol with which S deposits the coin to B and has his account credited. If a consumer wants to get a higher level of anonymity after getting a coin from the bank (withdrawal), she/he can contact the AP agent.

5.4.1 System Initialization

The bank setup and the consumer setup are described as follows, and the details of the shop setup are omitted. These two setup processes are similar to but not the same as that in Chapter 4.

Bank setup: (performed once by B)

Primes p and q are chosen such that $|p-1| = \delta + k$ for a specified constant δ, and

$p = \gamma q + 1$, for a specified small integer γ. Then a unique subgroup G_q of prime order q of the multiplicative group Z_p and generator g of G_q are defined. Secret key $x_B \in_R Z_q$ for a denomination is created. Hash function H from a family of collision intractable hash function is also defined. B publishes p, q, g, H and its public keys $Y = g^{x_B} (\text{mod} \quad p)$.

The secret key x_B is safe under the DLA. The hash function is used in payment transactions.

Consumer setup : (performed for each consumer U)
The bank B associates the consumer U with $I = g^{x_u} (\text{mod} \quad p)$ where $x_u \in G_q$ is the secret key of the consumer and is generated by U.

In system initialization, the communication complexity is $O(1)$ for the consumer only sends its account I of length l bits to the bank, and the computation complexity is $O(1)$. It requires only two exponentiations g^{x_B} and g^{x_u}.

After the consumer's account and the shop's account opening, we can describe the new payment scheme.

5.4.2 New off-line payment scheme

We now describe the new anonymity scalable electronic cash scheme which includes withdrawal, payment and deposit.

Withdrawal: As usual, an anonymous coin is a certified message, which embeds the public key of a consumer. In our scheme, the message is an encryption of this consumer's public key, using the public key Y of the bank.

Instead of using intricate zero-knowledge proofs to convince the bank of the va-

lidity of the encryption, the consumer shows some information to the bank including a signature. So the bank certifies the encryption with full confidence.

The consumer $I = g^{x_u}$ constructs a coin $c = (a = g^r, b = Y^r I^s)$ using the public key Y of the bank, where s is a secret key of the coin, which is kept by the consumer and r is a random number in Z_q. She/He also signs c together with the date, using his private key x_u and a Schnorr signature. She/He sends both to the bank together with r, I. Then the bank can check the correct encryption. With the signature of the coin and the date, only the legitimate consumer could have done it. After having modified the consumer's account, the bank sends back a certificate $Cert_c$. The consumer just has to remember $(r, s, Cert_c)$.

Anonymity scalability: The consumer can use the coin now without a higher anonymity since the bank can easily trace any transaction performed through the coin. This is because some information of the consumer such as $I, Cert_c$ has been known by the bank. To solve this problem, an AP agent is established to help the consumer to make a higher level of anonymity: the consumer can derive a new encryption of his identity in an indistinguishable way. However, the consumer needs a new certificate for a new issued ciphertext. The AP agent can provide this new certificate. Before certifying, the consumer requires both the previous coin $(c, Cert_c)$ and the proof of equivalence between the two ciphertexts. Details are described below.

The consumer contacts the AP agent if she/he needs to get a higher level of anonymity. The consumer chooses a random ρ and re-encrypts the coin:

$$c' = (a' = g^\rho a, b' = Y^\rho b).$$

1. The consumer generates a Schnorr signature S on $m = h^\rho$ using the secret key

x_u. Because of S, the consumer is not able to deny his knowledge of ρ later. Furthermore, nobody can impersonate the consumer at this step, since the discrete logarithm x_u of I is required to produce a valid signature. So there is no existential forgery.

2. The consumer also provides a designated -verifier proof of equality of discrete logarithms

$$log_h m = log_g(a'/a) = log_Y(b'/b). \tag{5.1}$$

3. The consumer finally sends c, c', S, m to the AP agent.

4. The AP agent checks the certificate $Cert_c$ on c, the validity of the signature S on the message m, then certifies c' and sends back a certificate $Cert_{c'}$ to the consumer.

After these processes the consumer gets a new certified coin $c' = (a' = g^\rho a, b' = Y^\rho b)$ and a new certification $Cert_{c'}$ which is now strongly anonymous from the point of view of the bank. The AP agent has to keep (c, c', m, S) to be able to prove the link between c and c', with the help of the consumer.

In the withdrawal process, the communication complexity is $O(1)$ since the consumer sends c, I and a signature to the bank and the bank returns $Cert_c$ to the consumer, six exponentiations are required in the withdrawal, four are from the consumer and two from the bank. Six exponentiations are required in the scalable anonymity providing process, four are from the consumer and two from the AP agent.

Following the process, the AP agent can also give many smaller new coins for an old one since the amount of new one can be embedded in the certificate $Cert_{c'}$.

Payment: (performed between the consumer and the shop over an anonymous channel)

When a consumer possesses a coin, she/he can simply spend it at shops: proves the knowledge of the secret key (x_u, s) associated with the coin c or c'. This proof is a signature $S = (e, u, v, t_1)$, which has shown in subsection 3.3, of the new certificate $Cert_{c'}$, purchase, date, etc with the secret key (x_u, s) associated to the coin to the receiver (which is later forwarded to the bank). Since the signature S of the message includes the current time which can not be changed and needs the secret key (x_u, s), only the consumer can use the coin. This means the shop can not pay the coin to another shop. This can prevent the shop using the coin sent by the consumer, otherwise, the shop can frame the user.

In payment transactions, the communication complexity is $O(1)$ for the consumer sending c and a signature $S = (e, u, v, t_1)$ to the shop. There are five exponentiations for the signature.

Deposit: (The receiver deposits a coin to a bank)

Since the system is off-line, the shop will send the payment transcript to the bank B later. The transcript consists of the coin c or c' (if the consumer applied a higher level of anonymity), the signature and the date/time of the transaction. The bank verifies the correctness of payment and credit the coin into shop's account.

In the deposit, the communication complexity is $O(1)$ because the shop sends the consumer's response c, and signature $S = (e, u, v, t_1)$ to the bank. The computation complexity is $O(1)$, since it only verifies whether c or c' was used before or not.

Untracebility: The receiver (shop) deposits the coin into its bank's account with a transcript of the payment. If the consumer uses the same coin c twice, then the

consumer is traced: two different receivers send the same coin c to the bank. The bank can easily search its records to ensure that c has not been used before. If the consumer uses c twice, then the bank has two different signatures. Thus, the bank can isolate the consumer and trace the payment to the consumer's account I.

In the new scheme, the communication complexity is $O(1)$, and required exponentiations are eighteen which is less than that in [79]. So it is quite efficient.

5.5 Security Analysis

We analyze the security of the system in this section. It includes how the system can preserve the requirements of a secure e-cash system and how to prevent "perfect crime" [99]. The "perfect crime" is a new problem in electronic payment, since users of coin have may been forced by criminals such as killed or kidnapped. Criminals want to use the illegal money from users. Our new payment scheme can stop criminals using the money.

5.5.1 Payment scheme security

As introduced in chapter 4, an off-line e-cash scheme is secure if the following requirements are satisfied:

1. Unreusable;
2. Unexpandable;
3. Unforgeable;
4. Untraceable;

The security in the e-cash scheme is based on the hardness of Discrete Logarithms [126] and hash functions. The system preserves the above four requirements.

Unreusable: The user owns two coins which represent the same money (the old and the new coins), but can exchange or spend both of them. The identity of users can be found when the old or the new coins are used twice or they are used separately. We analyze what will be happened if users use the higher level anonymous coin (the new coin), and omit the case of users use the old coin twice. This is because these two cases are similar.

When a consumer spends the new coin c' with the new certificate $Cert_{c'}$,she/he hands over the coin together with a signature $S = (e, u, v, t_1)$ to a shop. If the consumer uses a coin twice, then there are two signatures $S_1 = (e_1, u_1, v_1, t_{11})$ and $S_2 = (e_2, u_2, v_2, t_{12})$, where

$$u_1 = k_1 - (r + \rho)e_1 (mod\, p),\ v_1 = s - x_u e_1 (mod\, p).$$

$$u_2 = k_2 - (r + \rho)e_2 (mod\, p),\ v_2 = s - x_u e_2 (mod\, p).$$

Then $(v_2 - v_1)/(e_1 - e_2) = x_u$, this is the secret key of the consumer I. This means a coin in the new scheme cannot be reused. If the consumer uses the old and the new coin separately, there are two signatures, $S_1 = (e_1, u_1, v_1, t_{11})$ for the new coin and $S_2 = (e_2, u_2, v_2, t_{12})$ for the old one too, where

$$u_1 = k_1 - (r + \rho)e_1 (mod\, p),\ v_1 = s - x_u e_1 (mod\, p).$$

$$u_2 = k_2 - re_2 (mod\, p),\ v_2 = s - x_u e_2 (mod\, p).$$

Then $(v_2 - v_1)/(e_1 - e_2) = x_u$, this is the secret key of the consumer I. Therefore the consumer can not spend them separately. In a word, it is unreusable.

Untraceable: When a consumer constructs a coin, she/he uses the secret keys x_u and s, both are not shown to any other parties in the purchase process. So no one can trace the consumer from a coin.

Unforgeable: We first discuss whether the bank and the AP agent can forge a valid coin or not. Two requirements are necessary to produce a valid coin, the first is making a encryption $c = (a = g^r, b = Y^r I^s)$ of I, the second is using the secret key x_u of the consumer to sign a Schnorr signature of c together with the current time. The bank can do the first one but can not do the second one since it does not know the secret key x_u. This means the bank can not forge a valid coin. Similarly, the AP agent has no possibility to forge a valid coin. The AP agent knows c, c', S, m, but does not know how to sign the Schnorr signature S of the $m = h^\rho$. This is because the secret key (r, x_u) of the consumer has to be used in the signature S. So the AP agent can not forge a valid coin either. It should be noted that even though both the bank and the AP agent know a valid coin, they can not use it. This is because the signature $S = (e, u, v, t_1) = \Im((r, x_u), m)$ on the message m in the payment process can only be produced by the user. The message m includes the current time, purchase and the coin and so on. Therefore the bank, AP agent and shop can not use the coin even they get it. So only the user can use the coin.

As already seen, the secret key x_u of a consumer is never revealed, only used in some signatures. Any consumer is therefore protected against any impersonation, even from a collusion of the bank, the AP agent, and the shop. Only the consumer can construct a valid coin since there is a undeniable signature embedded in the coin. To prevent the bank from framing the consumer as a multiple spender in the scheme, we use digital signature I^s for s which is known only by the consumer. Then the system is unforgeable.

Unexpandable: For a legal consumer and a valid coin, the secret key x_u and the random number s are never shown to others at anytime. Furthermore, usually, the random number s is changed for different coins. With n withdrawal proceedings, the random number s is changed n times. Then, no one can compute $(n+1)th$ distinct and valid coins even they see n withdrawal proceedings.

We have seen the system is secure under the definition in [44] and no other parties can frame the user even they do cooperation. Next we discuss how to prevent the "perfect crime" by the system.

The aim of the criminal in the "perfect crime" is to get money from the bank and use it later. We show the criminal can not use the money even they get it. The user is found when a criminal forces a user to get the money of the user. The user's identity is found by the bank, and then the criminal can not withdrawal coins from the bank. The bank can also stop the criminal to use the money of the user even if it has been gotten by the user. This is because the bank can trace coins from the identity of the user and then send a warning message to the AP agent and shops. Either the AP agent or the shops do not accept the coins which can not be used anymore.

5.6 Comparisons

In this section, we compare the new scheme with the proposed approach in [79]. The computation complexity of the protocols is better than that in [79]. The main processes of David Pointcheval [79] are below.

Registration: The registration of a user is certified by a Certification Authority.

Withdrawal: Users construct coin using the public key of a Revocation Centre. So the Revocation Centre can trace users at any time even users have not spent coin twice.

Self-Scrambling Anonymizer: Users contact another third party (likes AP) to certify his message, and the latter provides a new certified coin to users after verifying the message.

Spending: Users send a coin and a signature of the purchase, date etc, with the secret key associated to the coin to the payee.

Revocation: The identity of users can be traced by the Revocation Centre at anytime. The Revocation Centre has to decrypt the coin. Therefore, the identity of users can be known even the coin does not be spent twice.

The phase of the Self-scrambling anonymity in [79] requires 10 exponentiations from the user point of view and 11 from the Anonymity Provider's point of view. In the protocol, the phase of the scalable anonymity required four exponentiations from the user point of view and two from the Anonymity Provider's point of view. Moreover, only the double spending user will be found by a simple linear computation, do not need description the coin. These show the new protocol is more efficient.

5.7 An example

In this section, we give a simple example and analyze two different purchase procedures. We show how to use the new e-cash for Internet purchases and how to get some smaller coins from the AP agent. As a result, we see the efficiency of the payment protocol.

5.7.1 An example

This example shows the main steps in the e-cash scheme. We omit the details of two undeniable signatures in withdrawal and scalable anonymity process, because they are only used for verifying the user. For simplicity, module 47 which has been used in the computation below is omitted in the expression.

Bank setup

Suppose $(p, q, \gamma, k) = (47, 23, 2, 4)$, then $G_q = \{0, 1, 2, ..., 22\}$ is a subgroup of order 23. $g = 3$ is a generator of G_q. The bank's secret key $x_B = 4$ and hash function $H(x, y) = 3^x * 5^y$. The bank publishes $H(x, y)$ and $\{p, q, g\} = \{47, 23, 3\}$. The public key of the bank is $Y = g^{x_B} = 34$.

User setup

We assume the secret of a user is $x_u = 7$ and the user sends $I = g^{x_u} = 32$ to the bank. After checking some things like social security card or drive license, the bank authorizes the user (consumer) with I.

After the bank setup and the user setup, the user can do purchase.

Withdrawal

The user chooses $(r, s) = (2, 3)$ and computes $c = (g^r, Y^r I^s) = (9, 2)$, then signs a Schnorr signature S for the message $m = (c, t)$, where t is the current time. The user sends $c = (9, 2)$ and S to the bank, the latter sends back a certificate $Cert_c$.

The user contacts the AP agent if she/he needs a high level of anonymity, or uses the coin in a shop directly (See Payment). The user and the AP agent follow the processes below. We suppose $h = 37$ is a public number.

Anonymity scalability

The user re-encrypts the coin c, chooses $\rho = 4$ and computes $c' = (a' = g^{\rho}a, b' = Y^{\rho}b) = (24, 14)$ and signs a Schnorr signature S on $m = h^{\rho} = 36$. Finally, the user sends (c, c', S, m) to the AP agent. The latter verifies the Schnorr signature S and the equation (1), and sends a certificate $Cert_{c'}$ to the user if they are correct.

Since the new coin $c' = (24, 14)$ and its certificate $Cert_{c'}$ has no relationship with the bank, the user has a high anonymity.

Payment

The user signs a signature $S = (e, u, v, t_1)$ of a message m which includes $c', Cert_{c'}$ and purchase time etc to prove the ownership of the new coin. For convenience, we assume $m = 11$. The user chooses $k = 5$ then computes $t = Y^k g^s = 19$, $e = H(m, t) = 40$, $u = 18, v = 5, t_1 = 28$.

The shop computes $t' = 15$ who is convinced that the user is the owner of the coin if the equation of $t' = tt_1$ and the signature S are successful. She/He does not know who is the user.

Deposit

The bank puts the money into the shop's account when the checking of the coin $C' = (24, 14)$ and the signature $S = (e, u, v, t_1) = (40, 18, 5, 28)$ are correct. The shop can also see that the money in his account is added.

5.7.2 Purchase procedures

Purchase procedure 1

In purchase procedure 1 a consumer decides how much money should be paid to

the shop, withdraws the money from the bank, and pays it to the shop.

1. *Consumer to shop*: The consumer wants to buy some goods in a shop, so contacts the shop for the price.

2. *Consumer to bank*: The consumer gets the money from the bank, the amount being embedded in the signature.

3. *Anonymity scalability*: If the consumer wants to maintain higher level of anonymity, she/he can ask the AP agent to certify a new coin which can be then used in the shop.

4. *Consumer to shop*: The consumer proves to the shop that she/he is the owner of the money, and pays it to the shop. Then the shop sends the goods to the consumer.

5. *Shop to bank*: The shop deposits the e-cash in the bank. The bank checks the validation and that there is no double spending of the coin. The bank transfers the money to the shop's account.

Purchase procedure 2

In purchase procedure 2 is that: the consumer does not have to ask the bank to send money since the consumer already has enough e-cash in his "wallet". All she/he needs to do is to get some smaller e-cash from the AP agent to pay the shop.

There are 4 steps in the purchase procedure 2. They are: (1) *consumer to shop*; (2) *consumer to AP agent*; (3) *consumer to shop* again and (4) *shop to bank*. Step 2, *consumer to AP agent* is different from the step 3 in procedure 1 and another three steps are similar to that in procedure 1. Therefore we focus only on step 2

consumer to AP agent. It should be noted that electronic-cash is a digital message and a certification. We say that the AP agent can provide certificates of coins then provide a service in changing small coin.

Consumer to AP agent: The consumer advises the AP agent of the amount of money to pay the shop from his wallet. She/He can ask the AP agent to make some smaller coins. By doing this, the consumer can also get a higher level of anonymity. After checking the old money sent by the consumer, the AP agent creates some new coins of an equivalent value to the original coin. One of these new coins can be used in the shop.

We have already seen that the consumer can keep money in his wallet or get money from the bank. In both purchase procedures 1 and 2 most computations are done by the consumers, so the system is very convenient for Internet purchases.

5.8 Conclusions

In this chapter, a new electronic cash scheme is designed to provide different degree of anonymity for consumers. Consumers can choose their level of anonymity. They choose a low level of anonymity if they want to spend coins directly after withdrawing them from the bank. Consumers can achieve a high level of anonymity through the AP agent without revealing their private information and are more secure in relation to the bank because the new certificate of a coin comes from the AP agent who is not involved in the payment process. This system does not need a trusted party to manage consumers' identities. In this new model, we have shown how to derive an efficient and untraceable cash scheme based on the variation of coins. It is an off-line scheme with low communication and computation. With its scalable

anonymity, the new payment protocol can effectively prevent eavesdropping, tampering, impersonation and "perfect crime". Finally, we have compared the new payment protocol with another one to show its efficiency.

Chapter 6

Role Based Access Control and its applications

This chapter presents basic definitions of role based access control RBAC such as user-role assignment, permission-role assignment and role-role assignment and so on. The advantages and significance of RBAC are discussed. Based on the payment model designed in the previous chapter, the use of RBAC to manage the electronic payment system is analyzed.

Most information in this chapter has been published in [113, 111, 116].

6.1 Introduction

With people's increased consciousness of the need for electronic commerce to protect their private information and to provide security of applications, system administrators are continuing to implement access control mechanisms and retain a critical and complex aspect of security administration. Traditional administrations of access control are mandatory, discretionary and role-based access control. Mandatory access controls (MAC) restrict access to data based on varying degrees of security requirements for information contained in the objects. Information is associated with

multi-level security requirements with labels such as TOP SECRET, SECRET, and CONFIDENTIAL [5]. An assigned right cannot be changed and modifications are permitted only to administrators. Users may need to register on a number of different servers of different operating system types, various databases and multiple business applications. Furthermore, an object classification reflects the sensitivity of the information contained in the object, that is, the potential damage that may come from unauthorized disclosure of information. Registration of each user with each facility is needed to control and prevent unauthorized use. Managing a system with MAC is a challenging task, especially when dealing with the changes on user positions and other access rights. Discretionary access controls (DAC) allow users to grant or revoke access to any authority under their control without the intercession of a system administrator [39]. Access rights to resources are based on the identity of persons and/or groups to which they belong. When the number of users increases, the management is costly. DAC grants authorization or privileges to users directly, authorized statically when they set up an account. Though it is convenient for users to pass on the authorization directly to other users, it brings a serious security problems. For example, when a user passes on some access controls to another user, it may change the level of access privilege of the second user who may then able to access or derive high level information based on the level of access control gained.

The concept of role based access control RBAC started with multi-user and multi- on-line application systems pioneered in the early 1970s [70, 16, 101, 13]. The major notion of RBAC is that permissions are assigned to roles and users are associated with appropriate roles. Users cannot associate with permissions directly. In other words, RBAC is described in terms of individual users being associated

with roles as well as roles being associated with permissions (Each permission is a pair of objects and operations). As such, a role is used to associate with users and permissions. A user in this model is a human being. Roles are created for the various job functions in an organization and users are assigned roles based on their authority and qualifications. Users can be easily reassigned from one role to another. Roles can be granted new permissions as new applications and systems are incorporated and permissions can be revoked from roles as needed.

Permission is an approval of a particular operation to be performed on one or more objects. The relationship between roles and permissions is shown in Figure 6.1, arrows indicate a many to many relationship (that is, a permission can be associated with one or more roles, and a role can be associated with one or more permissions). As shown in the Figure, RBAC has the capability to establish relations between roles as well as between permissions and roles and between users and roles. For example when two roles are established as mutually exclusive then the same user is not allowed to take on both roles. This problem may happen in user-role assignment. Another example is for permission-role relations, conflicting permissions cannot be assigned to the same role. Therefore in a bank, the permission for approving loan and that of funding loan are conflicting, these two permissions cannot be assigned to a role. In role-role assignment, roles have inheritance relations whereby one role inherits permissions assigned to another role. These relations between users and roles, permissions and roles and between roles and roles are used to establish security policies that include separation of duties and delegation of authority. The security policy of the organization determines role membership and the allocation of each roles capabilities.

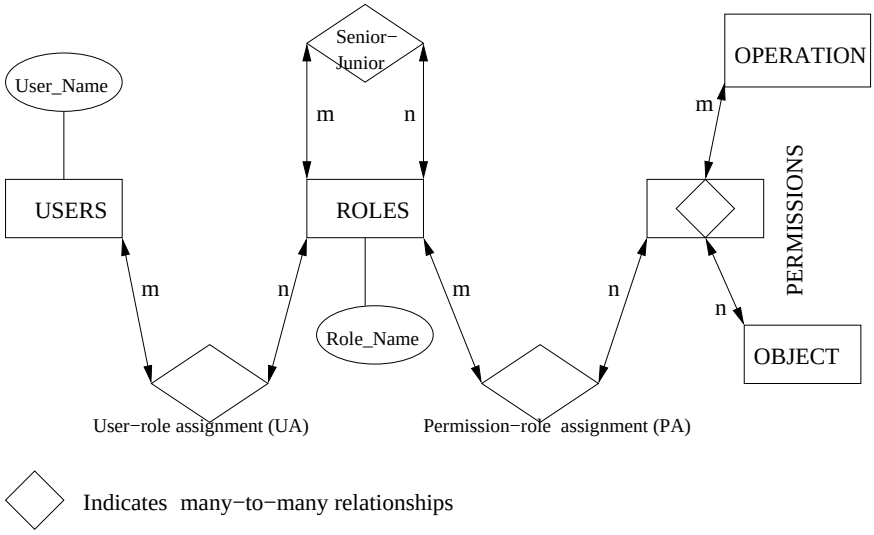

Figure 6.1: RBAC relationship

With RBAC it is possible to predefine role permission relationships, which makes it simple to assign users to the predefined roles. It can also be difficult, without RBAC, to determine what permissions have been authorized to what users.

