Privacy-Aware Authentication and Authorization in Trust Management

by
Danfeng Yao
B. S., Peking University, 1998
M. A., Princeton University, 2001
M. Sc., Indiana University, Bloomington, 2002

A dissertation submitted in partial fulfillment of the
requirements for the Degree of Doctor of Philosophy
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2008

This dissertation by Danfeng Yao is accepted in its present form by
the Department of Computer Science as satisfying the dissertation requirement
for the degree of Doctor of Philosophy.

Date _____ _____

Roberto Tamassia, Director

Recommended to the Graduate Council

Date _____ _____

Anna Lysyanskaya, Reader

Date _____ _____

Claire Mathieu, Reader

Approved by the Graduate Council

Date _____ _____

Sheila Bonde
Dean of the Graduate School

Abstract of "Privacy-Aware Authentication and Authorization in Trust Management" by Danfeng Yao, Ph.D., Brown University, May 2008.

Conventional access decisions in stand-alone systems are usually made based on the identity of the entity requesting a resource. By comparison, in open systems such as the Internet, computing grids, and mobile and ad hoc networks, this approach becomes less effective. The main reason is that there is no central authority that has global knowledge of all users and makes access decisions. Instead, the resource owner and the requester typically belong to different security domains administrated by different authorities and are unknown to each other. Trust management refers to access control frameworks that facilitate assured information sharing in open systems. In this thesis, we study authentication and authorization problems in various trust management scenarios, focusing on the requirements of security, efficiency, scalability, and privacy protection. We develop trust management models and cryptographic protocols that satisfy the above requirements without the need of a central authority. Specifically, we present solutions for administrator-free role-based dynamic collaborations, compact and anonymous authorization chains, on-line identity protection, and privacy-preserving recommendation system.

Vita

DANFENG (DAPHNE) YAO

Curriculum Vitae

Division of Computer and Information Sciences (401)222-9165 (Mobile)
Rutgers, the State University of New Jersey danfeng@cs.rutgers.edu
Piscataway, NJ 08854-8019 Homepage: http://www.cs.rutgers.edu/~danfeng/

RESEARCH INTERESTS

Computer and information security, applied cryptography, security middleware

EDUCATION

Ph.D., Computer Science, **Brown University**, Providence, RI Nov. 2007 (Expected)

M.S., Computer Science, **Indiana University**, Bloomington, IN May 2002

M.A., Chemistry, **Princeton University**, Princeton, NJ Nov. 2000

B.S., Chemistry, **Peking University**, Beijing, China Jul. 1998

EMPLOYMENT

Department of Computer Science, Rutgers University, New Brunswick, NJ Jan. 2008 –
Assistant Professor

HP Trust Systems Labs, Princeton, NJ May 2006 – Sep. 2006
Research intern (With Dr. Stuart Haber)

Department of Computer Science, Brown University Aug. 2002 – Present
Research assistant (With Professor Roberto Tamassia)

IAM Technology Inc., Providence, RI Apr. 2005 – Present
Researcher (With David Croston, CEO)

CERIAS, Purdue University, West Lafayette IN Summer 2005

Visiting research assistant (With Professor Elisa Bertino and Mikhail J. Atallah)

Center of Genomics and Bioinformatics, Indiana University, Bloomington May 2001 - Aug. 2002

Research assistant (With Dr. Donald Gilbert)

Department of Chemistry, Princeton University May 1999 - Dec. 2000

Research assistant (With Professor Daniel Kahne)

TEACHING EXPERIENCE

Computer Science Department, Brown University Spring 2004, 2006

Graduate teaching assistant for CS166 (Introduction to Computer Security)

Graduate teaching assistant for CS016 (Algorithms and Data Structures)

Computer Science Department, Indiana University, Bloomington Spring 2001

Graduate teaching assistant for A111 (Survey of Computers & Computing)

Princeton University, Chemistry Department Fall 1998, Spring 1999

Graduate teaching assistant for CHEM 371 and 372 (Experimental Chemistry I, II)

HONORS

Best Student Paper Award, ICICS 2006 Dec. 2006

Award for Technological Innovation from Brown University Apr. 2006

University Fellowship, Brown University Sep. 2002

Graduate with the Highest Honors, Peking University Jul. 1998 IEC Fellowship, Peking University Sep. 1996

Outstanding Student Fellowship, Peking University Sep. 1995

SONY Fellowship, Peking University Sep. 1995 Student Travel Grants from SIGAPP, ICICS, VizSEC, CCS, Brown University

JOURNALS

1. Yunhua Koglin, Danfeng Yao, and Elisa Bertino. Secure Content Distribution by Parallel Processing from Cooperative Intermediaries. To appear in *IEEE Transactions on Parallel and Distributed Systems.* July. 2007.