There are three advantages of RBAC management. Firstly, it is much easier to manage a system using RBAC. In RBAC, a security administrator adds transactions to roles or deletes transactions from roles, where transactions can be a program object associated with data [39]. Security problems are addressed by associating programming code and data into a transaction. Access control does not require any checks on the user's or the program's right to access a data item, since the accesses are built into the transaction. Secondly, RBAC can reduce administration cost and complexity [89]. Usually, there is a relationship between the cost of administration and the number of associations which must be managed in order to administer an access control policy. The larger the number of associations, the more cost and more error prone the access control administration is likely to be, but the use of RBAC reduces the number of associations to be managed. Thirdly, RBAC is better than a typical access control list (ACL) model [62]. RBAC can authorize and

audit capabilities so that people are simply assigned new roles while they change responsibilities. This allows for the authorities of a person to be easily documented. By contrast, in ACL, the entire set of authorities must be searched to develop a clear picture of a person's rights because ACLs only support the specification of user/permission and group/permission relationships.

The important feature of RBAC is policy neutral. However, it directly supports three well known security principles: least privilege, separation of duties and data abstraction [90]. RBAC gives support to the least privilege because RBAC can be configured so only those permissions required for the tasks are assigned to roles. Separation of duties is performed by ensuring that mutually exclusive roles must be invoked to complete a sensitive task, for instance, both an accounting manager and account clerk are required to participate in issuing a check. Data abstraction is achieved by means of abstract permissions such as credit and debit for an account object, rather than the read, write, execute permissions typically provided by the database system.

6.2 Administrative issues in RBAC

It has shown that there are many components to RBAC. RBAC administration is therefore multi-faceted. The components include the issues of assigning users to roles, assigning permissions to role, and assigning roles to roles. This section analyzes these issues.

6.2.1 User-role assignments

The user-role assignment (URA) model was originally defined by Sandhu and Bhamidipati [93]. Figure 6.2 shows the regular roles in a shop and an administrative role.

There is a junior-most role SHOP to which all employees belong and a senior-most role MANAGER. Roles AUDITOR and SELLER inherit the permissions of role SHOP while MANAGER inherit the permissions of roles AUDITOR and SELLER. The member of administrative role ShopSO are authorized to modify membership in the roles of the figure.

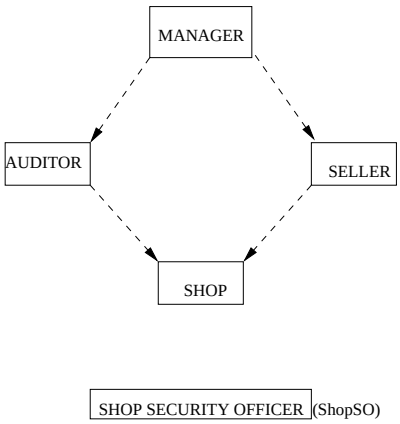

Figure 6.2: Administrative role and role Relationships in a shop

There are two kinds of work in user role assignment. The first is to specify what membership between user and role can be modified by an administrative role. The second is how many users can be assigned to a role. For example, user-role assignment requires that the administrative role ShopSO can assign users to the roles AUDITOR, SELLER and MANAGER, but these users must already be members of the shop, that is, the role SHOP. This is an example of a prerequisite role. More generally user-role assignment allows for a prerequisite condition [33, 54]. The prerequisite conditions is used later in RBAC applications.

A *session* is a mapping of a user to possibly many roles. In other words, users establish sessions during that users activate some roles. Both users and sessions are

important conceptions in RBAC management. The distinction between a user and a session is a fundamental aspect of RBAC and consequently arises in RBAC. To achieve the principle of least privilege a user should be allowed to login to a system with only those roles appropriate for a given session. With constraints it may not be possible for a user to activate all their roles simultaneously. For example, a constraint limits that two roles can be assigned to the same user but cannot be simultaneously activated in a session. Therefore, a person may be qualified to be a car driver and pilot but he/she can activate at most one of these roles at any time. The person cannot perform a single session with all the persons roles activated. On the other hand, for the security reason, RBAC has dynamic separation duty constraint related to roles. In the Figure 6.2, role AUDITOR and role SELLER has dynamic separation duty relationship. These two roles can be assigned to a person but cannot activate in a session.

The opposite operation of assignment is revocation. There are two kinds of revocations. One is weak revocation and the other one is strong revocation. The revocation operation is said to be weak because it only uses to the role that is directly revoked. On the other hand, strong revocation applies to all roles that include senior roles. Strong revocation cascades upwards in the role hierarchy. For example, suppose Bob is a member of AUDITOR and SHOP. If Alice has administrative role ShopSO revokes Bobs membership from SHOP, he continues to be a member of the senior role AUDITOR and therefore can still use permissions of role SHOP. If Alice strongly revokes Bob's membership from SHOP, his membership in AUDITOR is also revoked. Both revocations are analyzed in the applications of RBAC.

6.2.2 Permission-role assignments

Permission-role assignment applies to assign permissions to roles and revoke permissions from roles. From the perspective of a role, permissions character is similar to users character. Both of users and permissions are basic entities that are brought together by roles. There are two aspects in permission-role assignment. One is that not all permissions can be assigned to roles by an administrator. For example in Figure 6.2, only permissions which belong to roles SHOP and AUDITOR can assign to role SELLER. It means there are prerequisite conditions for the operations of permission-role assignment.

Prerequisite conditionp is an expression using Boolean operators $\wedge$ and $\vee$ on terms of the form x and $\bar{x}$ where x is a role and $\wedge$ means "and", $\vee$ means "or". A prerequisite condition is evaluated for a permission p by interpreting x to be true if $(\exists x' \geq x), (p, x') \in PA$ and $\bar{x}$ to be true if $(\forall x' \geq x), (p, x') \notin PA$, where PA is a set of permission-role assignments. $\diamond$

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign a permission to a role. The relation of **Can-assignp** $\subseteq AR \times CR \times 2^R$ provides what permissions can be assigned by administrators with prerequisite conditions, where AR is a set of administrative roles.

For example, the meaning of *Can-assignp* (x, y, Z) is that a member of the administrative role x can assign a permission whose current membership satisfies the prerequisite condition y to be a member of roles in range Z. Permission-role assignment (PA) is authorized by *Can-assignp* relation.

6.2.3 Role-role assignment

There are three kinds of roles. They are Ability-roles, Group-roles and UP-roles [90].

Ability-roles (ABR) are roles that can only have permissions and other ability-roles as members.

Group-roles (GR) are roles that can only have users and other group-roles as members.

UP-roles (UPR) are roles that have no restriction on membership, in other words, their membership can include users, permissions, group-roles, ability-roles and other UP-roles.

No role can be in two different kinds of roles. It means that the three kinds of roles are mutually disjoint. An Ability-role is a set of permissions that should be assigned as a single unit to a role. Administrators can treat these permissions as a single unit ABR. Assigning ABR to role likes assigning permissions to roles. It is convenient for developers to package basic permissions into an ABR with a task. For example, opening an account in bank includes different permissions such as show identity, check identity, save necessary information in bank system and so on. These permissions are same for everyone. Therefore, they can be setup as an ABR.

Similar to an Ability-role, a Group-role is a set of users who are assigned as a single unit to a role. A GR can be viewed as a team in system application. It can simplify system management. For example, a developing team as a GR which can be assigned to the direct role of the team.

Can-modify: $AR \rightarrow 2^{UPR}$ defines which administrative roles can create and

delete roles, assign and revoke memberships between roles.

Administrative.role	UP- Role Range
ShopSO	[SHOP, MANAGER)

Table 6.1: Example of Can-modify

Table 6.1 shows an example of *Can-modify*. The meaning of *Can-modify (ShopSO, [SHOP, MANAGER))* is that a member of the administrative role SHOP or a member of an administrative role that is senior to SHOP can create and delete roles in the range [SHOP, MANAGER) except for the endpoints of MANGER and can modify relationships between roles in the range [SHOP, MANAGER).

6.2.4 Duty separation constraints

Separation of duty (SOD) relations are used to enforce conflict of interest policies that prevent users from processing conflicting authorities for their positions. As a security principle, SOD has been widely recognized for its wide application [103], [21]. It ensures that failures of commission within an organization can be caused only as a result of collusion among individuals. To minimize collusion, different skills are assigned to separate tasks required in the performance of a system. SOD can protect that fraud and major errors if no deliberate collusion of multiple users. There are two types of SOD in RBAC. One is static separation of duty, and the other one is dynamic separation of duty.

Static Separation of Duty

In a role-based system, a user may authorize permissions associated with conflicting roles. Static separation of duty (SSD) limits constraints on the assignment

of users to roles to prevent this form of conflict. RBAC models have defined SSD relations with respect to constraints on RBAC management. That is no user can be simultaneously assigned to both roles in SSD. SSD relations may exist within hierarchical RBAC. When applying SSD relations in the presence of a role hierarchy, it should be ensured that user inheritance does not undermine SSD policies. For example, the role bank supervisor inherits the role of accounts clerk, and the clerk has an SSD relationship with the role of billing clerk, then supervisor also has an SSD relationship with the billing clerk.

Dynamic Separation of Duty

Dynamic separation of duty (DSD) relations, like SSD relations, are intended to limit the permissions that are available to a user. However DSD relations differ from SSD relations by the context. SSD relations define constraints on a user's total permission space. DSD properties specify the availability of the permissions over a user's permission space that can be activated in a user's sessions. DSD provides support in security management policy for the principle of least privilege in that each user requires different permissions at different times. SSD relations provide the capability to address potential conflict issues that a user is assigned to a role. DSD allows two or more roles that do not create a conflict of interest when acted on independently to be assigned to a user. DSD concerns activated simultaneously. In a bank, for instance, a user may be authorized for both the roles of Teller and Manager, where the Manager is allowed to acknowledge corrections to a Teller's open cash drawer. If a person acting in the role Teller attempted to switch to the role Manager, DSD relation would require the user to drop the Teller role. A conflict of interest situation does not arise as long as the same user is not allowed to be both of these roles at the same time.

However, RBAC cannot compel systems to use these principles. The security officer can configure RBAC so it implements these principles. In the next two sections, applications of user-role assignment and permission-role assignment including SSD, DSD are discussed. The user-role assignments for the flexible payment scheme are described, and then followed by its permission-role assignments.

6.3 User-role assignments for a flexible payment scheme

Using role-based access controls (RBAC) to manage user-role assignments for electronic payments is one of the most challenging problems. There are two types of problems which may arise in user-role assignment with RBAC. One is related to authorization granting process. Mutually exclusive roles may be granted to a user and the user may have or derive a high level of authority. Another is related to authorization revocation. When a role is revoked from a user, the user may still have the role since role hierarchies. This section presents user-role assignments for a flexible electronic payment scheme. To solve these problems, we first analyze the duty separation constraints of the roles and role hierarchies in the scheme, then discuss granting a role to a user, weak revocation and strong revocation for the scheme. The aim of this section is to provide a way to manage electronic payments with RBAC.

Recently, role-based access control (RBAC) has been widely used in database system management and operating system products. RBAC involves individual users being associated with roles as well as roles being associated with permissions (each permission is a pair of objects and operations). As such, a role is used to associate users and permissions. A user in this model is a human being (for example, a staff

member in a bank). A role is a job function or job title within an organization associated with authority and responsibility (for example, role BANK manages money for consumers). Permission is an approval of a particular operation to be performed on one or more objects. There are many relationships between users and roles, and between roles and permissions as shown in Figure 6.1. Assigning people to tasks is a normal managerial function. The assignments of users to roles is a natural part of assigning users to tasks. Hence, user-role assignment is a basic issue of RBAC.

Many RBAC practical applications have been implemented [6], [40] and [89] since 1993. However, there has been little research done on the usage of RBAC in payment scheme management [91]. This section analyzes duty separation constraints such as role-role relationship in a payment scheme and then discuss user-role assignment for the payment scheme.

A scalable anonymity electronic cash scheme has been published [110]. It differs from other electronic payment schemes [79] [20] and it is significant because consumers in the scalable scheme can get a required anonymity without showing their identities to any third party. This scheme is a benefit for consumers who are worried about their identities being traced by banks. However, how to manage the scheme with RBAC is a remained challenging problem.

Duty separation constraints of the scheme

Sandhu and Bhamidipati developed a model called URA97 in which RBAC is used to manage user-role assignment [93]. It did not discuss user-role assignments with electronic commerce. We consider the relationships of the four roles. Duty separation constraints are role-role associations indicating conflicts of interest. Static separated duty (SSD) specifies that a user cannot be authorized for two different

roles while dynamic separated duty (DSD) specifies that a user can be authorized for two different roles but cannot act simultaneously in both.

Role hierarchies specify which role may inherit all of the permissions of another role. In Figure 6.3, for example, since all staff in the AP agent, the bank and the shop are employees, their corresponding roles inherit the employee role. The role AP, SHOP and BANK have DSD relationships with the role CONSUMER. This indicates that an individual consumer cannot play the roles of AP, SHOP or BANK simultaneously. The staff in these three participants have to first log out if they want to register as consumers. For example, a consumer, who is a staff member of the AP agent and is able to act the role AP, can ask the AP agent to help him to get a coin with a high level anonymity. But as a consumer, she/he cannot give herself/himself a new certificate $Cert_{c'}$ of a coin when she/he works for the AP agent. Another staff member of the AP agent should do the job for this person.

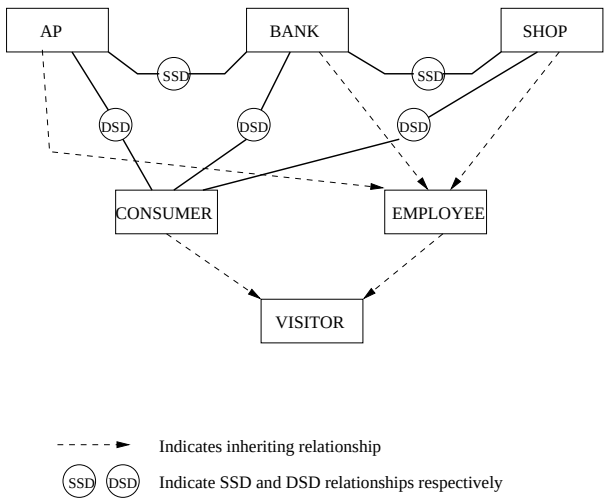

Figure 6.3: The relationships of the roles in the scheme

The role AP has an SSD relationship with BANK. This is because the duty of the

AP is to help a consumer to get a coin with a high level of anonymity. The BANK knows the old coin $c = (g^r, Y^r I^s)$ and its certificate $Cert_c$. The AP sends the new certificate $Cert_{c'}$ of the new coin $c' = (g^{r+\rho}, Y^{r+\rho} I^s)$ to the consumer. The role BANK knows the new certificate $Cert_{c'}$ and new coin c' if the same staff member from the AP agent and the bank processed the coin for the consumer. If this occurs, the consumer cannot have a coin with the required anonymity because the BANK has known the new coin. The SHOP also has an SSD relationship with the BANK since the BANK verifies the payment as well as depositing the coin to the shop's account. The SSD relationship is also a conflict of interest relationship like the DSD relationship but much stronger. If two roles have a SSD relationship, then they may not even be authorized to the same individual. Thus, the role AP, BANK, and SHOP may never be authorized to the same individual.

User-role assignments

To discuss user-role assignments, we add a manager role (M1) etc in an AP agent, a manager role (M2) etc in a bank, a manager role (M3) etc in a shop and some administrative roles Senior Officer(SSO) etc in the system as shown in Figure 6.4 and Figure 6.5. A hierarchy of roles and a hierarchy of administrative roles are also shown in these two Figures. The roles in Figure 6.4 can be granted and revoked by the administrative roles in Figure 6.5.

Let $x > y$ denote role x is senior to role y with obvious extension to $x \geq y$. The notion of a prerequisite condition is a key part in the processes of user-role assignment [93].

Prerequisite condition is an expression using Boolean operators $\wedge$ and $\vee$ on terms of the form x and $\bar{x}$ where x is a role and $\wedge$ means "and", $\vee$ means "or ".

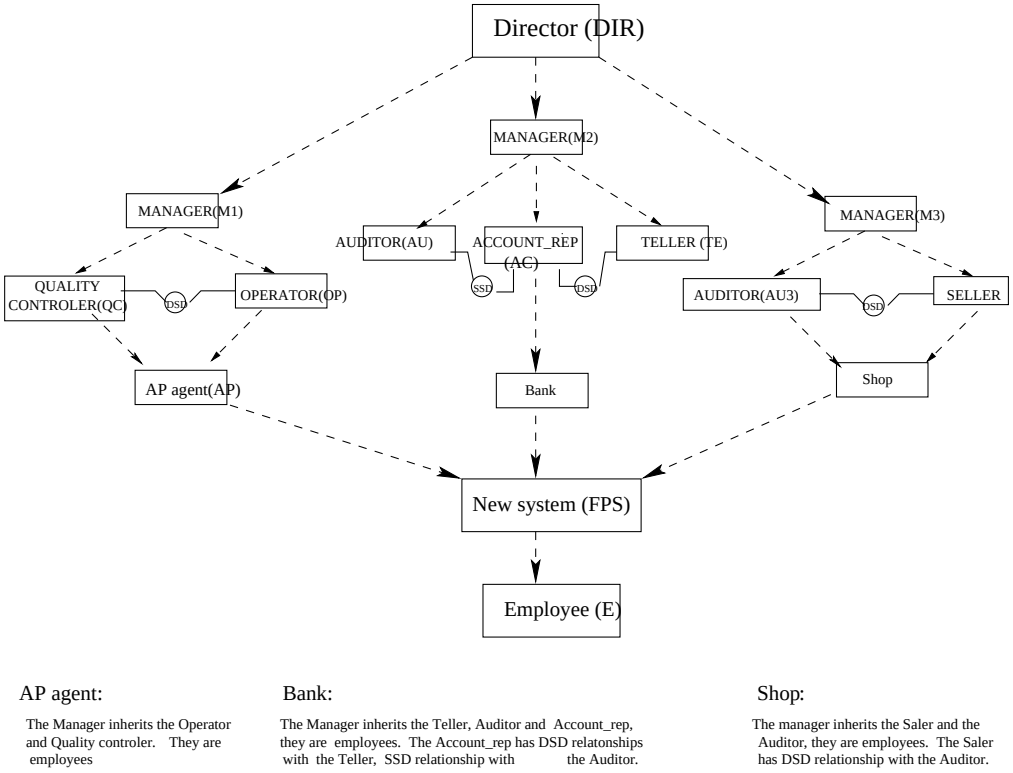

Figure 6.4: User_role assignment

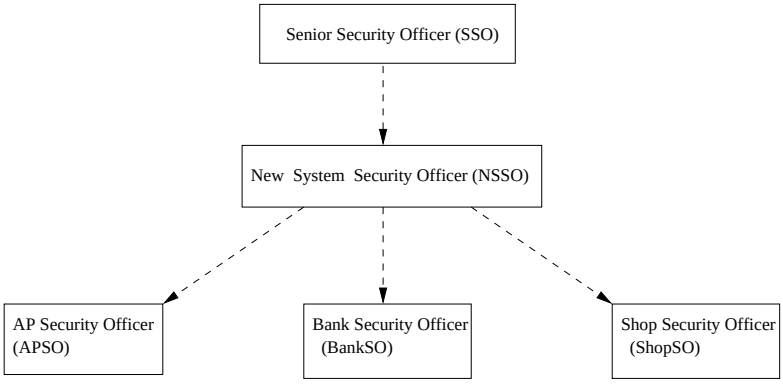

Figure 6.5: Administrative role assignment

A prerequisite condition is evaluated for a user u by interpreting x to be true if $(\exists x' \geq x), (u, x') \in UA$ and $\bar{x}$ to be true if $(\forall x' \geq x), (u, x') \notin UA$, where UA is a set of user-role assignments. $\diamond$

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign a role to a user. The relation of **Can-assign** $\subseteq AR \times CR \times 2^R$ provides what roles an administrator can assign with prerequisite conditions, where AR is a set of administrative roles.

Table 6.2 shows the *Can-assign* relation with the prerequisite conditions in the scheme. To identify a role range within the role hierarchy of Figure 6.4, we use the familiar closed and open interval notation.

$$[x, y] = \{r \in R | x \geq r \wedge r \geq y\}$$

$$(x, y] = \{r \in R | x > r \wedge r \geq y\}$$

$$[x, y) = \{r \in R | x \geq r \wedge r > y\}$$

$$(x, y) = \{r \in R | x > r \wedge r > y\}$$

Let us consider the APSO tuples (the analysis for BankSO and ShopSO are similar). The first tuple authorizes APSO to assign users with the prerequisite role FPS into members in the AP agent (AP). The second one authorizes APSO to assign users with the prerequisite condition $FPS \wedge \overline{OP}$ to be quality controllers (QC). Similarly, the third tuple authorizes APSO to assign users with the prerequisite condition $FPS \wedge \overline{QC}$ to be operators (OP). The second and third tuple show that the APSO can grant a user who is a member of the AP agent into one but not both of QC and OP. This illustrates how mutually exclusive roles can be forced.

Admin.role	Prereq.Condition	Role Range
APSO	FPS	[AP, AP]
APSO	FPS $\wedge\overline{OP}$	[QC, QC]
APSO	FPS $\wedge\overline{QC}$	[OP, OP]
APSO	QC $\wedge$ OP	[M1, M1]
BankSO	FPS	[Bank, Bank]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AU}$	[AC, AC]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AC}$	[AU, AU]
BankSO	FPS $\wedge\overline{AU} \wedge \overline{AC}$	[TE, TE]
BankSO	TE $\wedge AU\wedge$ AC	[M2, M2]
ShopSO	FPS	[Shop, Shop]
ShopSO	FPS $\wedge\overline{SELLER}$	[AUDITOR, AUDITOR]
ShopSO	FPS $\wedge\overline{AUDITOR}$	[SELLER, SELLER]
ShopSO	SELLER $\wedge$ AUDITOR	[M3, M3]
NSSO	FPS	(FPS, DIR)
SSO	E	[FPS, FPS]
SSO	FPS	(FPS, DIR]

Table 6.2: Can-assign

However, for the NSSO and SSO these are not mutually exclusive. The fourth tuple authorizes APSO to put a user who is a member of both QC and OP into a manager (M1). Of course, a user could have become a member of both QC and OP only by actions of a more powerful administrator than APSO.

There are related subtleties that arise in RBAC concerning the interaction between granting and revocation of user-role membership. A relation **Can-revoke** $\subseteq AR \times 2^R$ shows which role range administrative roles can revoke, where AR is a set of administrative roles. The meaning of *Can-revoke* (x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a user from any role $y \in Y$, where Y defines the *range of revocation*. Table 6.3 gives an example of the it Can-revoke relation. There are two kinds of revocations [93]. The first one is weak revocation, the second one is strong revocation.

Admin.role	Role Range
APSO	[AP, M1)
BankSO	[Bank, M2)
ShopSO	[Shop, M3)
NSSO	(FPS, DIR)
SSO	[FPS, DIR]

Table 6.3: Can-revoke

A user U is an *explicit member* of a role x if $(U, x) \in UA$, and U is an *implicit member* of role x if for some role $x' > x, (U, x') \in UA$. Weak revocation has an impact only on explicit membership. For weak revocation, the membership of a user is revoked only if the user is an explicit member of the role. Therefore, weak revocation from a role x has no effect when a user is not an explicit member of the role x. The following is an example of weak revocation for the flexible scheme where Alice and Bob are users.

Suppose Bob is an explicit member of role M1, QC, AU, AUDITOR, AP, FPS and E in the scheme. If Alice, with the activated administrative role APSO, weakly revokes Bob's membership from AP, he continues to be a member of the senior roles to AP since both M1 and QC are senior roles to AP, therefore he still has the permission of AP. It is necessary to note that Alice should have enough power in the session to weakly revoke Bob's membership from his explicitly assigned roles. For instance, if Alice has activated APSO and then tries to weakly revoke Bob's membership from M1, she is not allowed to proceed because APSO does not have the authority of weak revocation from M1 according to the *Can-revoke* relation in Table 6.3. Therefore, if Alice wants to revoke Bob's explicit membership as well as implicit membership from AP by weak revocation, she needs to activate SSO or NSSO and weakly revoke Bob's membership from AP, QC and M1.

Strong revocation requires revocation of both explicit and implicit membership.

Strong revocation of a user's membership in role x requires that the user be removed not only from explicit membership in x, but also from explicit (implicit) membership in all roles senior to x. Strong revocation therefore has a cascading effect up-wards in the role hierarchy.

In the scheme, for example, Bob is an explicit member of role M1, QC, AU, AP, AUDITOR, FPS and E. If Alice, with the activated administrative role SSO, strongly revokes Bob's membership from AP, then he is removed not only from explicit membership in AP, but also from explicit (and implicit) membership in all roles senior to AP. Actually, after the strong revocation from AP, Bob has been removed from M1, QC as well as AP. However, he still has a membership of FPS, AU, AUDITOR and E, since they are not senior roles to AP based on the role hierarchy of Figure 6.4. This brings about the same result as weak revocation from AP, QC, M1 by SSO. Note that all implied revocations upward in the role hierarchy should be within the revocation range of the administrative roles that are active in a session. For instance, if Alice activates APSO and tries to strongly revoke Bob's membership from M1, she is not allowed to proceed because M1 is out of the APSO's *Can-revoke* range in Table 6.3.

Weak revocation revokes explicit memberships only and strong revocation revokes both explicit and implicit memberships. Therefore a user may not have the permissions of a role if the user's membership is strongly revoked from the role.

6.4 Permission-role assignments with the payment scheme

The user-role assignment relation UA and permission-role assignment relation PA are many-to-many relations between users and roles, and between roles and permis-

sions as shown in Figure 6.1. Assigning permissions to roles is typically the province of application administrators. Thus a banking application can be implemented so credit and debit operations are assigned to a teller role. However, approval and funding operations cannot be assigned to a teller role since they are conflicting permissions. Users are authorized to use the permissions of roles to which they are assigned. This is the essence of RBAC [1].

Similar to user-role assignment, there are two types of conflicting problems that may arise in permission-role assignments. One is related to authorization granting process. Conflicting permissions may be granted to a role, and as a result, users with the role may have or derive a high level of authority. Another is related to authorization revocation. When a permission is revoked from a role, the role may still have the permission from other roles. This section discusses how to solve these problems with the payment scheme in [110]. The duty separation constraints of the roles and role hierarchies in the scheme has been analyzed in the last section, this section considers granting a permission to a role, weak revocation permissions and strong revocation permissions for the scheme.

There are a few applications with permission-role assignment [73, 72, 92]. For example, Sandhu and Bhamidipati developed an oracle implementation for permission-role assignment [92]. It does not discuss permission-role assignments for electronic commerce. We analyze permission-role assignment for the payment scheme in this section.

Granting and revocation models

RBAC administration encompasses the issues of assigning users to roles, assigning permissions to roles, and assigning roles to roles to define a role hierarchy. These

activities are all required to bring users and permissions together. In many cases, they are best done by different administrators. To analyze granting and revocation models, we add a manager role (M1) etc in an AP agent, a manager role (M2) etc in a bank, a manager role (M3) etc in a shop and some administrative roles Senior Officer(SSO) etc in the system as shown in Figure 6.4 and Figure 6.5. A hierarchy of roles and a hierarchy of administrative roles are also shown in these two Figures. Senior roles are shown towards the top of the hierarchies and junior are to the bottom. Senior roles inherit permissions from junior roles. Permissions can be granted to or revoked from the roles in Figure 6.4 by the administrative roles in Figure 6.5.

Let $x > y$ denote x is senior to y with obvious extension to $x \geq y$. The notion of a *prerequisite conditionp* is used to restrict on what permissions can be assigned to a role. A *prerequisite conditionp* is evaluated for a permission p by interpreting x to be true if $(\exists x' \geq x), (p, x') \in PA$ and $\bar{x}$ to be true if $(\forall x' \geq x), (p, x') \notin PA$, where PA is a set of permission-role assignments. It means that only permissions satisfy the conditions may be assigned to roles. On the other hand, whether an administrator can establish the relationship between permissions and roles depends on the relation of **Can-assignp** $\subseteq AR \times CR \times 2^R$. That means that permission-role assignment (PA) is authorized by *Can-assignp* relation.

For example, the meaning of *Can-assignp* $(NSSO, DIR, [M1, M1])$ in Table 6.4 is that a member of the administrative role NSSO can assign a permission whose current membership satisfies the prerequisite condition DIR to be a member of roles in range [M1, M1].

The motivation behinds the *Can-revokep* relation is in Figure 6.4 and Figure 6.5. Figure 6.4 shows that role E is junior-most to which all employees in the new

system and role Director (DIR) is senior-most to all employees. Figure 6.5 shows the administrative role hierarchy which co-exist with the roles in Figure 6.4. The senior-most role is the Senior Security Officer (SSO). Our interest is in the administrative roles junior to SSO. These consist of three security officer roles (APSO, BankSO and ShopSO) with the relationships illustrated in the Figure 6.5.

Based on the role hierarchy in Figure 6.4 and administrative role hierarchy in Figure 6.5, we define the *Can-revokep* relation shown in Table 6.4.

Admin.role	Prereq.ConditionP	Role Range
NSSO	DIR	[M1, M1]
NSSO	DIR	[M2, M2]
NSSO	DIR	[M3, M3]
APSO	FPS $\wedge\overline{OP}$	[QC, QC]
APSO	FPS $\wedge\overline{QC}$	[OP, OP]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AU}$	[AC, AC]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AC}$	[AU, AU]
BankSO	FPS $\wedge\overline{AU} \wedge \overline{AC}$	[TE, TE]
ShopSO	FPS $\wedge\overline{SELLER}$	[AUDITOR, AUDITOR]
ShopSO	FPS $\wedge\overline{AUDITOR}$	[SELLER, SELLER]

Table 6.4: Can-assignp

There are related subtleties that arise in RBAC concerning the interaction between granting and revocation of permission-role membership. A relation **Can-revokep**$\subseteq AR \times 2^R$ shows which permissions in what role range can be revoked by administrative, where AR is a set of administrative roles. The meaning of *Can-revokep* (x, Y) is that a member of the administrative role x (or a member of an administrative role that is senior to x) can revoke membership of a permission from any role $y \in Y$, where Y defines the *range of revocation.* Table 6.5 gives an example of the *Can-revokep* relation.

Admin.role	Role Range
NSSO	[FPS, DIR)
APSO	[AP, M1)
BankSO	[Bank, M2)
ShopSO	[Shop, M3)

Table 6.5: Can-revokep

A permission P is an *explicit member* of a role x if $(P, x) \in PA$, and P is an *implicit member* of role x if for some role $x' < x, (P, x') \in PA$. Weak revocation has an impact only on explicit membership. For weak revocation, the membership of a permission is revoked only if the permission is an explicit member of the role. Therefore, weak revocation from a role x has no effect when a permission is not an explicit member of the role x. We show an example of weak revocation for the flexible scheme.

Suppose P is an explicit member of role M1, QC, AU, AP and FPS in the scheme. If Alice, with the activated administrative role APSO, can weakly revoke P's membership from QC, P continues to be an implicit member of QC since AP is junior to QC and P is an explicit member of AP. It is necessary to note that Alice should have enough power in the session to weakly revoke P's membership from explicitly assigned roles. For instance, if Alice has activated APSO and then tries to weakly revoke P from FPS, she is not allowed to proceed because APSO does not have the authority of weak revocation from FPS according to the *Can-revokep* relation in Table 6.5. Therefore, if Alice wants to revoke P's explicit membership as well as implicit membership from QC by weak revocation, she needs to activate NSSO and weakly revoke P's membership from QC, AP and FPS.

Strong revocation requires revocation of both explicit and implicit membership. Strong revocation of a permission's membership in role x requires that the permission

be removed not only from explicit membership in x, but also from explicit (implicit) membership in all roles junior to x. Strong revocation therefore has a cascading effect downwards in the role hierarchy.

In the scheme, for example, P is an explicit member of role M1, QC, AU, AP and FPS. If Alice, with the activated administrative role NSSO, strongly revokes P's membership from QC, then P is removed not only from explicit membership in QC, but also from explicit (and implicit) membership in all roles junior to QC. Actually, after the strong revocation from QC, P has been removed from FPS, AP as well as QC. However, P still has a membership of AU and M1 since they are not junior roles to QC based on the role hierarchy of Figure 6.4. This brings about the same result as weak revocation from QC, AP and FPS by NSSO. Note that all implied revocations downward in the role hierarchy should be within the revocation range of the administrative roles that are active in a session. For instance, if Alice activates APSO and tries to strongly revoke P's membership from QC, she is not allowed to proceed because FPS is junior to QC but it is out of the APSO's *Can-revokep* range in Table 6.5.

6.5 Related work

Comparing with previous designed off-line payment schemes, the new payment scheme provides a flexible level of anonymity for consumers. This section continues to discuss the related work on user-role assignments and permission-role assignments. There are several other related works on these two assignments such as role-based access control models [1], role activation hierarchies [89].

A role-based separation of duty language (RSL 99) has been recently proposed [1]. It has given a formal syntax and semantics for RSL99 and has demonstrated its

soundness and completeness by using functions on conflicting permission sets. The proposal is different from ours in two aspects. First, It does not consider the case of the management for conflicting roles and permissions. Therefore, there is no support to deal administrative roles with permissions in the proposal. By contrast, our work provide a rich variety of options that can deal the document of administrative roles with roles and permissions. Second, the algorithm RSL99 does not provide access control models. It only gives separation of duty (SOD) policies. By contrast, we present a number of specialized authorization methods for access control which allow administrators to authorize a permission and user to role or revoke a permission and user from roles.

A separate role activation hierarchy which extends the permission-usage hierarchy has been proposed in [89]. The authors indicated two things. The first is to describe RBAC with respect to read-write access, and its relationship to traditional lattice-based access control (LBAC). The second is that roles are required to have dynamic separation of duty. RBAC with dynamic separation of duties is respected to write roles. However, our work substantially differs from that proposal. The main difference is that the paper [89] focuses on separated role activation hierarchy and we focus on an application of RBAC with a payment scheme. Furthermore, there is no e-commerce application test for role activation hierarchies in [89]. By contrast, we analyze the dynamic separation of duty (DSD) of roles in the payment scheme and use DSD to reduce conflicts between various roles and between various permissions in RBAC management.

6.6 Conclusions

The basic structure of RBAC has reviewed in this chapter. The user-role assignments and permission-role assignments and how to use RBAC with an electronic payment system are introduced.

Firstly, user-role assignments for the scalable anonymity payment scheme with RBAC are presented. It provides a way for using RBAC to manage electronic payment schemes. The duty separation constraints of the four roles in the scheme are analyzed. These constraints can be used to prevent unauthorized use of messages in the scheme. Based on the duty separation constraints, we have discussed how to grant a role to a user associated with a *Can-assign* relation. Because of role hierarchies, a user may still have a role which has been revoked by an administrative role. We have demonstrated this case in detail with weak revocation and strong revocation for the scheme.

Furthermore, permission-role assignments for electronic payment are analyzed. To address the problems that arise in permission-role assignment, how to grant a permission to a role associated with a *Can-revokep* relation are discussed. A role may still have a permission which has been revoked by an administrative role. The weak permission revocation and strong permission revocation for the scheme are also detailed disclosed.

Chapter 7

Formal Authorization Allocation Approaches for URA Based on Relational Algebra Operations

In this chapter, we develop formal authorization algorithms for role-based access control (RBAC). The formal approaches are based on relational structure, and relational algebra and operations. The process of user-role assignment (URA) is an important issue in RBAC because users may modify the authorization level or imply high-level confidential information to be derived while users change positions and request different roles. There are two types of problems which may arise in user-role assignment. One is related to authorization granting process. When a role is granted to a user, this role may be conflict with other roles of the user or together with this role; the user may have or derive a high level of authority. Another is related to authorization revocation. When a role is revoked from a user, the user may still have the role since role hierarchies.

To solve these problems, this chapter presents authorization granting algorithms, and weak revocation and strong revocation algorithms that are based on relational algebra for user-role assignments. The algorithms can be used to check conflicts and therefore to help allocate the roles and permissions without compromising the

security in RBAC. We extend our results to RBAC with mobility of user-role relationship. A user-role relationship is mobile if the user can be further assigned roles. Finally, comparisons with other related work are discussed.

The information in this chapter is based on a published paper [114].

7.1 Introduction

Role-based access control and its applications have been introduced in the last chapter. Many organizations, when they manage system applications, prefer to centrally control and maintain access rights, not so much at the system administrator's personal discretion but more in accordance with the organization's protection guidelines [32]. RBAC is being considered as part of the emerging SQL3 standard for database management systems, based on their implementation in Oracle 7 [90]. However, there is a consistency problem when using RBAC management. For instance, if there are hundreds of roles and thousands of users in a system, it is very difficult to maintain consistency because it may change the authorization level, or imply high-level confidential information to be derived when more than one role is requested and granted.

We develop formal approaches to check the conflicts and therefore help allocate the roles without compromising the security. The formal approaches are based on relational algebra, which has a set of complete and sound axioms. To the best of my knowledge, this is the first kind of work in this area to propose the formal approaches for role allocation and conflict detection.

The chapter is organized as follows. In the next section, the problems related to role assignment and revocation are identified. Relational algebra-based autho-

rization granting and revocation algorithms are developed in section 3 while their extensions are presented in section 4. The applications of the formal authorization approaches to a scheme are analyzed in section 5. Comparisons with related work are discussed in section 6 and the conclusions are in section 7.

7.2 Problem Definitions

RBAC model supports the specification of several aspects such as user-role assignment and permission-role assignment etc. A comprehensive administrative model for RBAC must account for all issues mentioned before. However, user-role assignments and permission-role assignments are particularly critical administrative activities. For user-role assignments, this is because assigning people to tasks is a normal managerial function and assigning users to roles is a natural part of assigning users to tasks. On the other hand for permission-role assignments, because conflict defined in terms of roles may allow conflicting permissions to be assigned to the same role but conflicts defined in terms of permissions eliminates this possibility. Therefore this chapter focuses on user-role assignments and permission-role assignments will be discussed in the next chapter.

Let D be a database with a set of relations REL, a set of attributes A. REL includes *ROLES, USERS, Can-assign, Can-revoke, SEN-JUN*, and *Role-User* etc. A includes attributes such as RoleName, UserID, UserName etc from the relations. R_1 is a set of roles $R_1 = \{r_1, r_2, ..., r_n\}$; U is a set of users $U = \{u_1, u_2, ..., u_l\}$. Roles are in two categories, one is administrative roles (admin.role), the other is regular roles (role) that need to be assigned to or revoked from users by administrative roles. The roles (permissions) assigned to a user (role) by administrators may be in conflicts. For example, in user-role assignments, SSD and DSD relationships may not

be sympathetic with the roles associated with the user; on the other hand, because of role hierarchies, the user (role) may still have the permissions of a role (permission) which has been revoked. In the latter case, the user is able to access objects in the permissions and has operations on the objects. Another example is in a bank, the permission for approving loan and that of funding loan are conflicting. These two permissions cannot be assigned to a role. The problems arising in processes of assigning and revocation are evident.

Conflicting problems in user-role assignments:

Authorization granting problem – How to check whether a role is in conflict with the roles of a user?

Authorization revocation problem – How to find whether permissions of a role have been revoked from a user or not?

For example, Figure 6.2 shows a system administrative role (ShopSO) in a shop to manage regular roles such as SELLER, AUDITOR and MANAGER. Role MANAGER inherits SELLER and AUDITOR. SELLER has a DSD relationship with AUDITOR. The administrative role ShopSO can assign a user to be AUDITOR or SELLER but not both simultaneously, otherwise it compromises the security of the shop system. It is easy to find conflicts between roles when assigning roles to a user in a small database system but it is hard to find them when there are thousands of roles in a system. Our aim is to provide relational algebra algorithms to solve the problems and then automatically check conflicts when assigning and revoking.

Some relations in set REL are detailed below.

ROLES - This relation has $(n+1)$ attributes when there are n roles.

The first attribute, RoleName is the primary key for the relation, and represents the name of a role. From the second attribute to $(n+1)th$ attribute, it describes the state of conflicts with the RoleName in the relation and its domain is {-1, 0}, where '-1' means conflicting with the RoleName and '0' means not.

The ROLES relation in Figure 6.2 is in Table 7.1. The attribute SELLERC shows whether the role SELLER is conflicting with the RoleName in the relation or not. For instance, in the third tuple, a user with role SELLER has conflicts with the role AUDITOR.

RoleName	MANAC	AUDC	SELLERC	SHOPC
MANAGER	0	0	0	0
AUDITOR	-1	0	-1	0
SELLER	-1	-1	0	0
SHOP	-1	-1	-1	0

Table 7.1: The relation ROLES in Figure shops

USERS - This relation has two attributes {UserID, UserName }, UserID is the identity of a user and UserName, which domain is the set of a list of users in the system. UserID and UserName are recorded for each user. UserName is the primary key for the table. For example, there are two users David and Tony in the system of Figure 6.2. The USERS relation is in Table 7.2.

UserID	UserName
0001	David
0002	Tony

Table 7.2: The relation USERS in Figure shops

Roles are managed by administrative roles. Senior roles are shown at the top of the hierarchies. Senior roles inherit permissions from junior roles. Let $x > y$ denote

x is senior to y with obvious extension to $x \geq y$.

SEN-JUN - This is a relation of roles in a system. Senior is the senior of the two roles. Tables 7.3 expresses the SEN-JUN relationship in Figure 6.2.

Senior	Junior
MANAGER	AUDITOR
MANAGER	SELLER
MANAGER	SHOP
SELLER	SHOP
AUDITOR	SHOP

Table 7.3: SEN-JUN table in Figure shops

ROLE-USER - defines a relationship between USERS and ROLES {RoleName, UserName}.

RoleName is a foreign key RoleName in the ROLES. It explains a role is assigned to a user.

UserName is a foreign key UserName from the USERS.

Suppose MANAGER is assigned to user Tony and SELLER to user David, Table 7.4 expresses the ROLE-USER relationship.

RoleName	UserName
MANAGER	Tony
SELLER	David

Table 7.4: ROLE-USER table

It is easy to know a role set associated with a user from ROLE-USER. Therefore, the authorization granting problem can be changed to whether a role is conflicting

with a set of roles R or not. Based on these relations, we describe how to solve the authorization granting problem and revocation problem in the next section.

7.3 Authorization granting and revocation algorithms

We develop granting and revocation algorithms for user-role assignments based on relational algebra in this section. Supposing an administrator role ADrole wants to assign a role r_j to a user with a set of roles R. R^* is an extension of R, $R^* = \{x|x \in R\} \cup \{x|\exists x' \in R, x' > x\}$. A authorization granting algorithm is designed for justifying whether the role r_j is in conflict with R or not. The following are main ideas of the algorithm. The first is to decide whether ADrole can assign the role r_j to the user or not. Based on the relation of *Can-assign*, we can get the prerequisite conditions. The ADrole can assign r_j to the user only if a role in role set R^* also belongs to the prerequisite conditions. The second is to decide whether the role r_j is conflicting with roles in R or not. From the relation of *ROLES*, we are able to obtain the number of the attribute r_jC that describes conflicting states of r_j with other roles. The details are below.

Authorization granting algorithm Grant(ADrole, R, r_j)

Input: ADrole, a set of roles R and a role r_j.

Output: true if ADrole can assign role r_j to R with no conflicts; false otherwise.

Begin:

Num: $= 0$

Step 1. /* *Whether the ADrole can assign the role r_j to R or not* */

Suppose $S_1 = R^* \cap S_2$ where

$$S_2 = \pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can - assign))$$

if $S_1 \neq \phi$,

then there exists role $r \in S_1$, such that

$r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign))$

go to step 2

/* r_j *is in the range to be assigned by ADrole in Can-assign* */

else

return false and stop.