2. Danfeng Yao and Roberto Tamassia. Compact and Anonymous Role-Based Authorization Chain. *ACM Transactions on Information and System Security (TISSEC).* Under revision. Jun. 2007.

PEER-REVIEWED CONFERENCES/WORKSHOPS

1. Danfeng Yao, Roberto Tamassia, and Seth Proctor. Private Distributed Scalar Product Protocol With Application To Privacy-Preserving Computation of Trust. In *Proceedings of IFIPTM 2007 – Joint iTrust and PST Conferences on Privacy, Trust Management and Security.* Moncton, New Brunswick, Canada. Jul. 2007.

2. Isabel F. Cruz, Roberto Tamassia, and Danfeng Yao. Privacy-Preserving Schema Matching Using Mutual Information. Extended Abstract. In *Proceedings of the 21th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec '07).* Redondo Beach, CA. Jul. 2007.

3. Danfeng Yao, Yunhua Koglin, Elisa Bertino, and Roberto Tamassia. Decentralized Authorization and Data Security in Web Content Delivery. In *Proceedings of the 22nd ACM Symposium on Applied Computing (SAC '07)*, Special Track on Web Technologies. ACM Press. Seoul, Korea. Mar. 2007.

4. Danfeng Yao, Keith B. Frikken, Mikhail J. Atallah, and Roberto Tamassia. Point-Based Trust: Define How Much Privacy Is Worth. In *Proceedings of the Eighth International Conference on Information and Communications Security (ICICS '06).* LNCS 4307, pages 190-209. Springer. Raleigh, NC. Dec. 2006. **Best Student Paper Award**.

5. Danfeng Yao and Roberto Tamassia. Cascaded Authorization with Anonymous-Signer Aggregate Signatures. In *Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '06).* West Point, NY. Jun. 2006.

6. Michael T. Goodrich, Roberto Tamassia, and Danfeng Yao. Notarized Federated Identity Management for Increased Trust in Web Services. In *Proceedings of the 20th Annual IFIP WG 11.3 Working Conference on Data and Applications Security (DBSec '06).* LNCS 4127, pages 133-147. Springer. Sophia Antipolis, France. Jul. 2006.

7. Danfeng Yao, Michael Shin, Roberto Tamassia, and William H. Winsborough. Visualization of Automated Trust Negotiation. In *Proceedings of the Workshop on Visualization for Computer Security (VizSEC '05) in Conjunction with Vis 2005 and InfoVis 2005.* Pages 65-74. IEEE Press. Minneapolis, MN. Oct. 2005.

8. Danfeng Yao, Roberto Tamassia, and Seth Proctor. On Improving the Performance of Role-Based Cascaded Delegation in Ubiquitous Computing. In *Proceedings of the IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)*. Pages 157-168. IEEE Press. Athens, Greece. Sep. 2005.

9. Michael T. Goodrich, Roberto Tamassia, and Danfeng Yao. Accredited DomainKeys: A Service Architecture for Improved Email Validation. In *Proceedings of the Second Conference on Email and Anti-Spam (CEAS '05)*. Stanford University, CA. Jul. 2005.

10. Danfeng Yao, Nelly Fazio, Yevgeniy Dodis, and Anna Lysyanskaya. ID-Based Encryption for Complex Hierarchies with Applications to Forward Security and Broadcast Encryption. In *Proceedings of the 11th ACM Conference on Computer and Communications Security (CCS '04)*. Pages 354-363. ACM Press. Washington, DC, Oct. 2004.

11. Roberto Tamassia, Danfeng Yao, and William H. Winsborough. Role-Based Cascaded Delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*. Pages 146-155. ACM Press. Yorktown Heights, NY, Jun. 2004.

PAPERS IN SUBMISSION

1. Michael T. Goodrich, Roberto Tamassia, and Danfeng Yao. Notarized Federated Identity Management for Increased Trust in Web Services. *Journal of Computer Security*, special issue of selected papers of *DBSec 2006*. *Submitted*. Apr. 2007.

2. Stuart Haber, William G. Horne, Tomas Sander, and Danfeng Yao. Audit-Log Integrity Using Redactable Signatures With Pseudonyms. *Submitted*. Oct. 2007.

3. Roberto Tamassia, Danfeng Yao, and William H. Winsborough. Independently Verifiable Administrator-Free Delegation. *Journal of Computer Security (JCS). Submitted*. Jun. 2006.

4. Danfeng Yao, Keith B. Frikken, Mikhail J. Atallah, and Roberto Tamassia. Private Information: To Reveal Or Not To Reveal. *ACM Transactions on Information and System Security (TISSEC). Submitted*. Sep. 2007.