/**the admin.role has no right to assign the role* r_j *to* R^* */

Step 2. /* *whether the role* r_j *is conflicting with roles in* R *or not*/

for each role r_i in R, do

$Num_i = \pi_{r_jC}(\sigma_{RoleName=r_i}(ROLES))$

/* r_jC *is an attribute that describes conflicting states of* r_j *with other roles* */

if $Num_i = -1$

r_j is a conflicting role with R, *return* false;

if for all $r_i \in R$, $Num_i = 0$ *return* true.

/* r_j *is not a conflicting role with* R */ ◇

Theorem 7.1 *The authorization granting algorithm can prevent conflicts when assigning a role to a user.*

Proof Assuming an administrator role ADrole wants to assign a role r_j to a user which associates with a role set R. While step 1 in the algorithm has checked whether the ADrole can assign the role r_j to a user or not, the second step has decided whether the role r_j is conflicting with roles in R or not. Indeed, r_j can be assigned to the user if for all $r_i \in R$, $Num_i = 0$. Otherwise r_j is a conflicting role with R. ◇

The authorization granting problem is solved by the authorization granting algorithm. Computing S_1 in the first step takes time proportional to n if n is presented as the number of roles. Computing $\sigma_{admin.role=ADrole}(Can - assign)$ and $\pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can - assign))$ needs time $O(n)$. Thus, the step 1 takes time $O(n^2)$. In the second step, the computations of $\sigma_{RoleName=r_i}(ROLES)$ and $\pi_{r_jC}(\sigma_{RoleName=r_i}(ROLES))$ spent $O(n)$ and $O(1)$ respectively. It needs time $O(n^2)$ to get Num_i for all $r_i \in R$. Therefore the total time spent in the authorization granting algorithm is proportional to n^2.

Corollary 7.1 *The authorization granting algorithm has time complexity $O(n^2)$, the n is the number of acting roles.*

Now we consider revocation of user-role membership. Due to role hierarchy, a role x' has all permissions of role x when $x' > x$. A user with two roles $\{x', x\}$ still has the permissions of x if only to revoke x from the user. The explicit member of a role x is a set of user $\{U|(U, x) \in UA\}$ and the implicit member of role x is a set of user $\{U|\exists x' > x, (U, x') \in UA\}$. To solve the authorization revocation problem,

we need to revoke the explicit member of a role first if a user is an explicit member, then revoke the implicit member.

Following are two algorithms for revocation of a role r_j from a set of roles R by an administrative role ADrole, where R is a set of roles which are assigned to a user. The first one is a weak revocation algorithm and another one is a strong revocation algorithm. The weak revocation only revokes explicit membership from a user and does not revoke implicit membership but the strong revocation revokes both explicit and implicit members.

In the weak revocation algorithm, the first step is to check whether r_j is in R or not. There is no affect when r_j is not in R. The second step is to verify whether r_j belongs to the role range of the relation *Can-revoke* with the attribute admin.role of ADrole. The r_j is revoked if it drops in the role range. Otherwise, no affect with the weak revocation processing.

Weak revocation Algorithm Weak_revoke(ADrole, R, r_j)
Input: ADrole, a set of roles R and a role r_j.
Output: true if ADrole can weakly revoke role r_j from R; false otherwise.
Begin:

if $r_j \notin R$,

return false;

/* *there is no effect with the operation of the weak revocation since the user is not an explicit member of* r_j*/

else /* *The user with the role set* R *is an explicit member of* r_j */

1. *if* $r_j \in \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revoke))$,

return true;

/**the user's explicit membership in r_j is revoked* */

2. *if* $r_j \notin \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revoke))$,

return, false.

/* *ADrole has no right to revoke the role* r_j *from the user* */ ◇

We have the following result with the weak revocation algorithm.

Theorem 7.2 *A role* r_j *is revoked by the weak revocation algorithm* $Weak\_revoke(ADrole, R, r_j)$ *if the user is an explicit member of role* r_j *and the ADrole has the right to revoke* r_j *from the Can-revoke relation.* ◇

It takes time $O(n)$ to check if $r_j \notin R$ when there are n roles in a system. The computations of $\sigma_{admin.role=ADrole}(Can - revoke)$ and $\pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revoke))$ take time $O(n)$. To check whether $r_j \in \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revoke))$ needs time $O(n)$. Thus, the time complexity of the weak revocation algorithm is $O(n)$.

Corollary 7.2 *Weak revocation algorithm has time complexity* $O(n)$ *when there are* n *roles in a system.*

A user still has permissions of a role which has been weakly revoked if a role associated with the user seniors the role revoked. To solve the authorization revocation problem, we need strong revocation which requires revocation of both explicit and implicit membership. Strong revocation of a user's membership in x requires

that the user be removed not only from explicit membership in x, but also from explicit and implicit membership in all roles senior to x.

Supposing ADrole likes to strong revoke role r_j from a user u with role set R. In strong revocation processing, there is no effect if u is neither an explicit member of r_j nor an implicit member. r_j is revoked if $r_j \in R$. From the relation of *SEN=JUN*, we can get a role set of all senior roles of r_j. When a role is in both the set and R^*, it has to be weak revoked by ADrole.

Strong revocation algorithm Strong_revoke(ADrole, R, r_j)

Input: ADrole, a set of roles R and a role r_j.

Output: true, if it can strong revoke role r_j from R; false otherwise.

Begin:

if $r_j \notin R^*$,

return false;

/* *there is no effect of the strong revocation since the user is not an explicit and implicit member of* r_j */

else,

1. if $r_j \in R$, do Weak_revoke(ADrole, R, r_j);

/*r_j *is weakly revoked from* R^*/;

2. Suppose

$Sen = R^* \cap \pi_{Senior}(\sigma_{Junior=r_j}(SEN - JUN))$,

for all $y \in Sen$, do Weak_revoke(ADrole, R, y);

/* *the user is weakly revoked from all such* $y \in Sen^*$/.

If all the weak revocations are successful,

return true;

otherwise, *return* false. ◇

It should be noted that Weak_revoke(ADrole, R, r_j) and Weak_revoke(ADrole, R, y) do not work if ADrole has no right to revoke r_j. We have the following consequence.

Theorem 7.3 *The explicit and implicit member of role r_j are revoked from the user by the strong revocation algorithm $Strong\_revoke(ADrole, R, r_j)$ if the ADrole has the right to revoke r_j from the Can-revoke relation.*

Corollary 7.3 *The authorization revocation problem is solved by the weak revocation algorithm and strong revocation algorithm.*

It needs $O(n)$ to check whether $r_j \notin R^*$ if there are n roles in a system. The computations of $\sigma_{Junior=r_j}(SEN-JUN)$ and $\pi_{Senior}(\sigma_{Junior=r_j}(SEN-JUN))$ takes time proportional to m where m is the number of tuples in the relation SEN-JUN and $m < n$. It takes time proportional to $n * m$ to compute Sen. Other operations $r_j \in R$, r_j weak revocation and $y \in Sen$ takes time $O(n)$. Therefore the total time spent with the Strong revocation algorithm is $O(n^2)$.

Corollary 7.4 *The strong revocation algorithm has time complexity $O(n^2)$ when there are n roles in a system.*

7.4 Algorithms for URA with the mobility of user-role relationship

The mobility of user-role relationship is a new feature relative to their counterparts in user-role assignments. When an administrative role assigns a role to a user with a mobile relationship, this allows the user to use the permissions of the role and to be further assigned other roles by administrators. If the relationship is immobile, the user is authorized to use the permissions, but cannot be granted other roles by junior administrators. Similar to the conflicting problems in the previous sections, there are two types of problems that may arise in user-role assignment with the mobility of user-role relationship. One is related to authorization granting process while the other is related to authorization revocation. The authorization algorithms developed in the last section do not work well for user-role assignment with the mobility of user-role relationship. This is because granting and revocation models are different in user-role assignment with the mobility and in those without the mobility.

In this section, we discuss granting and revocation models related to mobile and immobile memberships between users and roles, then provide an enhanced authorization granting algorithm, and weak revocation and strong revocation algorithms that are based on relational algebra and operations.

7.4.1 Introduction of mobility

Administrative roles in Figure 7.1 are used to explain the mobility of user-role relationship. The Figure shows hierarchies of administrative roles. The senior administrative role SAR inherits administrative role AR with all permissions of AR. Similarly, AR inherits junior administrative role JAR. The relationship between user and role is mobile relationship means that the user may further accept roles, and

we say the user is a mobile member of the role or the role is a mobile member of the user. With an immobile relationship the user cannot be granted other roles, and we say the user (role) is an immobile member of the role (user). Supposing the administrative role AR assigns a role to a user as mobile member, it means the user can use permissions of the role and other administrative role like SAR and JAR may assign other roles to the user. If the user is assigned a role as an immobile member by AR, JAR cannot update and assign other roles to the user except AR or SAR changes the immobile membership.

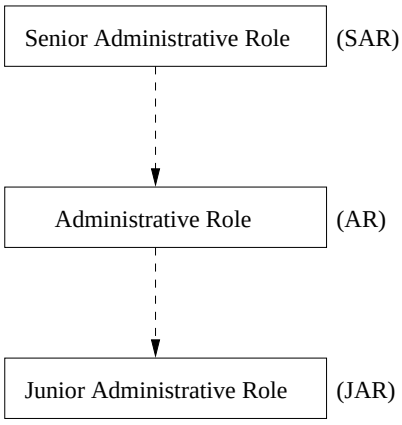

Figure 7.1: Hierarchies of administrative roles

There are two issues need to be addressed in user-role assignment. The first is to control the roles that an administrative role has authority over. Figure 6.2, for example, shows the administrative role ShopSO which co-exists with the regular roles SHOP, SELLER etc. The administrative role ShopSO controls roles in a shop, that are SHOP, SELLER, MANGER and AUDITOR. The second is to control which users are eligible for membership in these roles.

There are two kinds of membership between user and role, namely mobility and immobility. When a user is assigned a role with mobile membership by an

administrative role, the user is authorized to use the permissions of a role and he/she is eligible to further accept other roles. A user with an immobile member of a role only gets the permissions of the role but cannot be assigned other roles by the member of junior administrators.

This distinction between mobile and immobile memberships can be very important in practice [73, 94]. For example, in a University, a guest can be granted immobile membership in a role as an observer by an administrative role but he/she cannot be granted with staff roles by junior administrators. Therefore, the guest is able to visit the University without permissions to access original data as staff member. This can improve security of the system in the University. An other example is in a shop model mentioned above, a person under training can assign role SHOP as an immobile member by administrative role ShopSO and thus participate in the shop while preventing junior administrators from assigning other roles (for example SELLER and AUDITOR) to the person. After completion of training the membership of the person with immobile member SHOP can be upgraded to be mobile by ShopSO or its senior administrative roles. One more example is still in the shop, a consultant who might be granted the SHOP role as an immobile member by ShopSO. The consultant can participate in the shop and use the general resources of the shop. At the same time the immobility of the consultant prevents junior administrators from assigning SELLER or AUDITOR roles to the consultant.

These examples demonstrate that there are a lot of situations with mobile and immobile relationships between users and roles in practice. However, similar to the problems in [114], there are two problems may arise in user-role assignment with mobility of user-role relationship. They are:

Authorization granting problem – How to check whether a role is in conflict

with the roles of a user when granting the role to the user as mobile or immobile member?

Authorization revocation problem – How to find whether mobile or immobile membership of a role have been revoked from a user or not?

Our aim in this section is to provide relational algebra algorithms to solve these problems and then automatically check conflicts when assigning and revoking.

7.4.2 Authorization granting and revocation algorithms

We develop granting and revocation algorithms based on relational algebra in this subsection. A user's membership in a role can be mobile or immobile. Mobile membership of user u in role x means that u can use permissions of the role x and u can also be put into other roles. Immobile membership of user u in role x means that u can use permissions of role x but cannot be put into other roles by appropriate junior administrators. Each role x is separated to two sub-roles Mx and IMx. Membership in Mx is mobile while membership in IMx is immobile. Assignment of Mx to a user specifies that the user is a mobile member of x. Similarly, assignment of IMx to a user specifies that the user is an immobile member of x.

Based on the mobile and immobile membership with the notion of explicit and implicit membership, there are four kinds of user-role membership for a given role x [94].

1: *Explicit Mobile Member EMx*

$EMx = \{u, (u, Mx) \in UA\}$

2: *Explicit Immobile Member EIMx*

$EIMx = \{u, (u, IMx) \in UA\}$

3: *Implicit Mobile Member ImMx*

$ImMx = \{u, \exists x' > x, (u, Mx') \in UA\}$

4: *Implicit Immobile Member ImIMx*

$ImIMx = \{u, \exists x' > x, (u, IMx') \in UA\}$

A user may have all four kinds of membership in a role at the same time. However, we limit strict precedence amongst these four kinds of membership as follows:

$$EMx > EIMx > ImMx > ImIMx$$

Therefore only one of the membership is actually in effect at any time even though a user has multiple kinds of membership in a role.

The meaning of a prerequisite condition in paper [114] is simple since the notion of role membership does not include mobile and immobile, explicit and implicit memberships. Now we need to interpret a prerequisite condition with these memberships.

A **prerequisite conditionM** is evaluated for a user u by interpreting role x to be true if

$$u \in EMx \vee (u \in ImMx \wedge u \notin EIMx)$$

and $\bar{x}$ to be true if

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

In other words x denotes mobile membership (explicit or implicit) and $\bar{x}$ denotes absence of any kind of membership.

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign a role to a user. The following relations provide what roles an administrator can assign mobile members or immobile members with prerequisite conditions.

Can-assign-M is a relation of $\subseteq AR \times CR \times 2^R$, which is used for user-role assignments as mobile members. where AR is a set of administrative roles. On the other hand, user-role assignments as immobile members are authorized by the relation **Can-assign-IM** $\subseteq AR \times CR \times 2^R$ $\diamond$

User-role assignment (URA) is authorized by *Can-assign-M* and *Can-assign-IM* relations. Table 7.5 and Table 7.6 shows the *Can-assign-M* and *Can-assign-IM* relations with the prerequisite conditions in the shop example.

Admin.role	Prereq.Condition	Role Range
ShopSO	SHOP	[SHOP, SHOP]
ShopSO	SHOP $\wedge\overline{SELLER}$	[AUDITOR, AUDITOR]
ShopSO	SHOP $\wedge\overline{AUDITOR}$	[SELLER, SELLER]
ShopSO	SELLER $\wedge$ AUDITOR	[MANAGER, MANAGER]

Table 7.5: Can-assign-M in the example

Admin.role	Prereq.Condition	Role Range
ShopSO	SHOP	[SHOP, SHOP]
ShopSO	SHOP $\wedge\overline{SELLER}$	[AUDITOR, AUDITOR]
ShopSO	SHOP $\wedge\overline{AUDITOR}$	[SELLER, SELLER]

Table 7.6: Can-assign-IM in the example

The meaning of *Can-assign-M* $(ShopSO, SHOP, \{SELLER, AUDITOR\})$ is that a member of the administrative role ShopSO can assign a user whose current membership satisfies the *prerequisite conditionM* SHOP to be a mobile member of roles SELLER and AUDITOR. Whereas the meaning of *Can-assign-IM*

$(ShopSO, SHOP, \{SELLER, AUDITOR\})$ is that a member of the administrative role ShopSO can assign a user whose current membership satisfies the *prerequisite conditionM* SHOP to be a immobile member of roles SELLER and AUDITOR.

Supposing an administrator role ADrole wants to assign a role r_j to user u with a set of roles R which has mobile or immobile memberships with u. The role r_j has mobile or immobile membership with u if ADrole can assign without conflicts. We discuss both of mobile and immobile members in the following algorithm. The structure of the algorithm is shown in Figure 7.2. R^* is an extension of R, $R^* = \{x | x \in R\} \cup \{x | \exists x' \in R, x' > x\}$.

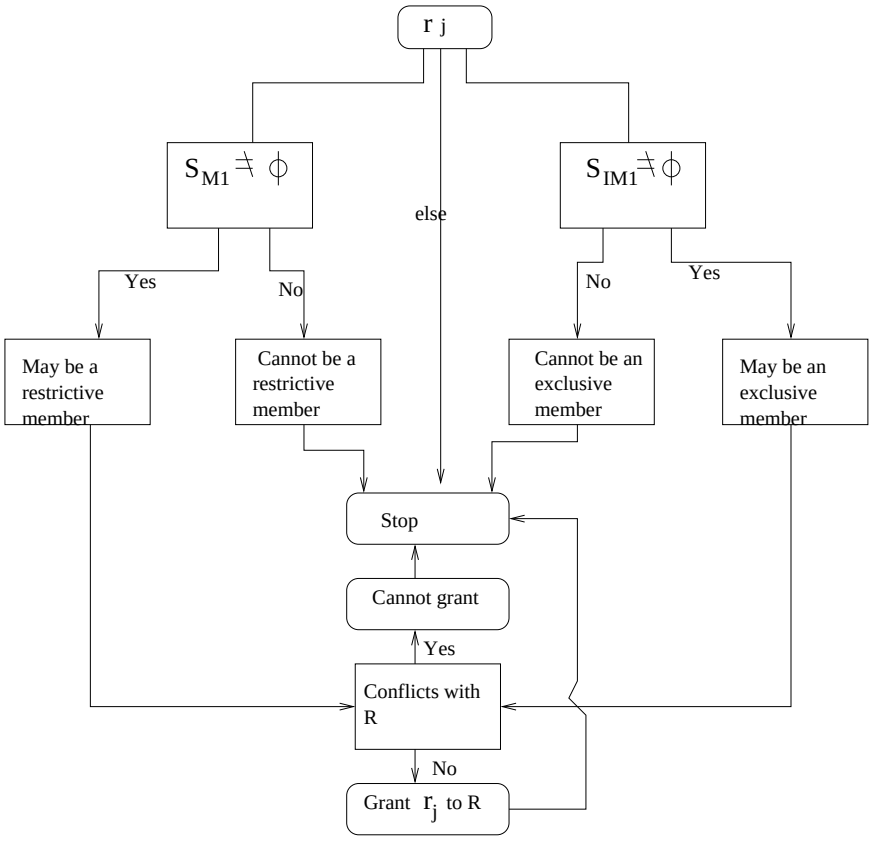

Figure 7.2: Grant algorithm structure

Authorization granting algorithm Grant(ADrole, R, r_j)

Input: ADrole, a set of roles R and a role r_j.

Output: true if ADrole can assign role r_j to R with no conflicts; false otherwise.

Begin:

Num: $= 0$

Step 1. /* *Whether the ADrole can assign the role r_j to R as mobile or immobile member or not* */

Suppose $S_{M1} = R^* \cap S_{M2}$ and $S_{IM1} = R^* \cap S_{IM2}$ where

$$S_{M2} = \pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can - assign - M))$$

$$S_{IM2} = \pi_{Prereq.Condition}(\sigma_{admin.role=ADrole}(Can - assign - IM))$$

if r_j will be a mobile member of the user and $S_{M1} \neq \phi$,

then there exists role $r \in S_{M1}$, such that

$r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - M))$

go to step 2

/* *r_j is in the range to be assigned as a mobile member by ADrole in Can-assign-M* */

if r_j will be an immobile member of u and $S_{IM1} \neq \phi$,

then there exists role $r \in S_{IM1}$, such that

$r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - IM))$

go to step 2

/* *r_j is in the range to be assigned as an immobile member by ADrole in Can-assign-IM* */

else

return false and stop.

*/*the admin.role has no right to assign the role r_j as a mobile or immobile member to R */*

Step 2. */* whether the role r_j is conflicting with roles in R or not*/*

for each role r_i in R, do

$Num_i = \pi_{r_jC}(\sigma_{RoleName=r_i}(ROLES))$

/ r_jC is an attribute that describes conflicting states of r_j with other roles */*

if $Num_i = -1$

r_j is a conflicting role with R, *return* false;

if for all $r_i \in R$, $Num_i = 0$

return true. */* r_j is not a conflicting role with R */* ◇

Remark 1: Normally we need to check the immobile roles in the role set R in the step 2. This is because the user may not be assigned other roles if it has immobile roles. However, this optional problem can be solved by the *prerequisite conditionM*. Indeed, when an administrative role assigned a role as mobile or immobile member to a user, the administrative role and its senior administrators can update the membership while a junior administrator may not. Therefore it only needs to check conflicts between the role r_j and the role set R.

This algorithm provides a way for whether a role can be assigned as mobile or

immobile member to a user. For mobile member, S_{M1} cannot be empty, and for immobile member, S_{IM1} cannot be empty.

Theorem 7.4 *The authorization granting algorithm provides a solution for the authorization granting problem. It can prevent conflicts when assigning a role to a user with mobile and immobile memberships.*

Proof Assuming an administrator role ADrole wants to assign a role r_j as a mobile member to a user which associates with a role set R. While step 1 in the algorithm has checked whether the ADrole can assign the role r_j as a mobile member to the user or not, the second step has decided whether the role r_j is conflicting with roles in R or not. Indeed, r_j can be assigned to the user if for all $r_i \in R$, $Num_i = 0$. Otherwise r_j is a conflicting role with R. ◇

The authorization granting problem is solved by the authorization granting algorithm. Computing S_1 in the first step takes time proportional to n if n is presented as the number of roles. Computing of $\sigma_{admin.role=ADrole}(Can - assign - M)$ and $\sigma_{admin.role=ADrole}(Can - assign - IM)$ take time $O(1)$. The time for getting S_{M2} and S_{IM2} is $O(n)$. It uses $O(n^2)$ to obtain S_{M1} and S_{IM1}. Thus, the step 1 takes time $O(n^2)$. In the second step, the computations of $\sigma_{RoleName=r_i}(ROLES)$ and $\pi_{r_jC}(\sigma_{RoleName=r_i}(ROLES))$ spent $O(n)$ and $O(1)$ respectively. It needs time $O(n^2)$ to get Num_i for all $r_i \in R$. Therefore the total time spent in the authorization granting algorithm is proportional to n^2.

Corollary 7.5 *The authorization granting algorithm has time complexity $O(n^2)$ for the case of n roles in a system.*

Now we consider revocation of user-role membership. Similar to *Can-assign-M*

Admin.role	Prereq.ConditionR	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 7.7: Can-revoke-M

and *Can-assign-IM* relations in granting a role to a user, there is a *Can-revoke-M* and *Can-revoke-IM* relations between administrative roles and regular roles.

In our previous paper [114] the relation *Can-assign* involves the prerequisite conditions but *Can-revoke* does not. The prerequisite conditions in revocation is also important. For example, if a user Bob is a member of SHOP then ShopSO may assign Bob to any role (mobile or immobile) of the shop, namely SHOP, SELLER and AUDITOR. These assignments are governed by the relations *Can-assign-M* and *Can-assign-IM*. Suppose ShopSO does not like Bob to be a member of any role outside the shop. If Bob is assigned a role that falls outside the shop then ShopSO needs revoke him from that role. Prerequisite conditions in *Can-revoke* are one means to provide this function.

Relations **Can-revoke-M** $\subseteq AR \times CR \times 2^R$ and **Can-revoke-IM** $\subseteq AR \times CR \times 2^R$ show which role range of mobile membership and immobile membership administrative roles can revoke respectively, where AR is a set of administrative roles. $\diamond$

The meaning of *Can-revoke-M* $(ShopSO, [SHOP, MANAGER))$ in Table 7.7 is that a member of the administrative role Shop can revoke mobile membership of a user from any role in [SHOP, MANAGER). Similarly, for *Can-revoke-IM* with respect to immobile membership.

The evaluation of a prerequisite condition for the revoke model is different from

Admin.role	Prereq.ConditionR	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 7.8: Can-revoke-IM

the grant model. In the revoke model a **prerequisite conditionR** is evaluated for a user u by interpreting role x to be true if

$$u \in EMx \vee u \in EIMx \vee u \in ImMx \vee u \in ImIMx$$

and $\bar{x}$ to be true if

$$u \notin EMx \wedge u \notin EIMx \wedge u \notin ImMx \wedge u \notin ImIMx$$

Due to role hierarchy, a role x' has all permissions of role x when x' inherits x. A user with two roles $\{x', x\}$ still has the permissions of x if only to revoke x from the user. To solve the authorization revocation problem, we need to revoke the explicit member of a role first if a user is an explicit member, then revoke the implicit member.

Following are two algorithms for revocation of a role r_j as mobile and immobile members from a user u by an administrative role ADrole, where R is a set of roles which are assigned to the user u. The first one is a weak revocation algorithm and another one is a strong revocation algorithm. The weak revocation only revokes explicit mobile or immobile memberships from u and does not revoke implicit mobile and immobile memberships but the strong revocation revokes both explicit and implicit mobile and immobile members. The structure of weak revocation algorithm is shown in Figure 7.3.

Weak revocation Algorithm Weak_revoke(ADrole, R, r_j)

Input: ADrole, a set of roles R and a role r_j.

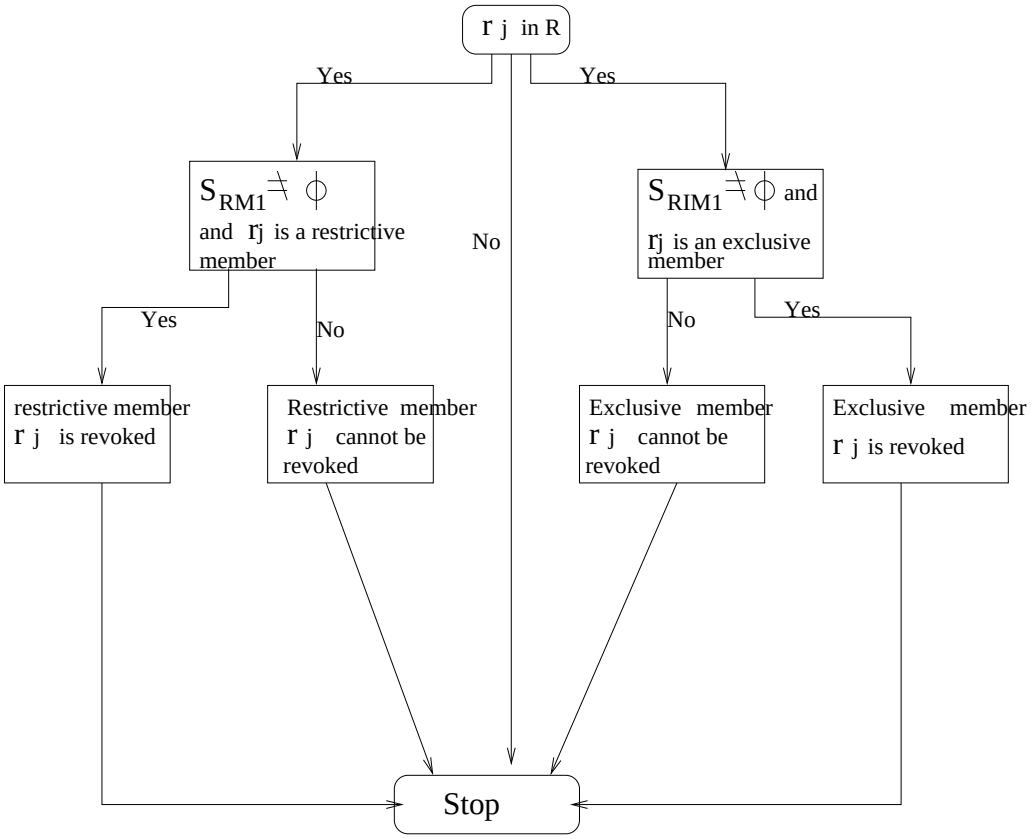

Figure 7.3: Structure of weak revocation algorithm

Output: true if ADrole can weakly revoke role Mr_j from R; false otherwise.

Begin:

Step 1: *if* $r_j \notin R$,

return false and stop;

/* *there is no effect with the operation of the weak revocation since the user is not an explicit member of* Mr_j*/

Step 2. /* *Whether the ADrole can revoke the role* r_j *from* R *or not* */

Suppose $S_{RM1} = R^* \cap S_{RM2}$ and $S_{RIM1} = R^* \cap S_{RIM2}$, where

$$S_{RM2} = \pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can - revoke - M))$$

$$S_{RIM2} = \pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can - revoke - IM))$$

if $S_{RM1} \neq \phi$ and $u \in EMr_j$,

then there exists role $r \in S_{RM1}$, such that

$r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - revoke - M))$

Return true;

/* r_j *is in the range to be revoked by ADrole in Can-revoke-M* */

if $S_{RIM1} \neq \phi$ and $u \in EIMr_j$,

then there exists role $r \in S_{RIM1}$, such that

$r_j \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - revoke - IM))$

Return true

/* r_j *is in the range to be revoked by ADrole in Can-revoke-IM and the immobile membership is revoked* */

else

return false and stop.

/**the admin.role has no right to revoke the role* r_j *from* u */ ◇

The weak revocation algorithm can be used to check whether an administrator can weakly revoke mobile and immobile memberships from users or not. We have the following result with the weak revocation algorithm.

Theorem 7.5 *A roles r_j as mobile or immobile member is revoked by the weak revocation algorithm $Weak\_revoke(ADrole, R, r_j)$ if the user is an explicit member of role r_j and the ADrole has the right to revoke r_j from the Can-revoke-M and Can-revoke-IM relations.*

It takes time $O(n)$ to check if $r_j \notin R$ when there are n roles in a system. The computations of $\sigma_{admin.role=ADrole}(Can-revoke-M)$, $\sigma_{admin.role=ADrole}(Can-revoke-IM)$, $\pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can-revoke-M))$, $\pi_{Prereq.ConditionR}(\sigma_{admin.role=ADrole}(Can-revoke-IM))$ take time $O(n)$. The time for computing S_{RM1} and S_{RIM1} needs $O(n^2)$. Thus, the time complexity of the weak revocation algorithm is $O(n^2)$.

Corollary 7.6 *Weak revocation algorithm has time complexity $O(n^2)$ when there are n roles in a system.*

A user still has permissions of a role which has been weakly revoked if a role associated with the user is senior the role revoked. To solve the authorization revocation problem, we need strong revocation which requires revocation of both explicit mobile and immobile memberships and implicit mobile and immobile memberships. Strong revocation of a user's mobile and immobile memberships in x requires that the user be removed not only from explicit mobile and immobile memberships in x, but also from explicit and implicit mobile and immobile memberships in all roles senior to x. Strong revocation therefore has a cascading effect up-wards in the role hierarchy.

Strong revocation algorithm Strong_revoke(ADrole, R, r_j)

Input: ADrole, a set of roles R and a role r_j.

Output: true, if it can strong revoke role r_j from R; false otherwise.

Begin:

if $r_j \notin R^*$,

return false;

/ there is no effect of the strong revocation since the user is not an explicit and implicit member of* r_j **/*

else,

1. if $r_j \in R$ is a mobile member of the user, do

Weak_revoke(ADrole, R, r_j) as $u \in EMr_j$

/ r_j as a mobile member is weakly revoked */*;

2. if $r_j \in R$ is a immobile member of the user, do

Weak_revoke(ADrole, R, r_j) as $u \in EIMr_j$;

/r_j is weakly revoked from R^*/*;

3. Suppose

$Sen = R^* \cap \pi_{Senior}(\sigma_{Junior=r_j}(SEN - JUN))$,

for all $y \in Sen$ with mobile membership with the user, do Weak_revoke(ADrole, R, y) as $u \in EMy$;

for all $y \in Sen$ with immobile membership with the user, do Weak_revoke(ADrole, R, y) as $u \in EIMy$;

/ the user is weakly revoked from all such user's members* $y \in Sen$ **/*.

If all the weak revocations are successful,

return true;

otherwise, *return* false. ◇

It should be noted that Weak_revoke(ADrole, R, r_j) and Weak_revoke(ADrole, R, y) do not work if ADrole has no right to revoke r_j. We have the following consequence.

Theorem 7.6 *The explicit mobile and immobile and implicit mobile and immobile members of role r_j are revoked from the user by the strong revocation algorithm $Strong\_revoke(ADrole, R, r_j)$ if the ADrole has the right to revoke r_j from the Can-revoke-M and Can-revoke-IM relations.*

Corollary 7.7 *The authorization revocation problem is solved by the weak revocation algorithm and strong revocation algorithm.*

It needs $O(n)$ to check whether $r_j \in R^*, u \in EMr_j$ and $u \in EIMr_j$ if there are n roles in a system. The computations of $\sigma_{Junior=r_j}(SEN - JUN)$ and $\pi_{Senior}(\sigma_{Junior=r_j}(SEN - JUN))$ takes time proportional to m where m is the number of tuples in the relation SEN-JUN and $m < n$. It takes time proportional to $n*m$ to compute Sen. Other operations $r_j \in R$, r_j weak revocation and $y \in Sen$ takes time $O(n^2)$. Therefore the total time spent with the Strong revocation algorithm is $O(n^3)$.

Corollary 7.8 *The strong revocation algorithm has time complexity $O(n^3)$ when there are n roles in a system.*

7.5 Applications of the relational algebra algorithms

The new relational algebra algorithms are applied to the payment scheme in this section. We only consider the relationships of users and roles without user's mobility in the scheme, then analyze applications of the relational algebra algorithms.

7.5.1 An application of the authorization granting algorithm

A hierarchy of roles and a hierarchy of administrative roles are shown in Figure 6.4 and Figure 6.5 respectively. Table 6.2 shows the *Can-assign* relations with the prerequisite conditions in the scheme and Table 7.11 shows ROLES relationship (the attribute M1C, for example, shows whether the role M1 is conflicting with the RoleName in the relation or not).

Assume Bob is a user with role FPS. The administrative role NSSO wants to assign the role AP to Bob. Using the granting algorithm *Grant(NSSO, FPS, AP)*, the first step, $R^* = R = \{FPS\}$ and

$$\pi_{Prereq.Condition}(\sigma_{Admin.role=NSSO}(Can-assign)) = \{FPS\},$$

then

$$R^* \cap \pi_{Prereq.Condition}(\sigma_{NSSO}(Can-assign)) \neq \phi.$$

This means NSSO can assign role AP to Bob. The second step, based on Table 7.11,

$$Num = \pi_{APC}(\sigma_{RoleName=FPS}(ROLES)) = 0$$

It means no conflicts when assigning AP to Bob.

7.5.2 Application of the authorization revocation algorithm

Table 7.9 and Table 7.10 give the *Can-revoke* and a part of senior-junior relationship of the payment scheme.

Admin.role	Role Range
APSO	[AP, M1)
ShopSO	[Shop, M3)
NSSO	(FPS, DIR)
SSO	[FPS, DIR]

Table 7.9: Can-revoke of the payment scheme

Senior	Junior
FPS	E
OP	AP
QC	AP
M1	OP
M1	QC
M1	AP
Director	M1

Table 7.10: A part of SEN-JUN relation of the payment scheme

Suppose Bob is an explicit member of M1, QC, AU, AUDITOR, AP, FPS and E in the payment scheme. If Alice, with the activated administrative role APSO, weakly revokes Bob's membership from AP, the revocation is successful by the weak revocation algorithm Weak_revoke(APSO, R, AP),

$$R = \{M1, QC, AU, AUDITOR, AP, FPS, E\},$$

$$AP \in \pi_{RoleRange}(\sigma_{Admin.role=APSO}(Can - revoke)).$$

However, he continues to be a member of the senior roles to AP since both M1 and QC are senior roles to AP, therefore he can use the permission of AP. It is necessary

to note that Alice should have enough power to weakly revoke Bob's membership from his explicitly assigned roles. For instance, if Alice has activated APSO and then tries to weakly revoke Bob's membership from M1, she is not allowed to proceed because APSO does not have the authority of weak revocation from M1 according to the *Can-revoke* relation in Table 7.9.

Therefore, if Alice wants to revoke Bob's explicit membership as well as implicit membership from AP by weak revocation, she needs to activate SSO and weakly revoke Bob's membership from AP, QC and M1. This brings about the same result as strong revocation from AP by SSO. However, Alice does not need to revoke Bob's membership from FPS, AU, AUDITOR and E, because they are not senior roles to AP based on the role hierarchy of Figure 6.4.

For example, Bob is an explicit member of M1, QC, AU, AP, AUDITOR, FPS and E. If Alice with the activated administrative role SSO strongly revokes Bob's membership from AP, then he is removed not only from explicit membership in AP, also from explicit and implicit membership in all roles senior to AP. Using the strong revocation algorithm Strong_revoke(SSO, R, AP),

$$R = \{M1, QC, AU, AP, AUDITOR, FPS, E\}$$

$$R^* = \{M1, QC, OP, AU, AP, Bank, AUDITOR, FPS, E\}$$

Step 1, role AP is revoked from R since AP is an explicit member.

Step 2,

$$\begin{aligned} Sen &= R^* \cap \pi_{Senior}(\sigma_{Junior=AP}(SEN - JUN)) \\ &= \{M1, QC\}. \end{aligned}$$

This means Bob has been removed from M1, QC as well as AP after the strong revocation from AP. However, he still has a membership of FPS, AU, AUDITOR and E, since they are not senior roles to AP based on the role hierarchy of Figure

RoleName	EC	FPSC	APC	QCC	OPC	M1C	BaC	TEC	ACC	AUC	M2C	ShC	SAC	AU3C	M3C	DiC
E	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
FPS	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
AP	0	0	0	-1	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	-1
QC	0	0	0	0	-1	-1	-1	-1	-1	-1	-1	0	0	0	0	-1
OP	0	0	0	-1	0	-1	-1	-1	-1	-1	-1	0	0	0	0	-1
M1	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	-1
Bank	0	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	-1
TE	0	0	0	0	0	0	0	0	0	-1	-1	-1	0	0	0	-1
AC	0	0	0	0	0	0	0	0	-1	0	-1	-1	0	0	0	-1
AU	0	0	0	0	0	0	0	0	-1	-1	0	-1	0	0	0	-1
M2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	-1
Shop	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	-1	-1	-1	-1
SELLER	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	-1	-1	-1
AUDITOR	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	-1	0	-1	-1
M3	0	0	0	0	0	0	-1	-1	-1	-1	-1	0	0	0	0	-1
Director	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 7.11: ROLES in the payment scheme

6.4. This brings out the same result as the one after weak revocation from AP, QC, M1 by SSO. Note that all implied revocations upward in the role hierarchy should be within the revocation range of the administrative roles that are active in a session. For instance, if Alice activates APSO and tries to strongly revoke Bob's membership from M1, she is not allowed to proceed because M1 is out of the APSO's can revoke range in Table 7.9.

The user-role membership is required to revoke when a role is wrongly assigned to a user. Whether or not an administrator can revoke the membership depends on revocation relations $Can-revoke, Can-revoke-M$ and $Can-revoke-IM$. For example, if the membership is mobile membership, the relation $Can-revoke-M$ is used. The algorithms in this chapter can be used to check which administrator can revoke the wrong membership.

7.6 Related work

There are several other related works on role-based access control models with mobility of user-role relationship [94], user-role assignment mechanisms developed for web-based intranets [95] and Oracle system [4].

An administrative role based access control model for administration of roles was introduced in [94]. The authors analyzed several subtle issues such as user-role assignment, permission-role assignment with motilities of user-role membership and permissions-role membership. The conception of mobility is also appeared in this chapter. However, our work substantially differs from that proposal in two aspects. First, paper [94] only introduced the definition of mobility of user-role membership in user-role assignment. By contrast, we thoroughly discuss various cases and focus on possible problems in user-role assignment with mobility of use-role

relationship. Second, paper [94] describes the management of user-role assignment with the mobility, but they did not mention conflicts when assigning roles to users. Therefore, there is no support to deal administrative roles with regular roles in the proposal, especially mobile and immobile members. By contrast, we present a number of specialized authorization algorithms for access control which allow administrators to authorize a role as mobile and immobile member to users or revoke them from users. These algorithms provide a rich variety of options that can deal the document of administrative roles with regular roles as mobile and immobile members.

Using RBAC with SQL in Oracle system is also discussed [4]. RBAC are partly used in [4]. Permissions can be assigned to users directly in Oracle system but not in our work. Another difference between our work and the work in Oracle system is in revocation process. We have analyzed the weak revocation and the strong revocation that depend on different revocation requirements. However, the revocation in [4] is as follows:

If you revoke a role to which other roles have been granted, the entire set of privileges associated with every role is revoked. Of course, if any of those roles and privileges had been granted directly to a user or a role affected by the revoke, they can still exercise the related privileges through the direct grant.

Finally, a user-role assignment for web-based intranet has been proposed in [95]. In the model in [95], the RBAC/web system with user-role assignment model (URA97) [93] is extended to decentralize the details of RBAC administration on the web. Authorizations can be given by administrators through prerequisite conditions. By contrast, this chapter focuses on the consistency problems on user-role assignment and how to provide algorithms for these problems. These algorithms can be

applied not only web-based intranet but also all systems with RBAC management. The authorization granting algorithms are used to find conflicts and prevent some secret information to be derived while the strong revocation algorithms are used to check whether a role still has permissions of another role.

7.7 Conclusions

This chapter has provided new authorization allocation algorithms for RBAC that are based on relational algebra operations. They are the authorization granting algorithm, weak revocation algorithm and strong revocation algorithm for user-role assignments. The algorithms can automatically check conflicts when granting more than one role to a user in a system. They can prevent users from accessing unauthorized use of facilities when users change position within the organization and demand the modification of security rights. The roles can be allocated without compromising the security in RBAC and provide secure management for systems. The complexities of the algorithms are also analyzed. These results have been extended to enhanced authorization algorithms with mobility of user-role relationship. Furthermore, we have discussed how to use the algorithms for the electronic payment scheme.

Chapter 8

Formal Authorization Allocation Approaches for PRA Based on Relational Algebra Operations

Our aim in this chapter is to analyze problems that may arise in permission-role assignments (PRA) and to develop authorization allocation algorithms to address the problems within permission-role assignments. The algorithms are extended to the case of PRA with the mobility of permission-role relationship. Comparisons with other related work are also discussed.

This chapter is based on a published paper [118].

8.1 Introduction

Similar to the last chapter, there is also a consistency problem may arising within permission-role assignments. When there are hundreds of permissions and thousands of roles in a system, it is very difficult to maintain consistency problems because it may change the authorization level of roles, or imply high-level confidential information to be derived when more than one permission is requested and granted to roles. The permissions assigned to a role by administrators may conflict.

For example, the permission for approving a loan in a bank is conflicting with the permission of funding a loan. These two permissions cannot be assigned to a role; however, because of role hierarchies, a role may still have these permissions even if they have been revoked from the role. In the latter case, a user with this role is able to access objects in the permission and has operations on the objects. There are evident problems with the processes of assigning and revocation.

Authorization granting problem – How to check whether a permission is in conflict with the permissions of a role?

Authorization revocation problem – How to find whether permissions of a role have been revoked from the role or not?

For example, Figure 8.1 shows a system administrative role (BankSO) in a bank to manage regular roles such as AUDITOR, TELLER, ACCOUNT_REP and MANAGER. Role MANAGER inherits AUDITOR and TELLER. ACCOUNT_REP has a SSD relationship with AUDITOR as well as DSD relationship with TELLER.

The administrative role BankSO can assign audit permission or cash operation permission to a role but not both, otherwise it compromises the security of a bank system. Our aim is to provide relational algebra algorithms to solve the problems and then automatically check conflicts when assigning and revoking.

Based on the database and its tables such as ROLES, SEN-JUN in the last chapter, this chapter is going to develop formal approaches to check the conflicts and thereby help allocate the permissions without compromising the security. The formal approaches are based on relational structure and relational algebra operations. To my knowledge, this is the first attempt in this area to develop formal approaches for

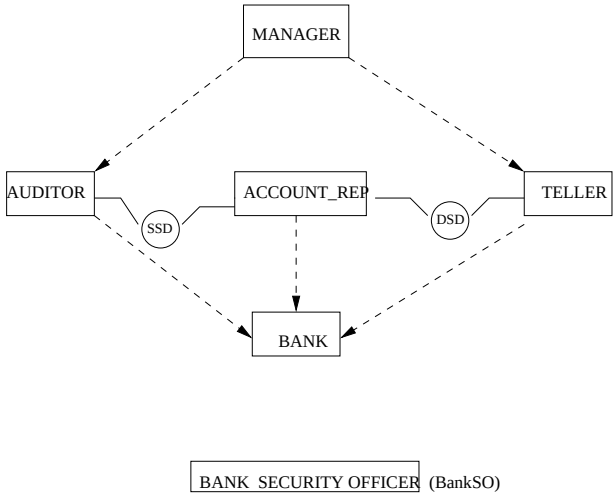

Figure 8.1: Administrative role and role Relationships in a bank

permission allocation and conflict detection.

The ROLES relation in Figure 8.1 is in Table 8.1. The attribute TELLERC shows whether the role TELLER is conflicting with the RoleName in the relation or not. For instance, in the third tuple, a user with role TELLER has conflicts with the role AUDITOR.