5. Distributed Scalar Product Protocol With Application To Privacy-Preserving Computation of Trust. *IEEE Transactions on Dependable and Secure Computing (TDSC). Submitted*. Sep. 2007.

INVITED TALKS

1. Verification of Integrity for Outsourced Content Publishing and Database Queries. *Purdue University CERIAS Seminar*, West Lafayette, IN. Oct. 2006.

2. Trust Management and Private Communication in Role-Based Systems. *Purdue University Computer Science Departmental Seminar*, West Lafayette, IN. Nov. 2005.

3. Trust and Service Negotiations Using WSPL. Sun Microsystems Lab, Burlington MA. Nov. 2003.

PROFESSIONAL ACTIVITIES

Program Committee member for *23rd ACM Symposium on Applied Computing*, *IEEE. Information Assurance Workshop '07*, *6th International Workshop on Privacy Aspects of Data Mining*
Reviewer for *IEEE Symposium on Security and Privacy '07*, *ICDE '07*, *Journal of Computer Security*, *International Workshop on Practice and Theory in Public Key Cryptography (PKC '05)*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Vehicular Technology*.
IEEE member, ACM member.

PATENT

1. Integrity Verification of Pseudonymized Documents. Stuart Haber, William G. Horne, Tomas Sander, and Danfeng Yao. U.S. Patent pending. Sep. 2007.

2. Notarized Federated Identity Management for Web Services. Michael T. Goodrich, Roberto Tamassia, and Danfeng Yao. U.S. patent pending 60/833,983. Jul. 28, 2006.

3. Domain and Link Authentication with *iDomainGuard*. Michael T. Goodrich, Roberto Tamassia, and Danfeng Yao. U.S. patent pending 60/854,589. Oct. 26, 2006.

Acknowledgements

I would like to thank my Ph.D. advisor Professor Roberto Tamassia for his patient guidance and training. He taught me how to be a computer scientist, inspired me to tackle complex problems, and showed me how to work with different people. He has also given me much valuable advice and considerate support for my personal life and my future career, helping me have a balanced work and family life. Most importantly, Roberto taught me to keep an open mind to be willing to tackle a broad spectrum of research problems. Another person who influenced me most during my Ph.D. is Professor Anna Lysyanskaya. She taught me the fundamentals of modern cryptography, which is the foundation of my thesis work. Anna also showed me how to write an interesting research paper and give a memorable presentation. I also want to thank Professor Claire Mathieu for her great advice on my thesis work. I enjoyed every lecture of her Approximate Algorithms class. Claire taught me how to give a clear talk about complex theoretical concepts. Anna and Claire have been important female role models for me during my study at Brown.

I would also like to thank Professors Michael T. Goodrich, Elisa Bertino, Mikhail J. Atallah, Keith B. Frikken, William H. Winsborough, Dr. Yunhua Koglin, and Mr. Seth Proctor for working with me on some security problems. I learned tremendously from our collaboration. I would like to thank Professors Tom Doeppner, Ugur Cetintemel, Stan Zdonik, Shriram Krishnamurthi, Meinolf Sellmann, Eli Upfal, John Jannotti, Philip Klein, Andy van Dam, Franco P. Preparata, Mr. David Croston, Dr. John Nuber, and Mr. Steven T. Carmody for their inspiring discussions and valuable comments on my thesis work. I also want to thank my fellow graduate students Aris Anagnostopoulos, Nikos Triandopoulos, Charalampos (Babis) Papamanthou, Lijuan Cai, Ye Sun, Ying Xing, Song Zhang, Glencora L. Borradaile, Tijian Ge, Wenjin Zhou, Olga A. Karpenko, Yanif Ahmad, Olga Papaemmanouil, Mira Belenkiy, Melissa

Chase, and Matt Lease for making my graduate school life enjoyable and memorable. I would like to give special thanks to my husband, Chang Lu, for his warm support for my relentless pursuit of a research career. Last but not least, I want to thank my parents, Guosheng Yao and Yuzhu Sun, for their patience with me during my graduate studies.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

In many applications, sharing of local resources and information across different domains greatly benefits participating parties. The resource owner needs to be assured that unauthorized users cannot access the shared information. However, traditional access control models require a central party that knows every user and their privileges, which is not feasible in open systems such as Internet and Grid computing. This thesis studies decentralized access control frameworks that facilitate assured information sharing in open systems. We develop models and protocols for the authentication and authorization in various trust management scenarios, focusing on security, efficiency, scalability, and privacy protection. The application domains studied include resource sharing, e-commerce, recommendation systems, and identity management. In these domains, we demonstrate simple and secure trust management models and protocols without the need of a central authority. Our authorization models allow secure and efficient sharing of information by parties across different administrative domains. Compared to existing trust management frameworks, our models support more flexible and scalable trust establishment mechanism between entities that are unknown to each other. Our authentication protocols that prevent unauthorized entities from accessing the shared information are shown to have provable security and privacy properties. These features facilitate dynamic and assured resource sharing in open systems.