RoleName	MANAC	AUDC	ACCOUNT_REPC	TELLERC
MANAGER	0	0	0	0
AUDITOR	-1	0	-1	-1
ACCOUNT_REP	-1	-1	0	-1
TELLER	-1	-1	-1	0

Table 8.1: The relation ROLES in Figure 8.1

SEN-JUN - This is a relation of roles in a system. Senior is the senior of the two roles. Table 8.2 expresses the SEN-JUN relationship in Figure 8.1.

The new tables like PERM and ROLE_PERM are needed.

Senior	Junior
MANAGER	AUDITOR
MANAGER	TELLER
TELLER	BANK
AUDITOR	BANK

Table 8.2: SEN-JUN table in Figure 8.1

PERM - This is a relation of {PermName, Oper, Object, ConfPer }:

PermName is the primary key for the table, and is the name of the permission in the system.

Oper is the name of the operation granted. It has information about the object that the operation is granted on.

Object is the database item that can be accessed by the operation. It can be a database, a table, a view, an index or a database package.

ConfPer is a set of permissions that conflicts with the PermName in the relation.

For example, a staff member in a bank cannot have both permissions of approval and funding as well as both permissions of audit and teller. The relation of PERM can be expressed as Table 8.3.

PermName	Oper	Object	ConfPerm
Approval	approve	cash or check	Funding
Funding	invest	cash	Approval
Audit	audit	record	Teller
Teller	transfer	cash	Audit

Table 8.3: An example of the relation PERM

ROLE-PERM - is a relationship between the ROLES and the PERM, listing what

permissions are granted to what roles. It has two attributes:

RoleName is a foreign key RoleName from the table ROLES.

PermName is a foreign key PermName from the table PERM which is assigned to the role.

Suppose the permission Approval is assigned to role TELLER and the permission Funding to role MANAGER, Table 8.4 expresses the permission-role relationship.

RoleName	PermName
MANAGER	Funding
TELLER	Approval

Table 8.4: An example of ROLE-PERM table

Based on these relations, we describe the Authorization granting algorithm and revocation algorithms in this chapter.

The chapter is organized as follows. Relational algebra-based authorization granting and revocation algorithms are developed in the next section. The extensions of the algorithms are described in section 3. In section 4, the formal authorization approaches are applied to a payment scheme. Comparisons with related work are discussed in section 5 and the conclusions are in section 6.

8.2 Authorization granting and revocation algorithms for PRA

We develop granting and revocation algorithms for PRA based on relational algebra in this section. The notion of a *Prerequisite conditionp, Can-assignp* and *Can-*

revokep mentioned before is a key part in the processes of permission_role assignment. The *Prerequisite conditionp* is used to test whether or not permission can be assigned to roles while the *Can-assignp* is used to verify what role range's permissions an administrator can assign.

For a given set of roles R let CR denote all possible prerequisite conditions that can be formed using the roles in R. Not every administrator can assign permission to a role. The relation of **Can-assignp** $\subseteq AR \times CR \times 2^R$ provides what permissions can be assigned by administrators with prerequisite conditions, where AR is a set of administrative roles.

For example, the meaning of *Can-assignp* (x, y, Z) is that a member of the administrative role x can assign a permission whose current membership satisfies the prerequisite condition y to be a member of roles in range Z.

Permission-role assignment (PA) is authorized by *Can-assignp* relation. Table 8.5 shows the *Can-assignp* relations with the prerequisite conditions in the example.

Admin.role	Prereq.ConditionP	Role Range
BankSO	BANK $\wedge\overline{TELLER} \wedge \overline{AUDITOR}$	[ACCOUNT_REP, ACCOUNT_REP]
BankSO	BANK $\wedge\overline{TELLER} \wedge \overline{ACCOUNT\_REP}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge\overline{AUDITOR} \wedge \overline{ACCOUNT\_REP}$	[TELLER, TELLER]

Table 8.5: Can-assignp relation in Figure 8.1

Supposing an administrator role ADrole wants to assign a permission p_j to a role r with a set of permissions P. P^* is an extension of P, $P^* = \{p | p \in P\} \cup \{p | \exists r' \in R, r' < r, (p, r') \in PA\}$. There are two major steps in the following permission granting algorithm. The first step is to check whether the ADrole can assign the permission p_j to r or not. The set of Prerequisite conditionp associated with ADrole

can be obtained from the table *Can-assignp* while the set of roles associated with permission p_j is obtained from the table *ROLE-PERM*. The ADrole can build the membership of permission p_j and role r only if there is a role in the both sets. This means permission p_j satisfies prerequisite condition. The second step is to determine whether the permission p_j conflicts with the permissions of r or not, in other words, whether p_j conflicts with permission set P^* or not. The set of conflicting permissions of p_j can be retrieved from table *PERM*. Permission p_j is conflicting with role r if the intersection of the set and P^* is not empty. The details are below.

Authorization granting algorithm Grantp(ADrole, r, p_j)

Input: ADrole, role r and a permission p_j.

Output: true if ADrole can assign the permission p_j to r with no conflicts; false otherwise.

Begin:

Step 1. /* *Whether the ADrole can assign the permission* p_j *to* r *or not* */

Let

$$S = \pi_{Prereq.ConditionP}(\sigma_{admin.role=ADrole}(Can - assignp))$$

/* *S is a set of prerequisite conditionP associated with ADrole* */

and

$$R = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLE - PERM))$$

/* *R is a set of role associate with the permission* p_j */

if $S_1 = S \cap R \neq \phi$,

then there exists role $r_1 \in S_1$, such that $(p_j, r_1) \in PA$ and

$$r_1 \in \pi_{Prereq.ConditionP}(\sigma_{admin.role=ADrole}(Can - assignp)$$

go to step 2

/* p_j *is satisfied the prerequisite conditionP to be assigned by ADrole in Can-assign* */

else

return false and stop.

/**the admini.role has no right to assign the permission* p_j *to role* r */

Step 2. /**whether the permission* p_j *is conflicting with permissions of* r *or not*, in other words, whether p_j is conflicting with permission set P^* or not*/

Let

$$ConfPermS = \pi_{ConfPerm}(\sigma_{PermName=p_j}(PERM))$$

/* *It is the conflicting permission set of the permission* p_j */

if $ConfPermS \cap P^* \neq \phi$,

then

return false; /* p_j is a conflicting permission with role r */

else

return true.

/* p_j *is not a conflicting permission with* r */ ◇

Theorem 8.1 *The authorization granting algorithm can prevent conflicts when assigning a permission to a role.*

Proof Assuming an administrator role ADrole wants to assign a permission p_j to a role which associates with a permissions set P. While the step 1 in the algorithm

has checked whether the ADrole can assign the permission p_j to a role or not, the second step has decided whether the permission p_j is conflicting with permissions in P^* or not. Indeed, p_j can be assigned to the role if for all $p_i \in P^*$, p_i is not in the conflicting permission set of p_j. Otherwise p_j is a conflicting permission with P^*. $\diamond$

The authorization granting problem is solved by the authorization granting algorithm. Computing S in the first step takes time proportional to n^2 if n is presented as the number of roles. This is because computing $\sigma_{admin.role=ADrole}(Can - assignp)$ and $\pi_{Prereq.ConditionP}(\sigma_{admin.role=ADrole}(Can - assignp))$ needs time $O(n)$. It spends time $O(n)$ to compute R and $O(n^2)$ for S_1. Thus, the step 1 takes time $O(n^2)$. In the second step, the computations of $\sigma_{PermName=p_j}(PERM)$ and $\pi_{ConfPerm}(\sigma_{PermName=p_j}(PERM))$ spend time $O(m)$ and $O(1)$ respectively when there are m permissions in the system. It needs time $O(m^2)$ to decide whether there is a permission in $ConfPermS \cap P^*$ or not . Therefore the total time spent in the authorization granting algorithm is proportional to $(n^2 + m^2)$.

Corollary 8.1 *The authorization granting algorithm has time complexity* $O(n^2 + m^2)$ *for the case of* n *roles and* m *permissions in a system.*

There are related subtleties that arise in RBAC concerning the interaction between granting and revocation of permission-role membership. A relation **Can-revokep**$\subseteq AR \times 2^R$ provides which permissions in what role range can be revoked. Table 8.6 gives an example of the *Can-revokep* relation. We have two revocation algorithms, one is a weak revocation algorithm that is for explicit member of a role only, the other one is a strong revocation algorithm that is used to delete explicit memberships between permissions and roles as well as implicit memberships.

Admin.role	Role Range
BankSO	[Bank, MANAGER)

Table 8.6: An example of Can-revokep

The meaning of *Can-revokep* $(BankSO, [Bank, MANAGER))$ in Table 8.6 is that a member of the administrative role BankSO can revoke the membership of a permission from any role in [Bank, MANAGER).

Following are two algorithms for revocation of a permission p_j from a role r by an administrative role ADrole. In the weak revocation approach, only whether or not permission p_j is an explicit member of role r needs to be determined. Operation of the weak revocation has no effect when the permission p_j is not an explicit member of the role r. The role set associated with p_j is gained from the relation of *ROLE-PERM* while the role set of role range with ADrole is obtained from the relation *Can-revokep*. The permission p_j can be revoked if the intersection of these two role sets is not empty.

Weak revocation Algorithm Weak_revokep(ADrole, r, p_j)

Input: ADrole, a roles r and a permission p_j.

Output: true if ADrole can weakly revoke role p_j from r; false otherwise.

Begin:

if $p_j \notin \{p|(p, r) \in PA\}$,

return false;

/* *there is no effect with the operation of the weak revocation since the permission* p_j *is not an explicit member of the role* r*/

else /* p_j *is an explicit member of* r */

Let

RevokeRange $= \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep))$

/* *The role range can be revoked by ADrole*/

and

$$Roleswithp_j = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLEPERM))$$

/**Roles with permission* p_j*/

If $RevokeRange \cap Roleswithp_j \neq \phi$

return true; /**the* p_j *can be revoked* */

else

return, false.

/* *ADrole has no right to revoke the permission* p_j *from the role* */ ◇

We have the following result with the weak revocation algorithm.

Theorem 8.2 *A permission* p_j *is revoked by the weak revocation algorithm* $Weak\_revokep(ADrole, r, p_j)$ *if the permission is an explicit member of the role* r *and the ADrole has the right to revoke* p_j *from the Can-revokep relation.* ◇

It takes time $O(m)$ to check if $p_j \notin \{p|(p, r) \in PA\}$ when there are m permissions in a system. The computations of $\sigma_{admin.role=ADrole}(Can - revokep)$ and $\pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep))$ take time $O(n)$ when there are n roles in the system. The process of $\pi_{RoleName}(\sigma_{PermName=p_j}(ROLEPERM))$ needs time $O(n)$. To check whether $RevokeRange \cap Roleswithp_j \neq \phi$ or not needs time $O(n^2)$. Thus, the time complexity of the Weak revocation algorithm is $O(n^2 + m)$.

Corollary 8.2 *Weak revocation algorithm has time complexity* $O(n^2 + m)$ *when there are* n *roles and* m *permissions in a system.*

A role still owns a permission of a system, which has been weakly revoked, if the role is senior to another role associated with the permission. To solve the authorization revocation problem, we need strong revocation, which requires revocation of both explicit and implicit membership. Strong revocation of a permission's membership in role r requires that the permission be removed not only from explicit membership in r, but also from explicit and implicit membership in all roles junior to r. Strong revocation therefore has a cascading effect up-wards in the role hierarchy. The first step in the strong revocation algorithm is to test whether p_j is in P^* or not. If the test is negative, that means p_j is neither an explicit member nor an implicit member of the role r. When this case occurs, the strong revocation has no effect for the role. Otherwise, p_j is either an explicit member or an implicit member of r. In this step, the membership of p_j is revoked from r if $pj \in P$; then the role set of roles that are junior to role r can be retrieved from the relation *SEN-JUN*. For all roles in both the set and P, the relationships between these roles and permission p_j are revoked.

Strong revocation algorithm Strong_revokep(ADrole, r, p_j)

Input: ADrole, a role r and a permission p_j.

Output: true, if it can strong revoke the permission p_j from r; false otherwise.

Begin:

if $p_j \notin P^*$,

return false; /* *there is no effect of the strong revocation since the permission is not an explicit and implicit member of the role* r */

else,

1. *if* $p_j \in P$, do Weak_revokep(ADrole, r, p_j);

/*p_j *is weakly revoked from* r*/;

2. Suppose

$Jun = \pi_{Junior}(\sigma_{Senior=r}(SEN - JUN))$,

for all $y \in Jun$, do Weak_revokep(ADrole, y, p_j);

/**the permission* p_j *is weakly revoked from all such* $y \in Jun$*/.

If all the weak revocations are successful,

return true;

otherwise,

return false. /* if one weak revocation cannot finish*/ ◇

It should be noted that strong revocation algorithm does not work if ADrole has no right to revoke p_j from any role in Jun. We have the following consequence.

Theorem 8.3 *The explicit and implicit member of permission* p_j *are revoked from the role* r *by the Strong revocation algorithm* $strong\_revokep(ADrole, r, p_j)$ *if the ADrole has the right to revoke* p_j *from the Can-revokep relation.*

Corollary 8.3 *The authorization revocation problem is solved by the Weak revocation algorithm and Strong revocation algorithm.*

Time $O(m)$ is needed to check whether $p_j \notin P^*$ if there are m roles in a system. The computations of $\sigma_{Senior=r}(SEN - JUN)$ and $\pi_{Junior}(\sigma_{Senior=r}(SEN - JUN))$

take time proportional to l $(l < n)$ where l is the number of tuples in the relation SEN-JUN and n is the number of roles in the system. It takes time proportional to (n^2+m) to do a weak revocation. A role may be junior $(n-1)$ roles, hence all weak revocations need time $O(n*(n^2+m))$. Other operations $p_j \in P$ and $y \in Jun$ takes time $O(m)$ and $O(n)$ respectively. Therefore the total time spent with the Strong revocation algorithm is $O(n*(n^2+m))$.

Corollary 8.4 *The strong revocation algorithm has time complexity* $O(n*(n^2+m))$ *when there are* n *roles and* m *permissions in a system.*

8.3 Extensions of the algorithms with mobility of permissions

Similar to the mobility of user-role relationship, permissions can also be assigned to roles as mobile and immobile members. There are four kinds of permission-role membership for a given role x [94].

1: *Explicit Mobile Member EMPx*

$EMPx = \{p, (p, Mx) \in PA\}$

2: *Explicit Immobile Member EIMPx*

$EIMPx = \{p, (p, IMx) \in PA\}$

3: *Implicit Mobile Member ImMPx*

$ImMPx = \{p, \exists x' < x, (p, Mx') \in PA\}$

4: *Implicit Immobile Member ImIMPx*

$ImIMPx = \{p, \exists x' < x, (p, IMx') \in PA\}$

Admin.role	Prereq.ConditionPM	Role Range
BankSO	BANK	[BANK, BANK]
BankSO	BANK $\wedge\overline{TELLER}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge\overline{AUDITOR}$	[TELLER, TELLER]
BankSO	TELLER $\wedge$ AUDITOR	[MANAGER, MANAGER]

Table 8.7: Can-assignp-M in the example

A **prerequisite conditionPM** is evaluated for a permission p by interpreting role x to be true if

$$p \in EMx \vee (p \in ImMx \wedge p \notin EIMx)$$

and $\bar{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

In other words x denotes mobile membership (explicit or implicit) and $\bar{x}$ denotes absence of any kind of membership.

For a given set of roles R let CR denote all possible prerequisite conditions with mobility of permission-role relationship that can be formed using the roles in R. Not every administrator can assign a role to a user. The following relations provide what permissions an administrator can assign as mobile members or immobile members with prerequisite conditions.

Can-assignp-M is a relation of $\subseteq AR \times CR \times 2^R$, which is used for permission-role assignments with mobile members; where AR is a set of administrative roles. Permission-role assignments with immobile members are authorized by the relation **Can-assignp-IM** $\subseteq AR \times CR \times 2^R$ $\diamond$

Permission-role assignment(PA) is authorized by *Can-assignp-M* and *Can-assignp-IM* relations. Table 8.7 and Table 8.8 shows the *Can-assignp-M* and *Can-assignp-IM* relations with the prerequisite conditions in the bank example.

Admin.role	Prereq.ConditionPM	Role Range
BankSO	BANK	[BANK, BANK]
BankSO	BANK $\wedge\overline{TELLER}$	[AUDITOR, AUDITOR]
BankSO	BANK $\wedge\overline{AUDITOR}$	[TELLER, TELLER]

Table 8.8: Can-assignp-IM in the example

The meaning of *Can-assignp-M*$(BankSO, BANK, \{TELLER, AUDITOR\})$ is that a member of the administrative role BankSO can assign a permission whose current membership satisfies the prerequisite condition BANK to be a mobile member of roles TELLER and AUDITOR.

Supposing an administrator role ADrole wants to assign a permission p_j to role r with a set of permissions P which has mobile and immobile memberships with r. The p_j has mobile or immobile membership with r if ADrole can assign without conflicts. The following algorithm applies to both of mobile and immobile members. P^* is an extension of P, $P^* = \{p|p \in P\} \cup \{p|\exists r', r' < r, (p, r') \in PA\}$.

Authorization granting algorithm GrantMP(ADrole, P, p_j)

Input: ADrole, role r and a permission p_j.

Output: true if ADrole can assign the permission p_j to r with no conflicts; false otherwise.

Begin:

Step 1. /* *Whether the ADrole can assign the permission* p_j *to* r *as mobile or immobile member or not* */

Suppose $S_{M1} = S_M \cap R$ and $S_{IM1} = S_{IM} \cap R$ where

$$S_M = \pi_{Prereq.ConditionPM}(\sigma_{admin.role=ADrole}(Can - assignp - M))$$

$$S_{IM} = \pi_{Prereq.ConditionPM}(\sigma_{admin.role=ADrole}(Can - assignp - IM))$$

$$R = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLE - PERM))$$

if p_j will be a mobile member of r and $S_{M1} \neq \phi$,

then there exists role $r_1 \in S_{M1}$, such that

$r_1 \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assign - M))$ and $(p_j, r_1 \in PA)$,

go to step 2

/* *p_j is in the range to be assigned as a mobile member by ADrole in Can-assignp-M* */

if p_j will be an immobile member of r and $S_{IM1} \neq \phi$,

then there exists role $r_i \in S_{IM1}$, such that

$r_i \in \pi_{RoleRange}(\sigma_{\{ADrole,r\}}(Can - assignp - IM))$ and $(p_j, r_i \in PA)$

go to step 2

/* *p_j is in the range to be assigned as an immobile member by ADrole in Can-assign-IM* */

else

return false and stop.

/**the admini.role has no right to assign the role r_j as a mobile or immobile member to R* */

Step 2. /**whether the permission p_j is conflicting with permissions of r or not**/

Let

$$ConfPermS = \pi_{ConfPerm}(\sigma_{PermName=p_j}(PERM))$$

/* *It is the conflicting permission set of the permission* p_j */

if $ConfPermS \cap P^* \neq \phi$,

then

return false;

/* p_j *is a conflicting permission with role* r */

else

return true.

/* p_j *is not a conflicting permission with* r */ ◇

This algorithm provides a way to decide whether a permission can be assigned to a role as mobile or immobile member. For mobile member, S_{M1} cannot be empty, and for immobile member, S_{IM1} cannot be empty.

Theorem 8.4 *The authorization granting algorithm can prevent conflicts when assigning a permission to a role with mobile and immobile memberships.*

Proof Assuming an administrator role ADrole wants to assign a permission p_j as a mobile member to a role which associates with a permission set P. Step 1 in the algorithm has checked whether the ADrole can assign p_j as a mobile member to the role or not, and the second step has decided whether the permission p_j conflicts with permissions in P^* or not. Indeed, p_j can be assigned to the role if for all $p_i \in P^*$, p_i is not in the conflicting permission set of p_j. Otherwise p_j is a conflicting permission with P^*. ◇

Similar to the time complexity analysis in the last section, we have the following corollary.

Corollary 8.5 *The authorization granting algorithm has time complexity* $O(n^2)$ *for the case of* n *roles in a system.*

Now we consider revocation of permission-role membership. Similar to *Can-assignp-M* and *Can-assignp-IM* relations in granting a permission to a role, there are *Can-revokep-M* and *Can-revokep-IM* relations.

Relations **Can-revokep-M** $\subseteq AR \times CR \times 2^R$ and **Can-revokep-IM** $\subseteq AR \times CR \times 2^R$ show which role range of mobile membership and immobile membership administrative roles can revoke respectively, where AR is a set of administrative roles. $\diamond$

The meaning of *Can-revokep-M* $(ShopSO, SHOP, [SHOP, MANAGER))$ in Table 8.9 is that a member of the administrative role ShopSO can revoke mobile membership of a permission from any role in [SHOP, MANAGER) subject to the prerequisite condition SHOP. *Can-revokep-IM* is similar with respect to immobile membership.

Admin.role	Prereq.ConditionPRM	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 8.9: Can-revokep-M

The evaluation of a prerequisite condition for the revoke model is different from the grant model. In the revoke model a **prerequisite conditionPRM** is evaluated

Admin.role	Prereq.ConditionPRM	Role Range
ShopSO	SHOP	[SHOP, MANAGER)

Table 8.10: Can-revokep-IM

for a permission p by interpreting role x to be true if

$$p \in EMx \vee p \in EIMx \vee p \in ImMx \vee p \in ImIMx$$

and $\bar{x}$ to be true if

$$p \notin EMx \wedge p \notin EIMx \wedge p \notin ImMx \wedge p \notin ImIMx$$

Due to role hierarchy, a role x' has all permissions of role x when $x' > x$. A user with two roles $\{x', x\}$ still has the permissions of x if only to revoke x from the user. To solve the authorization revocation problem along with mobility of permission, we need to revoke the explicit member of a permission first if a role is an explicit member, then revoke the implicit member.

Following are two algorithms for revocation of a permission p_j as mobile or immobile members from a set of permission P by an administrative role ADrole, where P is a set of permissions which are assigned to a role r. The first one is the weak revocation algorithm and the second is the strong revocation algorithm. The weak revocation only revokes explicit mobile and immobile memberships from r and does not revoke implicit mobile and immobile memberships but the strong revocation revokes both explicit and implicit mobile and immobile members. The structure of the weak revocation algorithm is shown in Figure 8.2.

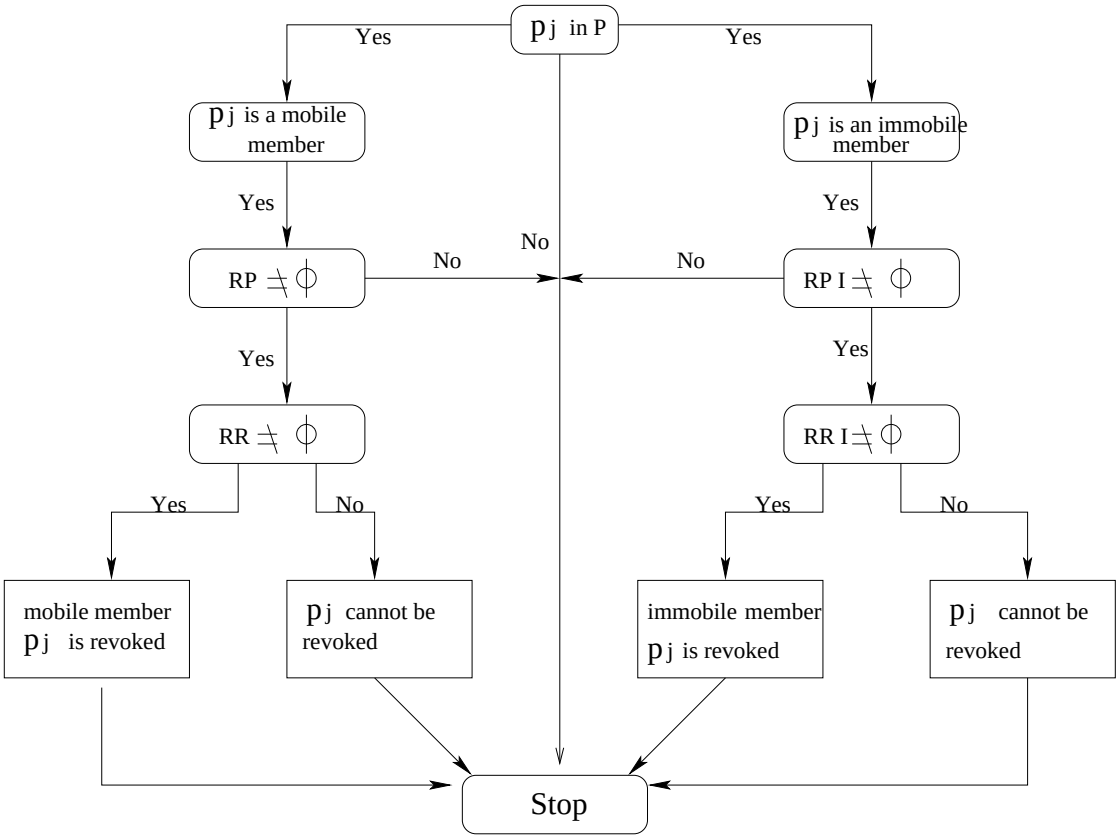

Figure 8.2: Structure of weak revocation algorithm for permission

Weak revocation Algorithm Weak_revokeMP(ADrole, r, p_j)

Input: ADrole, a roles r and a permission p_j.

Output: true if ADrole can weakly revoke role p_j from r; false otherwise.

Begin:

if $p_j \notin P = \{p|(p,r) \in PA\}$,

return false;

/ there is no effect with the operation of the weak revocation since the permission* p_j *is not an explicit member of the role* r**/*

*else /** p_j *is an explicit member of* r **/*

Case1: p_j is an mobile member of r,

$$Roleswithp_j = \pi_{RoleName}(\sigma_{PermName=p_j}(ROLE - PERM))$$

/* *Roles with permission* p_j */

$$PreM = \pi_{Prereq.ConditionPRM}(\sigma_{admin.role=ADrole}(Can - revokep - M))$$

/**Prerequisite condition with ADRole* */

if $RP = Roleswithp_j \cap PreM \neq \phi$

$$RevokeRangeM = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep - M))$$

if $RR = Roleswithp_j \cap RevokeRangeM \neq \phi$,

return, true. /* *the mobile member* p_j *is revoked* */

else return false;

/* *the mobile member* p_j *cannot be revoked since the role* r *is not in the role range to be revoked* */

else return false and stop.

/**The* p_j *does not satisfy the prerequisite conditions** /

Case 2: if p_j is an immobile member of r

$$PreIM = \pi_{Prereq.ConditionPRM}(\sigma_{admin.role=ADrole}(Can - revokep - IM))$$

/**Prerequisite condition with ADRole** /

if $RPI = Roleswithp_j \cap PreIM \neq \phi$

$$RevokeRangeIM = \pi_{RoleRange}(\sigma_{admin.role=ADrole}(Can - revokep - IM))$$

if $RRI = Roleswithp_j \cap RevokeRangeIM \neq \phi$,

return true, /* *the immobile member* p_j *is revoked*

else return false ; /* *the immobile member* p_j *cannot be revoked* */

else return false and stop.

/**The* p_j *does not satisfy the perrequisite conditions**/ ◇

The weak revocation algorithm can be used to check whether an administrator can weakly revoke mobile and immobile memberships from roles or not. We have the following result with the weak revocation algorithm.

Theorem 8.5 *A permission* p_j *as mobile or immobile member is revoked by the weak revocation algorithm* $Weak\_revoke(ADrole, r, p_j)$ *if the permission is an explicit member of role* r *and the ADrole has the right to revoke* p_j *from the Can-revoke-M and Can-revoke-IM relations.* ◇

A role still owns a permission of a system, which has been weakly revoked, if the role is senior to another role associated with the permission. To solve the authorization revocation problem, we need strong revocation, which requires revocation of both explicit-implicit membership and mobile-immobile memberships. Strong revocation of a permission's membership in role r requires that the permission be removed not only from explicit mobile and immobile membership in r, but also from explicit, and implicit mobile and immobile membership in all roles junior to r.

Strong revocation algorithm Strong_revoke(ADrole, r, p_j)

Input: ADrole, a role r and a permission p_j.

Output: true, if it can strong revoke the permission p_j from r; false otherwise.

Begin:

if $p_j \notin P^*$,

return false;

/ there is no effect of the strong revocation since the permission is not an explicit and implicit member of the role r */*

else,

1. *if* $p_j \in P$, do Weak_revoke(ADrole, r, p_j);

/p_j is weakly revoked from r as mobile or immobile */*;

2. Suppose

$Jun = \pi_{Junior}(\sigma_{Senior=r}(SEN - JUN))$,

for all $y \in Jun \cap P$,

if $p_j \in R$ is a mobile member of the role y, do Weak_revoke(ADrole, y, p_j) as $p_j \in EMy$;

if $p_j \in R$ is a immobile member of the role y, do Weak_revoke(ADrole, y, p_j) as $p_j \in EIMy$;

*/*the permission p_j is weakly revoked from all such $y \in Jun \cap P$*/*.

If all the weak revocations are successful,

return true;

otherwise,

return false. /* if one weak revocation cannot finish*/ ◇

We have the following consequence.

Theorem 8.6 *The explicit mobile and immobile and implicit mobile and immobile members of role p_j are revoked from a role by the Strong revocation algorithm Strong_revoke$(ADrole, r, p_j)$ if the ADrole has the right to revoke p_j from the Can-revokep-M and Can-revokep-IM relations.*

Corollary 8.6 *The authorization revocation problem is solved by the Weak revocation algorithm and Strong revocation algorithm.*

In the remainder of this chapter, the new relational algebra approaches are used with a payment scheme.

8.4 Illustration of the relational algebra algorithms with permission-role assignments

The new relational algebra algorithms for permission-role assignments are applied to the payment scheme in this section. Only the applications of the algorithm, without mobility of role-permission relationship, are introduced.

An application of the authorization granting algorithm

Figure 6.4 shows that role E is the most junior to all other employees in the new system and role Director (DIR) is the most senior to all other employees. Figure 6.5 shows the administrative role hierarchy which co-exists with the roles in Figure 6.4. The senior-most role is the Senior Security Officer (SSO). Our interest is in the administrative roles junior to SSO. These consist of three security officer roles (APSO, BankSO and ShopSO) with the relationships illustrated in the Figure 6.5. Table 8.3 and Table 8.11 show part of the relations between permissions and between roles in the scheme.

RoleName	PermName
Director (DIR)	Funding
Director (DIR)	Approval
Director (DIR)	Teller
TELLER	Approval
FPS	Approval
Bank	Teller

Table 8.11: ROLE-PERM table in the scheme

Based on the role hierarchy in Figure 6.4 and administrative role hierarchy in Figure 6.5, we define the *Can-assignp* relation shown in Table 8.12.

Admin.role	Prereq.ConditionP	Role Range
NSSO	DIR	[M1, M1]
NSSO	DIR	[M2, M2]
NSSO	DIR	[M3, M3]
APSO	FPS $\wedge\overline{OP}$	[QC, QC]
APSO	FPS $\wedge\overline{QC}$	[OP, OP]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AU}$	[AC, AC]
BankSO	FPS $\wedge\overline{TE} \wedge \overline{AC}$	[AU, AU]
BankSO	FPS $\wedge\overline{AU} \wedge \overline{AC}$	[TE, TE]
ShopSO	FPS $\wedge\overline{SALER}$	[AUDITOR, AUDITOR]
ShopSO	FPS $\wedge\overline{AUDITOR}$	[SALER, SALER]

Table 8.12: Can-assignp in the payment example

Let us consider the NSSO tuples in Table 6.4 (the analysis for APSO, BankSO and ShopSO are similar). The first tuple authorizes NSSO to assign permissions with the prerequisite condition role DIR into members of M1 in the AP agent (AP). The second and third tuples authorize NSSO to assign permissions with the prerequisite condition DIR to be a member of M2 and M3 respectively. Similarly, the fourth tuple authorizes APSO to assign permissions with the prerequisite condition $FPS \wedge \overline{OP}$ to be members of operators (QC). The fourth and fifth tuples show that the APSO can grant a permission that is a member of the AP agent into one but not both of QC and

OP. This illustrates how mutually exclusive roles can be forced by permission-role assignment.

Assume the role FPS with permission set $P = \{Approval\}$ and $P^* = P = \{Approval\}$. The administrative role NSSO wants to assign the permission *Teller* to the role FPS. Using the granting algorithm *Grantp(NSSO, FPS, Teller)*, the first step,

$$\begin{aligned} S &= \pi_{Prereq.ConditionP}(\sigma_{Admin.role=NSSO}(Can - assignp)) \\ &= \{DIR\} \end{aligned}$$

and

$$\begin{aligned} R &= \pi_{RoleName}(\sigma_{PermName=Teller}(ROLE - PERM)) \\ &= \{DIR, Bank\} \end{aligned}$$

Since $R \cap S = \{DIR\} \neq \phi$. This means NSSO can assign permission *Teller* to role FPS.

The second step, based on Table 8.3

$$\begin{aligned} Conf - perm &= \pi_{ConfPerm}(\sigma_{PermName=Approval}(PERM)) \\ &= \{Funding\} \end{aligned}$$

and

$$Conf - perm \cap P^* = \phi.$$

Meaning there are no conflicts when assigning the permission Teller to role FPS.

Application of the authorization revocation algorithm

Table 8.13 and Table 8.2 give the *Can-revokep* and a part of senior-junior relationship of the payment scheme.

Admin.role	Role Range
NSSO	[FPS, DIR)
APSO	[AP, M1)
BankSO	[Bank, M2)
ShopSO	[Shop, M3)

Table 8.13: Can-revokep in the payment example

Based on the Table 8.11, The permission *Approval* is an explicit member of role DIR, TELLER and FPS in the scheme. If Alice, with the activated administrative role BankSO, weakly revokes *Approval*'s membership from TELLER, the revocation is successful by the weak revocation algorithm Weak_revokep(BankSO, TELLER, Approval). This is because

$$
\begin{aligned}
&RevokeRange \cap RoleswithApproval \\
&= \{TELLER\} \\
&\neq \phi
\end{aligned}
$$

where

$$
\begin{aligned}
&RevokeRange \\
&= \pi_{RoleRange}(\sigma_{Admin.role=APSO}(Can-revokep)) \\
&= [Bank, M2)
\end{aligned}
$$

and

$$
\begin{aligned}
&RoleswithApproval \\
&= \pi_{RoleName}(\sigma_{PermName=Approval}(ROLEPERM)) \\
&= \{DIR, TELLER, FPS\}
\end{aligned}
$$

Approval continues to be an implicit member of TELLER since FPS is junior to TELLER and *Approval* is an explicit member of FPS. It is necessary to note that Alice should have enough power in the session to weakly revoke *Approval* from explicitly assigned roles. For instance, if Alice has activated BankSO and then tries to weakly revoke *Approval* from FPS, she is not allowed to proceed because BankSO does not have the authority of weak revocation from FPS according to the *Can-revokep* relation in Table 6.5. Therefore, if Alice wants to revoke *Approval*'s explicit membership as well as implicit membership from TELLER by weak revocation, she

needs to activate NSSO and weakly revoke *Approval* from TELLER and FPS.

If Alice, with the activated administrative role NSSO, strongly revokes *Approval*'s membership from TELLER, then *Approval* is removed not only from explicit membership in TELLER, but also from explicit (and implicit) membership in all roles junior to TELLER. Actually, using the strong revocation algorithm Strong_revoke(NSSO, TELLER, Approval), $P = \{Approval\} = P^*$. It does need to do Weak_revoke (NSSO, TELLER, Approval) since $Approval \in P$. The junior set of role TELLER is {FPS}. Then the permission *Approval* has been removed from FPS as well as TELLER by running Weak_revoke (NSSO, TELLER, Approval) and Weak_revoke (NSSO, FPS, Approval). However, *Approval* still has a membership of DIR since it is not a junior role to TELLER based on the role hierarchy of Figure 6.4. This brings about the same result as weak revocation from TELLER and FPS by NSSO. Note that all implied revocations downward in the role hierarchy should be within the revocation range of the administrative roles that are active in a session. For instance, if Alice activates BankSO and tries to strongly revoke *Approval* from TELLER, she is not allowed to proceed because FPS is junior to TELLER but it is out of the BankSO's *Can-revokep* range in Table 6.5.

The permission-role membership is required to revoke when a permission is wrongly assigned to a role. Whether or not an administrator can revoke the membership depends on revocation relations $Can - revokep, Can - revokep - M$ and $Can - revokep - IM$. For example, if the membership is immobile membership, the relation $Can - revokep - IM$ is used. The algorithms in this chapter can be used to check which administrator can revoke the wrong membership.

8.5 Related work

There are several other related works on relational databases [76, 92].

The interaction between RBAC and relational databases are presented in [76]. Two experiments are described. One is a role-based front end to a relational database with discretionary access control. The other is a role graph to show the roles in a standard relational database. Some relational concepts like roles, users and permissions are provided. Our model also supports such concepts even though it has a large variety. However, the main difference between our algorithms and the scheme in [76] is that we focus on the solutions of the conflicts of roles and permissions, and the latter focuses on the correlation of RBAC with discretionary access controls. Their work discusses the relationship between roles and discretionary access controls, they do not address the allocation of permissions to roles without conflicts. In our work, we developed detailed algorithms for allocating roles and permissions and checking their conflicts.

An oracle implementation for permission-role assignment has been proposed in [92]. In [92], the difference between permission-role assignment (PRA97) and Oracle database management system was analyzed. Furthermore, through prerequisite conditions, the paper has demonstrated how to use Oracle stored procedures for implementation. However, the work in this chapter substantially differs from that proposal. Differences are due to the consistency problem that arises in [92]:

It is very difficult to keep the consistency by reflecting security requirements between global network objects and local network objects if there are hundreds of roles and thousands of users in a system.

This problem is completely overcome in our algorithms because the algorithms

focus on the conflicts between roles and permissions. The authorization granting algorithms are used to find conflicts and prevent some secret information from being derived while the strong revocation algorithms are used to check whether a role still has permissions of another role.

8.6 Conclusions

This chapter has provided new authorization allocation algorithms for permission-role assignments that are based on relational algebra operations. They are the authorization granting algorithm, weak revocation algorithm, and strong revocation algorithm. The algorithms can automatically check conflicts when granting more than one permission to a role in a system. They can prevent users associated with roles from accessing unauthorized use of facilities when the permissions of the roles are changed within the organization and demand the modification of security rights. The permissions can be allocated without compromising the security in RBAC and provide secure management for systems. The complexities of the algorithms are indicated. Furthermore, the extensions of the algorithms for mobility of permissions are extensively analyzed. Finally, we have discussed the related work in this area and how to use the algorithms in the electronic payment scheme.

Chapter 9

Conclusions and future work

This chapter lists the contributions of this dissertation and introduce future work. The contributions of this dissertation are presented in section 9.1 and section 9.2 explains the future research work.

9.1 Contributions

Software technology, including access scheme, payment, and access control management for e-commerce, is an important dimension in e-commerce. Although the importance of electronic-payment in e-commerce has been recognized for a long time, it has not received much attention in research literature while role based access control has been discussed. There are three enhancements in this dissertation. The first enhancement is a ticket-based access scheme for mobile and non-mobile users. The second is a scalable anonymity payment scheme and the third is formal authorization approaches for role based access control.

9.1.1 Enhancements on ticket-based access control scheme for mobile user

We have proposed a ticket-based access control model for mobile users. The model supports efficient authentication of users and service providers over different domains and can also be used by wired terminal users. Tickets are used to verify correctness of the requested service as well as to direct billing information to the appropriate user. The service providers can avoid roaming to multiple service domains, only contacting a Credential Centre to certify the user's ticket since tickets carry all authorization information needed for the requested services. The user can preserve anonymity and read a clear record of charges in the Credential Centre at anytime. Furthermore, the identity of misbehaving users can be revealed by a Trusted Centre. Other related work either has the weakness of not providing anonymity to users or solving only for particular mobile access problems.

9.1.2 Enhancements on anonymity payment scheme

A secure, scalable, anonymous, and practical payment protocol for Internet purchases has been presented. The protocol uses electronic cash for payment transactions. From the viewpoint of banks, this new protocol allows users are worried about disclosure of their identities to maintain anonymity. An agent provides a higher anonymous certificate and improves the security of the consumers. The agent certifies re-encrypted data after verifying the validity of the content from consumers, without requiring the private information of the consumers. With this new method, each consumer can get required anonymity level, depending on the available time, computation, and cost.

We also analyze how to prevent a consumer from spending a coin more than

once and how to use the proposed protocol for Internet purchases. After comparing with another scheme and discussing the properties of the new payment protocol, the new method has been proven more efficient, and can effectively prevent from eavesdropping, tampering, and "perfect crime". It is promising for electronic trades through the Internet.

9.1.3 Enhancements on formal authorization approaches for role based access control

We have developed formal authorization allocation algorithms for role-based access control (RBAC). The formal approaches are based on relational structure, and relational algebra and operations. The processes of user-role assignments and permission-role assignments are two important issues in RBAC because they may modify the authorization level or imply high-level confidential information to be derived while users change positions and request different roles and permissions. There are two types of problems which may arise in user-role assignment. One is related to authorization granting process. When a role is granted to a user, this role may conflict with other roles of the user or together with this role; the user may have or derive a high level of authority. The other is related to authorization revocation. When a role is revoked from a user, the user may still have the role from other roles. Similarly, there are two types of problems that may arise in permission-role assignments. One is related to authorization granting process. Conflicting permissions may be granted to a role, and as a result, users with the role may have or derive a high level of authority. The other is related to authorization revocation. When permission is revoked from a role, the role may still have the permission from other roles.

To solve these problems, authorization granting algorithms, and weak revocation and strong revocation algorithms that are based on relational algebra for user-role assignments and permission-role assignments have been presented. The algorithms can automatically check conflicts when granting more than one role (permission) to a user (a role) in a system. They can prevent users from accessing unauthorized use of facilities when users change position within the organization and demand the modification of security rights. The roles and permissions can be allocated without compromising the security in RBAC and provide secure management for systems. The complexities of the algorithms are also analyzed. Furthermore, the extensions of the algorithms are deeply analyzed that include the mobility of user-role and permission-role relationships. As shown in this thesis, the mobility of users and permissions are very significant and therefore some users with roles and some roles with permissions can be further assigned while some cannot. We have discussed how to use the algorithms for the electronic payment scheme. It gives ideas for using role-based access control to manage electronic payment system. The algorithms can be applied in system management which uses RBAC.

9.2 Future work

Based on the research work in this dissertation, we propose the following future research directions and issues.

9.2.1 Improvement of the payment scheme

We have built a consumer scalable anonymity payment protocol that provides different requirements of anonymity for users. When a system with the payment protocol has many users such as hundreds of millions, the data of consumers and coins in-

crease very quickly. The bank and the AP agent need to keep the data of consumers and coins that are used to verify and solve problems between consumers and shops, this may cause a bottleneck problem. In the future, we plan to develop a distributed payment solution to address the bottleneck issue.

9.2.2 Extension of formal authorization approaches for role based access control

In this dissertation we presented formal authorization approaches for user-role assignment and permission-role assignment which can be used to automatically check conflicts when granting roles (permission) to a user (a role) and revoking roles (permission) from a user (a role) in a system. The roles and permissions can be allocated without compromising the security in RBAC. In the future, we would like to extend formal authorization approaches, investigating formal authorization approaches for user-user assignment. We must also determine how these algorithms can be used to enforce an RBAC management system.

9.2.3 Electronic commerce with RBAC

There are several examples of role based access control for payment scheme. These examples provide a way to use RBAC to manage electronic payment schemes. However, the use of RBAC for generic electronic commerce systems is still far from being achieved. We like to analyze how RBAC management can be used in generic electronic payment systems and implement the system to prove its practicality. Then we will extend the results to generic electronic commerce systems.

9.2.4 Implementation issues

We would like to build tools that can check the syntax and the semantics of the ticket based access control system, consumer anonymity scalable payment scheme, and formal authorization approaches for role based access control. The tools might provide visual support for the payment scheme and authorization approaches to make them more efficient. The visual tool of formal authorization approaches, for example, would display all of the components such as role and permissions that can be used in database. In addition, it shows what components are used and are available for the authorization approaches. Furthermore, using the visualization tool, system administrators can easily check the current status of components in systems.

Index

Bibliography

[1] Ahn G. J. and Sandhu R. The rsl99 language for role-based separation of duty constraints. In *4th ACM Workshop on Role-Based Access Control*, pages 43–54. Fairfax, VA, 1999.

[2] Aline Baggio and Gerco Ballintijn and Maarten van Steen. Mechanisms for effective caching in the globe location service. In *Proceedings of the 9th workshop on ACM SIGOPS European workshop*, pages 55–60. ACM Press, 2000.