Next, we summarize the trust management problems studied in this thesis and briefly describe our contributions. Details of the work are presented in later chapters.

In Chapter 2, we study the verification problem of role-based delegation. In decentralized trust management, a delegation chain connects trusted entities by the resource owner with unknown users. We give the first role-based trust management model that supports independently-verifiable delegation, where a role member is able to delegate on behalf of a role and the delegation credential can be independently verified by any third-party without the participation of the administrator [114]. Our model, called role-based cascaded delegation, supports the delegation of authority in role-based trust management systems. We also describe an efficient realization of role-based cascaded delegation using aggregate signatures, where the authentication information for an arbitrarily long role-based delegation chain is captured by one short signature of constant size.

In ubiquitous computing environments, computing devices may have small storage units and limited bandwidths. A trust management system needs to be efficient in order to keep communication and computation costs low. The trust establishment mechanism needs to be flexible, because credentials are usually scattered at *distributed* locations. Also, the authorization process needs to be decentralized and support dynamic resource-sharing in order to handle emergency situations. We discuss how to improve the efficiency, flexibility, and privacy of role-based cascaded delegations in a ubiquitous computing environment [129]. Operations for managing delegation chains in the role-based cascaded delegation (RBCD) model are presented. These operations can significantly improve the performance of the decentralized delegation in the RBCD model, without increasing the management overhead.

In Chapter 3, we study how to support efficient and anonymous role-based delegation. We introduce a decentralized trust management model called anonymous role-based cascaded delegation [128]. In this model, a delegator can issue authorizations on behalf of her role without revealing her identity. This type of delegation protects the sensitive membership information of a delegator and hides the internal structure of an organization. To provide an efficient storage and transmission mechanism for credentials used in anonymous role-based cascaded delegation, we present a new digital signature scheme that supports both signer anonymity and signature

aggregation. Our scheme has compact role signatures that make it especially suitable for ubiquitous computing environments, where users may have mobile computing devices with narrow communication bandwidth and small storage units.

In Chapter 4, we study how to protect private information in trust management. We have studied the notion of point-based policies for trust management, and gives protocols for realizing them in a disclosure-minimizing fashion [127]. Specifically, Bob values each credential with a certain number of points, and requires a minimum total threshold of points before granting Alice access to a resource. In turn, Alice values each of her credentials with a privacy score that indicates her reluctance to reveal that credential. Bob's valuation of credentials and his threshold are private. Alice's privacy-valuation of her credentials is also private. Alice wants to find a subset of her credentials that achieves Bob's required threshold for access, yet is of as small a value to her as possible. We give protocols for computing such a subset of Alice's credentials without revealing any of the two parties' above-mentioned private information.

In Chapter 5, we study how to protect sensitive trustworthiness data in a recommendation system. We first present a private distributed scalar product protocol that can be used for obtaining trust values from private recommendations [130]. Our protocol allows Alice to infer the trustworthiness of Bob based on what Alice's friends think about Bob and Alice's confidence in her friends. In addition, the private information of Alice and her friends are not revealed during the computation. We also propose a credential-based trust model where the trustworthiness of a user is computed based on his or her affiliations and role assignments. The trust model is simple to compute, yet it is scalable as it classifies large groups of users.

1.2 Summary of Contributions

The contributions of this thesis include

- Models and protocols for decentralized delegation for transferring privileges without a central administrator.
- Anonymity role-based delegation protocol and an anonymous-signer aggregate signature scheme.

- Privacy-preserving authorization model that supports personalized access policies for users.
- Role-based computational trust model and a privacy-preserving distributed scalar product protocol.

Chapter 2

Independently-Verifiable Decentralized Role-Based Delegation

A preliminary version of the paper was presented at the 2004 *ACM Symposium on Access Control Models and Technologies* [114].

2.1 Introduction

Trust management is an approach to access control in environments where entities that are not in the same security domain need to share resources. Several trust management systems have been proposed in recent years, e.g., PolicyMaker [18], KeyNote [17], SPKI/SDSI [43], and the *RT* framework [89]. The notion of delegation is essential in transferring trust and authorization in trust management systems. It facilitates information and resource sharing in distributed collaborative environment such as Grid computing and peer-to-peer networks. Delegation chains trace sequences of entities, starting from the resource owner and including entities authorized by (though possibly unknown to) the owner. These entities play a central part in authorization by providing the credentials that represent their own delegation acts, which enable the delegation chain to be verified.

In role-based delegation, delegated privileges are issued to a role rather than to

an individual. The abstraction of roles makes delegation scalable as one delegation benefits all members. Although the concept of role-based delegation is not new in access control and trust management literature, multi-step role-based delegation chain and its verification have not been much studied. Most prior work that addresses the problem of determining whether credentials prove an entity's resource request is authorized [17, 18, 43] assumes that all potentially relevant credentials are available in one central storage. However, this assumption is not valid in decentralized environment where there is no central authority. In fact, the verification cost may be quite expensive in a typical trust management system implementation. Collecting and verifying delegation credentials incurs communicational and computational costs, as does checking that together the credentials provide proof that a given user is authorized. We present techniques that can be used to significantly reduce these costs. Next, we illustrate in Example 2.1.1 a simple multi-step delegation scenario that transfers rights among roles within one administrative domain.

Example 2.1.1 *A hospital has roles* Doctor*,* Nurse*, and* Intern*. The hospital permits all doctors to access a medical storage room. Bob is a doctor and has a doctor role credential issued by the hospital. When Bob is out of town, he authorizes his nurses to access the storage room by issuing the nurses a delegation credential. Alice is Bob's nurse and has a nurse role credential. She has short-term interns who also need to access the storage room. Then Alice passes the access privilege onto her interns by creating another delegation credential. The two-step delegation chain gives the authorization to interns to access the storage room, which consists of the two delegation credentials and Bob and Alice's role credentials. The role credentials show the delegators have the proper roles to issue the delegation. When an intern, say Carl, requests for access, the delegation credentials and role credentials are verified. In addition, Carl's* Intern *role credential is also verified to ensure he is indeed an intern.*

Example 2.1.1 only involves one administrative domain, namely the hospital. Therefore, the credentials and the public keys of delegators (Bob and Alice) can be reasonably assumed to be available to the verifier that is the hospital server. However, trust management is to facilitate information sharing across different administrative domains. Delegation is usually decentralized and typically involves users and roles

from multiple organizations. The verification process is much more complex, because there is no single trusted authority and public keys for credentials may not be known or trusted. We show in Section 2.4 a more complex cross-domain role-based delegation. In this work, we study the role-based delegation in the general setting. We propose a *role-based cascaded delegation* protocol that supports assured information sharing in a decentralized fashion.

For efficient transmission and storage, compact digital credentials are desirable. Multi-step delegation credentials may be lengthy because the verification of a delegation chain requires checking a number of signatures linear in the length of the chain, where the length is defined as the number of delegations on the chain. Conventional signature schemes, such as RSA [100] and DSA [51], produce relatively long signatures compared to the security they provide. For a 1024-bit modulus (security level), RSA signatures are 1024 bits long and standard DSA signatures are 320 bits long. The number of signatures required to authenticate a role-based delegation chain of length n is about $2n$, because in addition to verifying each of the delegation transactions, one must verify that the intermediate delegators are members of the required roles. Among the signatures associated with a delegation chain, the signature on a role credential is generated by the administrator of that role independently from the rest of the signatures.

Unfortunately, it is not known how to aggregate individually generated signatures from different signers on different messages in conventional cryptosystems, such as RSA [24, 92]. This means that the entire set of signatures has to be stored by delegated entities, and transmitted across networks at each delegation and verification. Because intermediate delegators in our model may be entities who have limited computational power and communicational bandwidth, the implementation of role-based cascaded delegation using conventional credentials is inefficient. We overcome these problems by realizing the role-based cascaded delegation with short aggregate signatures [23, 26].

2.1.1 Our Contributions

Existing delegation models assume the delegation is issued by administrators. However, to enable flexible resource sharing, the decision of introducing new role members

into a collaboration needs to be dynamically made by members of existing roles, without the involvement of administrators. In the meantime, the shared information needs to be adequately protected against unauthorized or unqualified users. These goals drives us to reexamine conventional assumptions in role-based delegation, and define our model from a different perspective. We summarize our contributions next.

- We present a role-based delegation mechanism that supports the efficient verification of multi-step delegation chains. It has two main features: **(1)** flexible delegation where a delegation can be issued by a valid member of a role, not just by the administrator; and **(2)** simple verification where a delegation credential is self-contained and the verification does not require the participation of any role administrators. Our delegation mechanism takes a simple accumulation approach, where each intermediate delegator passes down the relevant digital credential to the delegated entity for later verification.