[3] Andreoli J., Pacull F., Pagani D. and Pareschi R. Multiparty negotiation of dynamic distributed object services. *In the Journal of Science of Computer Programming*, June 1998.

[4] Austin D. and Lunawat V. etc. *Using Oracle 8*. QUE, A Division of Macmillan Computer Publishing, USA, 2000.

[5] Barkley J. F. Application engineering in health care. In *Second Annual CHIN*. http://hissa.ncsl.nist.gov/rbac/proj/paper/paper.html, 1995.

[6] Barkley J. F., Beznosov K. and Uppal J. Supporting relationships in access control using role based access control. In *Third ACM Workshop on RoleBased Access Control*, pages 55–65, October 1999.

[7] Beam C. and Segev A. Electronic Catalogs and Negotiations. CITM Working Paper 96-WP-1016, August 1996.

[8] Bellare M., Canetti R., and Krawczyk H. Pseudorandom functions revisited: The cascade construction and its concrete security. extended abstract. In *37th Annual Symposium on the Foundations of Computer Science*. IEEE, 1996.

[9] Bellare M., Goldreich O., and Krawczyk H. Stateless evaluation of pseudorandom functions: Security beyond the birthday barrier. In *Advances in Cryptology - Crypto 99*, volume 1666 of *Lectures Notes in Computer Science*. Springer-Verlag, 1999.

[10] Bellare M., Roga way P. Random oracles are practical: a paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. IEEE, 1993.

[11] Ben-Shaul I., Gidron Y. and Holder O., editor. *A Negotiation Model for Dynamic Composition of Distributed Applications*. Institute of Electrical and Electronics Engineers, Inc., 1998.

[12] Bertino E., Castano S., Ferrari E. and Mesiti M. Specifying and enforcing access control policies for XML document sources. *World Wide Web, 3*, pages 139–151, 2000.

[13] Boris K. and Jajodia S. Concurrency control in multilevel-secure databases based on replicated architecture. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 153–162. ACM Press, 1990.

[14] Boyko V., Peinado M., and Venkatesan R. Speeding up discrete log and factoring based schemes via precomputations. In *Advances in Cryptology - Eurocrypt'98*, volume 1807 of *Lectures Notes in Computer Science*. Springer-Verlag, 1998.

[15] Bradford C. Legislating market winners, digital signature laws and the electronic commerce marketplace. *World Wide Web Journal*, 1997.

[16] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16(10):613–615, 1973.

[17] Buttyan L. and Hubaux J. Accountable anonymous access to services in mobile communication systems. In *Symposium on Reliable Distributed Systems*, pages 384–389. citeseer.nj.nec.com/article/buttyan99accountable.html, 1999.

[18] Canetti R., Goldreich O., and Halevi S. The random oracle methodology. In *Proceedings of the 30th ACM STOC '98*, pages 209–218. IEEE, 1998.

[19] Canetti R., Micciancio D., and Reingold O. Perfectly One-Way Probabilistic Hash Functions. In *Proceedings of the 30th ACM STOC '98*. IEEE, 1998.

[20] Chan A., Frankel Y., and Tsiounis Y. An efficient off-line electronic cash scheme as secure as RSA. Research report nu-ccs-96-03, Northeastern University, Boston, Massachussets, 1995.

[21] Charles G. Limoges and Ruth R. Nelson and John H. Heimann and David S. Becker. Versatile integrity and security environment (vise) for computer systems. In *Proceedings of the 1994 workshop on New security paradigms*, pages 109–118. IEEE Computer Society Press, 1994.

[22] Chaum D. Blind signature for untraceable payments. In *Advances in Cryptology - Crypto 82*, pages 199–203. Plenum Press N.Y., 1983.

[23] Chaum D., editor. *An introduction to e–cash*. DigiCash, http://www.digicash.com, 1995.

[24] Chaum D. *An Introduction to e-cash.* DigiCash, http://www.digicash.com, 1995.

[25] Chaum D. and Van Antwerpen H. Undeniable signatures. In *Advances in Cryptology–Crypto89*, volume 435 of *Lectures Notes in Computer Science*, pages 212–216. Springer-Verlag, 1990.

[26] Chaum D., Fiat A., and Naor M. Untraceable electronic cash. In *Advances in Cryptology - Crypto 88*, volume 403 of *Lectures Notes in Computer Science*, pages 319–327. Springer-Verlag, 1990.

[27] Chii-Hwa Lee and Min-Shiang Hwang and Wei-Pang Yang. Enhanced privacy and authentication for the global system for mobile communications. *Wireless Networks*, 5(4):231–243, 1999.

[28] China-Investigation Company. Necessity of insurance claim investigation. In *http://www.china-investigation.com/english/doc11.htm*, 2000.

[29] Clarke R. Key issues in electronic commerce and electronic publishing. In *In Proc. Information Online and On Disc 99*, Sydney, January, 1999.

[30] Cox B., Tygar J.D., Sirbu M. Netbill security and transaction protocol. In *The first USENIX Workshop on Electronic Commerce*, New York, 1995.

[31] Chaum D. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[32] David F. F., Dennis M. G. and Nickilyn L. An examination of federal and commercial access control policy needs. In *NIST NCSC National Computer Security Conference*, pages 107–116. Baltimore, MD, September 1993.

[33] David F. F., Riva S., Serban G., Kuhn D. and Ramaswamy C. Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[34] Dogac A. Survey of the Current State-of-the-Art in Electronic Commerce and Research Issues in Enabling Technologies. In *Proceeding of uro-Med Net 98 Conference, Electronic Commerce Track*, March 1998.

[35] EIGamal T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, IT-31(4):469–472, 1985.

[36] Eng T., Okamoto T. Single-trem divisible electronic coins. In *Advances in cryptology–Eurocrypt'94*, volume 950 of *Lectures Notes in Computer Science*, pages 306–319. Springer-Verlag, 1995.

[37] Eric O. Securing m-commerce. *ebizQ.net*, 2000.

[38] Feinstein H. L. Final report: Nist small business innovative research (sbir) grant: role based access control: phase 1. technical report. In *SETA Corp.*, 1995.

[39] Ferraiolo D. F. and Kuhn D. R. Role based access control. In *15th National Computer Security Conference*, pages 554–563. http://www.citeseer.nj.nec.com/ferraiolo92rolebased.html, 1992.

[40] Ferraiolo D. F., Barkley J. F. and Kuhn D. R. Role-based access control model and reference implementation within a corporate intranet. In *TISSEC*, volume 2, pages 34–64, 1999.

[41] Ford W. and Baum M. S. *Secure electronic commerce: Building the Infrastructure for Digital Signatures & Encryption.* Prentice Hall PTR, 1997.

[42] Frankel Y., Herzberg A., Karger P., Krawczyk H., Kunzinger C. and Yung M. Security issues in a cdpd wireless network. In *IEEE personal communications*, August 1995.

[43] Frankel Y., Yiannis T., and Yung M. Indirect discourse proofs: achieving fair off-line electronic cash. In *Advances in cryptology–Asiacrypt'96*, volume 1163 of *Lectures Notes in Computer Science*, pages 286–300. Springer-Verlag, 1996.

[44] Franklin M., Yung M. Secure and efficient off-line digital money. In *Proceedings of the Twentieth International Colloquium on Automata, Languages and Programming*, volume 700 of *Lectures Notes in Computer Science*, pages 265–276. Springer-Verlag, 1993.

[45] Gabber E. and Silberschatz A. Agora: A minimal distributed protocol for electronic commerce. In *The 2rd USENIX workshop on electronic commerce*, Oakland, CA, 1996.

[46] Garfinkel S. and Spafford G. *Web Security & Commerce Risks, Technologies, and Strategies.* O'Reilly & Associates, Inc., 1997.

[47] Goldreich O., Krawczyk H. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing*, 25(1):159–192, February 1996.

[48] Goldschlag D., Reed M., and Syverson P. Onion routing for anonymous and private Internet connections. *Communications of the ACM*, 24(2):39–41, 1999.

[49] Green S., Hurst L., Nangle B., Cunningham P., Somers F.and Evans R. Software Agents: A review. Tcd-cs-1997-06, Trinity College Dublin and Broadcom Eireann Research Ltd., Ireland, May 1997.

[50] Guttman R.H. and Maes P. Cooperative vs. Competitive Multi-Agent Negotiations in Retail Electronic Commerce. In *Proceedings of the Second International Workshop on Cooperative information Agents (CIA'98).*, Paris, France, July 1998.

[51] Herzberg A. and Yochai H. *Mini-Pay: Charging per Click on the Web.* http://www.ibm.net.il, 1996.

[52] Horn G. and Preneel B. Authentication and payment in future mobile systems. In *ES-ORICS*, 1998.

[53] Housley R., Ford W., Polk W. and Solo D. Internet x.509 public key infrastructure certificate and crl profile. In *Internet X.509 Public Key Infrastructure Certificate and CRL Profile*, http://www.ietf.org/rfc/rfc2459.txt., 1999.

[54] James J., Elisa B. and Arif G. Temporal hierarchies and inheritance semantics for gtrbac. In *Seventh ACM Symposium on Access Control Models and Technologies*, pages 74–83. ACM Press, 2002.

[55] Jansen W., etc. Security policy management for handheld devices. In *Proceedings of the 2003 International Conference on Security and Management (SAM'03)*, 2003.

[56] Jeffrey F., Bernard J. and Jeffrey R. *E-Commerce.* McGraw-Hill/Irwin, 2000.

[57] Juels A., Luby M. and Ostrovsky R. Security of blind digital signatures. In *Advances in Cryptology - Crypto 97*, volume 1294 of *Lectures Notes in Computer Science*, pages 150–164. Springer-Verlag, 1997.

[58] Kalakota, R. and Whinston A. *Frontiers of electronic commerce.* Addison-Wesley, 1996.

[59] Ketchpel S.P. and Garcia-Molina H. Making Trust Explicit in Distributed Commerce Transactions. In *IEEE Proceedings of the 16th ICDCS*, pages 270–281, 1996.

[60] Loudon D.L. and Della Bitta A. J. *CONSUMER BEHAVIOR: Concepts and Applications Fourth Edition.* McGRAW-HILL, Inc., 1993.

[61] Lubinski A. and Heuer A. Configured replication for mobile applications. In *Rostocker informatik berichte*, volume 24, pages 101–112, 2000.

[62] Lupu E., Marriott D., Sloman M. and Yialelis N. A policy based role framework for access control. In *ACM/NIST Workshop on Role-Based Access Contro.* http://www-dse.doc.ic.ac.uk/ ecl1/papers/rbac95/rbac95.pdf, 1995.

[63] Klusch M., editor. *Intelligent Information Agents: Agent-Based Information Discovery and Management on the Internet.* Springer, 1998.

[64] Malloy A., Varshney U. and Snow A. Supporting mobile commerce applications using dependable wireless networks. *Mobile Networks and Applications*, 7(3):225–234, 2002.

[65] Marilyn Greenstein and Todd M Feinman. *Electronic Commerce: Security, Risk Management and Control.* McGraw-Hill, MA, USA, 1999.

[66] Martin K., Preneel B., Mitchell C., Hitz H., Poliakova A., and Howard P. Secure billing for mobile information services in umts. In *IS-N*, 1998.

[67] MastercardVisa, editor. *SET 1.0 - Secure electronic transaction specification.* http://www.mastercard.com/set.html, 1997.

[68] Mehrotra A. GSM system engineering. Norwood, Artech House, 1997.

[69] Mehrotra A. and Golding L. Mobility and security management in the GSM system and some proposed future improvements. In *Proceedings of IEEE*, volume 86(7), 1998.

[70] Michael D. Schroeder and Jerome H. Saltzer. A hardware architecture for implementing protection rings. *Communications of the ACM*, 15(3):157–170, 1972.

[71] Michael S. and Achim K. Mobile commerce for financial services—killer applications or dead end? *ACM SIGGROUP Bulletin*, 22(1):22–25, 2001.

[72] Najam Perwaiz. Structured management of role-permission relationships. In *Proceedings of the Sixth ACM Symposium on Access control models and technologies*, pages 163–169. ACM Press, 2001.

[73] Oh S. and Sandhu R. A model for role administration using organization structure. In *Seventh ACM Symposium on Access Control Models and Technologies*, pages 155–162. ACM Press, 2002.

[74] Okamoto T. An efficient divisible electronic cash scheme. In *Advances in Cryptology–Crypto'95*, volume 963 of *Lectures Notes in Computer Science*, pages 438–451. Springer-Verlag, 1995.

[75] Okamoto T., Ohta K. Disposable zero-knowledge authentication and their applications to untraceable electronic cash. In *Advances in Cryptology–Crypto89*, volume 435 of *Lectures Notes in Computer Science*, pages 481–496. Springer-Verlag, 1990.

[76] Osborn S. L., Reid L. K. and Wesson G. J. On the Interaction Between Role-Based Access Control and Relational Databases. In *IFIP WG11.3 Tenth Annual Working Conference on Database Security*, pages 139–151, July 1996.

[77] Papazoglou M. and ATsalgatidou A. Special Issue on Information Systems Support for Electronic Commerce. *Information Systems*, 24(6), 1999.

[78] Pfitzmann B., Waidner M. How to break and repair a 'provably secure' untraceable payment system. In *Advances in Cryptology - Crypto'91*, volume 576 of *Lectures Notes in Computer Science*, pages 338–350. Springer-Verlag, 1992.

[79] Pointcheval D. Self-scrambling anonymizers. In *Proceedings of Financial Cryptography*, Anguilla, British West Indies, 2000. Springer-Verlag.

[80] Pointcheval D. and Stern J. Security arguments for digital signatures and blind signatures . *Journal of cryptology*, 13(3):361–396, 2000.

[81] Poutanen T., Hinton H. and Stumm M. Netcents: A lightweight protocol for secure micropayments. In *The 3rd USENIX Workshop on Electronic Commerce*, Boston, Massachusetts, August, 1998.

[82] Pratel B. and Crowcroft J. Ticket based service access for the mobile user. In *In Proceedings of MobiCom: International Conference on Mobile Computing and Networking*, pages 223–232, Budapest, Hungary, 1997.

[83] Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors. *Fair Electronic Cash Based on a Group Signature Scheme*, volume 2229 of *Lecture Notes in Computer Science*. Springer, 2001.

[84] Rabin M. *Digital Signatures, Foundations of secure communication.* New York: Academic Press, NewYork, 1978.

[85] Ralf Neubert, Oliver langer, Otmar Gorlitz, Wolfgang Benn. Virtual enterprises – challenges from a database persperctive. In *Proceedings of ADC'01*, GoldCoast, Australia, 2001. IEEE.

[86] Rivest R. L., Shamir A., and Adleman L. M. A method for obtaining digital signatures and public-Key cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[87] Rivest R. T. The MD5 message digest algorithm. *Internet RFC 1321*, April 1992.

[88] Rohm A.W. and Pernul G. COPS: A Model and Infrastructure for Secure and Fair Electronic Markets. In IEEE Computer Society Press, editor, *Proc. 32nd Hawaii International Conference on System Sciences (HICSS-32)*, Hawaii, January 1999.

[89] Sandhu R. Role activation hierarchies. In *Third ACM Workshop on RoleBased Access Control*, October 1998.

[90] Sandhu R. Role-Based Access Control. *Advances in Computers*, 46, 1998.

[91] Sandhu R. Future directions in role-based access control models. In *MMS, 2001.* http://www.list.gmu.edu/confrnc/misconf/, 2001.

[92] Sandhu R. and Bhamidipati V. An oracle implementation of the pra97 model for permission-role assignment. In *ACM Workshop on Role-Based Access Control*, pages 13–21. http://citeseer.nj.nec.com/27106.html, 1998.

[93] Sandhu R. and Bhamidipati V. The ura97 model for role-based administration of user-role assignment. *T. Y. Lin and Xiao Qian, editors, Database Security XI: Status and Prospects*, North-Holland, 1997.

[94] Sandhu R. and Munawer Q. The arbac99 model for administration of roles. In *the Annual Computer Security Applications Conference*, pages 229–238. ACM Press, 1999.

[95] Sandhu R. and Park J. S. Decentralized User-Role Assignment for Web-based Intranets. In *3th ACM Workshop on Role-Based Access Control*. Fairfax, Virginia, October 1998.

[96] Schnorr C. P. Efficient signature generation by smart cards. *Journal of cryptology*, 4(3):161–174, 1991.

[97] Scourias, J. An overview of the Global System for Mobile communications. Technical report, niversity of Waterloo, Canada, May 1995.

[98] Simon D. Anonymous communication and anonymous cash. In *Advances in Cryptology - Crypto'96*, volume 1109 of *Lectures Notes in Computer Science*, pages 61–73. Springer-Verlag, 1997.

[99] Solms S.von and Naccache D. On blind signatures and perfect crimes. *Computers and Security*, 11:581–583, 1992.

[100] Spegel N. M., Rogers B. and Buckley R. P. *Negotiation Theory and Techniques*. Skills Series. Butterworths, 1998.

[101] Steven B. Lipner. A comment on the confinement problem. In *Proceedings of the fifth symposium on Operating systems principles*, pages 192–196, 1975.

[102] Stinson D. R. *Cryptography: Theory and practice.* CRC Press, Boca Raton, 1995.

[103] Terry R. Application level security using an object-oriented graphical user interface. In *Proceedings on the 1992-1993 workshop on New security paradigms*, pages 105–108. ACM Press, 1993.

[104] Timmers P. Global and Local in Electronic Commerce. In *Proceedings of EC-Web*, volume 1875 of *Lectures Notes in Computer Science*, London, 2000. Springer-Verlag.

[105] Varshney U. and Vetter R. Mobile commerce: framework, applications and networking support. *Mobile Networks and Applications*, 7(3):185–198, 2002.

[106] Waleffe D. d. and Quisquater J. J. Better login protocols for computer networks. In *ESORICS'90*, pages 163 – 172, October, 1990.

[107] Wang H. and Duan T. A signature scheme for security of e-commerce. *Computer Engineering*, 25:79–80, 1999.

[108] Wang H. and Zhang Y . A protocol for untraceable electronic cash. In Hongjun Lu and Aoying Zhou, editor, *Proceedings of the First International Conference on Web-Age Information Management*, volume 1846 of *Lectures Notes in Computer Science*, pages 189–197, Shanghai, China, 2000. Springer-Verlag.

[109] Wang H. and Zhang Y. Untraceable off-line electronic cash flow in e-commerce. In *Proceedings of the 24th Australian Computer Science Conference*

ACSC2001, pages 191–198, GoldCoast, Australia, 2001. IEEE computer society.

[110] Wang H., Cao J. and Kambayashi Y. Building a consumer anonymity scalable payment protocol for the internet purchases. In *12th International Workshop on Research Issues on Data Engineering: Engineering E-Commerce/E-Business Systems*, San Jose, USA, Feb. 25-26 2002.

[111] Wang H., Cao J. and Zhang Y. A consumer anonymity scalable payment scheme with role based access control. In *2nd International Conference on Web Information Systems Engineering (WISE01)*, pages 53–62, Kyoto, Japan, December 2001.

[112] Wang H., Cao J., and Zhang Y. A Flexible Payment Scheme and Its Role-based User-role Assignment. In *Proceedings of the second International Workshop on Cooperative Internet Computing (CIC2002)*, pages 58–68, Hong Kong, China, 2002.

[113] Wang H., Cao J., and Zhang Y. A flexible payment scheme and its user-role assignment. In *Alvin Chan, etc. Editor, Cooperative Internet Computing*, pages 107–128. Kluwer Academic Publisher, 2002.

[114] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for role-based access control based on relational algebra operations. In *3nd International Conference on Web Information Systems Engineering (WISE02)*, pages 301–312, Singapore, December 2002.

[115] Wang H., Cao J. and Zhang Y. Ticket-based service access scheme for mobile users. In *Twenty-Fifth Australasian Computer Science Conference*

(ACSC2002), Monash University, Melbourne, Victoria, Australia, Jan. 28 - Feb. 02 2002.

[116] Wang H., Cao J., and Zhang Y. A flexible payment scheme and its permission-role assignment. In *Proceedings of the Twenty-Sixth Australasian Computer Science Conference (ACSC2003)*, pages 189–198, Adelaide, Australia, 2003.

[117] Wang H., Cao J., and Zhang Y. An Electronic Payment Scheme and Its RBAC management. *Concurrent Engineering: Research and Application*, To appear, 2003.

[118] Wang H., Cao J. and Zhang Y. Formal authorization allocation approaches for permission-role assignments using relational algebra operations. In *Proceedings of the 14th Australian Database Conference ADC2003*, Adelaide, Australia, 2003.

[119] Wang H., Zhang Y., Cao J., Kambayahsi Y. A global ticket-based access scheme for mobile users. *Special Issue on Object-Oriented Client/Server Internet Environments, Information Systems Frontiers*, 5(3), 2003.

[120] Wang H., Zhang Y., Cao J., Varadharajan V. . Achieving secure and flexible m-services through tickets. *IEEE Transactions on Systems, Man, and Cybernetics, Special issue on M-Services*, 2003.

[121] Wilhelm U. Staamann S. and Buttyan L. On the problem of trust in mobile agent systems. In *IEEE network and distributed systems security symposium*, pages 11–13, San Diego, CA, USA, 1998.

[122] Wu R. Building a legal framework for e-commerce in hong kong. *Journal of Information Law and Technology*, 2000.

[123] Zhang Y. and Jia X. Transaction processing. *Wiley Encyclopedia of Electrical and Electronics Engineering*, 22:298–311, 1999.

[124] Yacobi Y. Efficient electronic money. In *Advances in Cryptology–Asiacrypt'94*, volume 917 of *Lectures Notes in Computer Science*, pages 153–163. Springer-Verlag, 1995.

[125] Yiannis T. Fair off-line cash made easy. In *Advances in Cryptology–Asiacrypt'98*, volume 1346 of *Lectures Notes in Computer Science*, pages 240–252. Springer-Verlag, 1998.

[126] Yiannis T. and Yung M. On the security of ElGamal-based encryption. In *International Workshop on Practice and Theory in Public Key Cryptography (PKC '98)*, volume 1346 of *Lectures Notes in Computer Science*, Yokohama, Japan, 1998. Springer-Verlag.

[127] Zhaohui Chen, Matthew K. O. Lee, Christy Cheung. A framework for mobile commerce. In *In Proc. Americas Conference on Information Systems 2001, E-Commerce: Wireless/Mobile*. AISeL, 2001.