- We give a detailed protocol specification for public key signing and management that ensures the integrity of shared resources. The significance of our RBCD protocol is that we do not assume the existence of a public key infrastructure (PKI), which may be expensive to adopt widely. This feature makes our protocol general enough for many decentralized and open system environments such as peer-to-peer networks, where there are no central authorities and PKI is usually not available. Another important feature of our delegation protocol is that a delegation is issued to a role, yet we are able to support access accountability. That is, in case of misbehaving individuals, the resource owner can identify their identities, who are authorized through our role-based delegation.

- We also present a concrete realization of RBCD that gives compact delegation credentials. Traditionally, the number of signatures required for the verification of a delegation chain is linear in the number of entities of the chain. Our implementation needs only one aggregate signature, which is a significant improvement in efficiency over the existing delegation chain protocols.

Although our delegation model is role-based, it can be simplified to support individual delegation, i.e., a role member further extends his or her delegated privileges

to another individual. Because role-based delegation is more general and scalable, it is the focus of our presentation in this work.

2.1.2 Organization

We define the terminology and notations used in our role-based cascaded delegation in Section 2.2. The necessary cryptographic knowledge is also described. The overview of our role-based cascaded delegation mechanism is given in Section 2.3. In Section 2.4, an example of role-based cascaded delegation is presented. Our delegation protocol is described in Section 2.5. A realization of RBCD with aggregate signatures is presented in Section 2.6. In Section 2.7, we address the issues of revocation, security, scalability, and efficiency for our model and implementation. A comparison of role-based cascaded delegation with existing trust management approaches is given in Section 2.8. Section 2.9 is the conclusion.

2.2 Definitions and Preliminaries

In this section, we give our definitions for affiliated and delegated roles, and define our terminology and notations. We also give our definition for independently-verifiable decentralized role-based delegation. Finally we describe the necessary cryptographic knowledge.

2.2.1 Roles and their scopes

In our model, we define the *administrator* of a role as the organization that creates and manages the role. If a role credential of an entity D is signed and issued by the administrator of the role, that role is said to be an *affiliated role* of D. (This type of role is usually obtained through the affiliation with an organization, and thus the name.) If a role credential of D is instead issued through delegation and signed by entities other than the administrator of the role, that role is called a *delegated role* of D.

The following example illustrates the difference between affiliated and delegated roles. Bob is a full-time professor at University U. He has a credential signed by

university U for the role *professor* at U, denoted $U.professor$. Thus, role $U.professor$ is an affiliated role of Bob. Alice is not the university's employee, but she is Bob's collaborator. Bob delegates Alice the role $U.professor$ to allow her to access university's resources. However, Alice does not have a credential signed by U for $U.professor$. Thus, $U.professor$ is a delegated role of Alice.

The reason of making this distinction is to protect sensitive resources and to provide easy management for resource owners. An affiliated role and a delegated role have different access scopes. Delegations to a role r of an organization only apply to those entities who have r as an affiliated role. In the above example, if a privilege is delegated by a third-party to role $U.professor$, then Bob is entitled to this privilege, whereas Alice is not. This is because $U.professor$ is Alice's delegated role. The new privileges delegated to role $U.professor$ do not automatically propagate to her. A real-life analogy to this distinction is professor vs. visiting professor. In a university, a full-time professor is appointed by the university, whereas a visiting professor position is temporary and typically approved by a full-time professor. A visiting professor has fewer privileges than a full-time professor in terms of access rights.

Our delegation model for roles is different from conventional delegation models, where delegations to a role *automatically* propagate to *all* the entities that are delegated the role. However, it is important to make this distinction and our definitions provide higher assurance to the security of shared resources. For example, if hospital H delegates the right of reading a patient's medical record to the role $U.professor$, Alice would be entitled to this privilege in conventional delegation models, but not in our model. For sensitive data such as medical records, the automatic propagation of delegations to unknown roles may not always be desired by the resource owner. In comparison, our delegation model allows easy management of delegations for resource owners.

To support flexible decentralized delegation, we give to both role types (affiliated and delegated) the capability of delegating the role to other roles. Thus, in the above example, both Bob and Alice are able to delegate role $U.professor$ to other roles.

Even though our delegation mechanism is privilege-oriented, it is more efficient than the capability-list style delegations [17] because of the role abstraction. Delegations in our model may be issued to roles, as well as to individuals, which benefit from

the efficiency, scalability, and simplicity brought by the role-based delegation. The delegated privileges are role assignments, therefore, role-based cascaded delegation approach is more efficient than the capability-lists.

2.2.2 Terminology

As in the *RT* framework [90], we define an *entity* to be either an organization or an individual. An entity may issue credentials and make requests. Also, an entity may have one or more affiliated roles or delegated roles, which are authenticated by role credentials. An *affiliated role credential* is the credential for an affiliated role, and is signed by the administrator of the role. Similarly, a *delegated role credential* is the credential for proving a delegated role. Both credentials are issued using our delegation protocol. An affiliated role can be viewed as delegated directly by the administrator of the role. A *privilege* can be a role assignment or an action on a resource. The *original issuer* or *original delegator* of privilege P is the first entity on a delegation chain, and is the owner of the resources associated with privilege P. A *delegation chain* of privilege P is the path that shows the delegation sequence of P between entities. The chain connects a delegated entity to the original issuer of P. Given an entity on a delegation chain, the preceding entities on the chain are the *ancestor* entities.

An *extension credential* is generated and signed by a delegator on delegation transaction information, such as identities of the delegator and delegatee, and the delegated privilege. An *extension signature* is the signature on an extension credential. A *role signature* of an entity is the signature on an affiliated role credential of the entity. The *identity signature* of an entity is a signature computed by the entity using her private key. A *complete delegation credential* includes the identity signature of the requester, extension signatures, and role signatures. A *partial delegation credential* is a delegation credential issued to a role. It cannot be used by an individual for proving authorization, as it lacks the identity and role signatures of the requester.

We give the definition for independently-verifiable decentralized role-based delegation as follows.

Definition 2.2.1 *A delegation is an independently-verifiable decentralized role-based delegation if and only if the following requirements are satisfied:*

1. [**Decentralized**] *$Domain_1$ and $Domain_2$ are two independent administrative domains. Let $Domain_1.r$ be a role administrated by $Domain_1$.*

2. [**Role-Based**] *The delegation is issued by a member of role $Domain_1.r$ to delegate privileges associated with r to members of $Domain_2$.*

3. [**Independently-verifiable**] *Given the public key associated with $Domain_1$, the delegation credential can be verified by any third-party without the participation of the administrator of $Domain_1$.*

We will illustrate further this above definition in later sections.

2.2.3 Notations

We use a simple notation to express delegation credential. We allow the delegation of role memberships, and delegation to roles. A role r administered by entity A is denoted as $A.r$. Entity A is the administrator of role $A.r$. A role defines a group of entities who are members of this role. An affiliated role $A.r_a$ defines a subset of a role $A.r$ that contains a group of entities whose role credentials are directly issued by A. Similarly, a delegated role $A.r_d$ defines a subset of the role $A.r$ that contains entities whose role credentials are not directly issued by A. Role $A.r_a$ and $A.r_d$ define $A.r$, i.e. $A.r = A.r_a \cup A.r_d$. If an entity D has an affiliated role $A.r$, his *role credential* is denoted by $A \xrightarrow{A.r} D$, which indicates that D is assigned role $A.r$ by the role administrator A. Entity D can further delegate role $A.r$ to a role $B.s$ (administered by B) by issuing an *extension credential*, which is denoted by $D \xrightarrow{A.r} B.s$. Similarly, any member entity E of role $B.s$ can further delegate role $A.r$ to a role $C.t$ (administered by C). The corresponding extension credential is denoted by $E \xrightarrow{A.r} C.t$.

2.2.4 HCBE Preliminaries

Here, we give a brief overview of the necessary cryptographic knowledge.

The *Hierarchical Certificate-based Encryption* (HCBE) scheme [61] is a public key cryptosystem, where messages are encrypted with public keys and decrypted with corresponding private keys. What is unique about HCBE is that it makes the decryption

ability of a keyholder contingent on that keyholder's acquisition of a hierarchy of signatures from certificate authorities. To decrypt a message, a keyholder needs both his private key and the public key certificates (signatures) that are respectively signed by a chain of CAs. The CA hierarchy consists of a root CA and lower-level CAs. Higher-level CA certifies the public key of the next-level CAs, and the CAs at the bottom (leaf positions) of the hierarchy certify the public keys of individual users.

HCBE is based on the aggregate signature scheme [23, 26], which supports aggregation of multiple signatures on distinct messages from distinct users into one short signature. The HCBE scheme [61] has six algorithms, HCBE_Setup, HCBE_Cert_of_CA, HCBE_Cert_of_Bob, HCBE_Aggregate, HCBE_Encryption, and HCBE_Decryption. The second and the third algorithms are essentially the same; HCBE_Cert_of_CA is for certifying the public keys of CAs, and HCBE_Cert_of_Bob is for certifying the public key of an individual. The API of the algorithms are given below.

HCBE_Setup: A set of system parameters *params* is generated. Among other parameters, *params* contain two cryptographic hash functions H and H', a bilinear map $\hat{e}$, and a constant π with certain properties. A bilinear map [22] is a mapping function $\hat{e}(x, y)$ that takes two inputs x and y, and outputs a value. Each entity D chooses his private key s_D, computes and publishes his public key $s_D\pi$.

HCBE_Cert_of_CA(s_i, $info_{i+1}$): CA at i-th level runs this algorithm to certify the public key of the CA at level $i + 1$ by computing a signature. The first input is the private key of CA_i, and the second input is a string $info_{i+1}$ that contains the public key $s_i\pi$ of the signer and the public key $s_{i+1}\pi$ of CA_{i+1}. The string $info_{i+1}$ may also include information such as the expiration date, etc.

HCBE_Cert_of_Bob(s_{n-1}, $info_n$): CA_{n-1} runs this algorithm to certify the public key of Bob. The first input is the private key of CA_{n-1}, and the second input is a string $info_n$ that contains the public key $s_{n-1}\pi$ of the signer and the public key $s_n\pi$ of Bob.

HCBE_Aggregate(s_n, $info'$, $sig_2, \ldots, sig_n$): This algorithm is run by Bob, who uses his private key s_n and the public key certificates on his chain to compute an aggregate signature, which will be used as his decryption key. The inputs to this algorithm are Bob's private key s_n, the string $info'$ that contains the information of Bob, and a

number of signatures [1] that contains the public key certificate signatures associated with his certification chain.

HCBE_ENCRYPTION($M, info_1, \ldots, info_n, info'$): Alice computes the ciphertext to send to Bob. The inputs are a message M, string $info_i$ of the certification at level i on Bob's chain for $1 \leq i \leq n$, and string $info'$ that Bob signs in HCBE_AGGREGATE.

HCBE_DECRYPTION(C, S_{Agg}): Bob decrypts the ciphertext C to retrieve the message using his aggregate signature S_{Agg}.

The security of HCBE assures that a ciphertext for an individual can only be correctly decrypted using both the receiver's private key and his public key certificate obtained from the hierarchy of CAs. We slightly modify the encryption and decryption schemes in HCBE scheme for our verification of delegation chain. In our protocol in Section 2.6, the requester computes an aggregate signature, and gives it to the verifier. The verifier encrypts a message with the delegation chain information, and attempts to decrypt the ciphertext with the aggregate signature. Successful decryption verifies the delegation chain.

2.3 Overview of Role-Based Cascaded Delegation

We propose a model for the delegation of authority in role-based trust management systems, called *role-based cascaded delegation*. The main goal of this model is to allow flexible transfer of privileges and sharing of resources in decentralized environments. Our model allows a role member to delegate his or her privileges to users who may belong to different organizations, as opposed to restricting this delegation ability to role administrators in traditional access control models. In addition, our role-based cascaded delegation model allows a delegatee to further extend the delegated privileges to other collaborators. The challenge arise in realizing this goal in a decentralized environment is that the public key of an intermediate delegator may not be known by a verifier or the resource owner. Therefore, the delegation credential signed by that delegator may not be trusted by the verifier.

To solve this problem, we borrow the concept of cascaded delegation from distributed systems literature [93, 112]. The distributed cascaded delegation problem

[1] HCBE_AGGREGATE can take any number of signatures.

is essentially to design a delegation mechanism that efficiently verifies a hierarchical delegation chain. In the cascaded delegation model, a delegation recipient E may further extend the delegated privilege to another entity E', and the delegation credentials of E are passed to entity E' along with the delegation certificate signed by E as the issuer. The public key of the next delegatee is encoded in the delegation credential, which naturally forms a chain of trust. Therefore, trusting the original delegator means that the delegatees' public keys are authorized by the delegation. In addition, the authorization chain is stored in delegation credentials and does not have to be dynamically discovered. However, previous cascaded delegation protocols support neither multiple administrative domains nor the use of roles in the delegation. We give support to both in our role-based cascaded delegation model.

In our role-based cascaded delegation, given a privilege, two types of entities can delegate the privilege to others. One is the resource owner of the privilege. The other is a member of a role who is delegated the privilege. A role r is delegated a privilege by receiving a delegation credential C that explicitly assigns the privilege to role r. Members of the role r are allowed to further delegate the privilege to another role r' as follows. A member D of the role r uses the delegation credential C to generate a delegation credential C'. C' comprises multiple component credentials, which include the credential of the current delegation authorization, the credential C from the preceding delegation, and the role membership credential of the delegator D. The verifier can make the authorization decision based on delegation credential C' and the role membership credential of the requester. The verification can be done by any party without the participation of any role administrators, which is called by us as independent verifiability (See also Section 2.2).

The length of a delegation chain in role-based cascaded delegation refers to the number of delegators involved. A privilege P is delegated by an entity E to a role r_1. A member D of role r_1 further delegates the same privilege P to role r_2. The delegation chain of privilege P involves entity E, role r_1, entity D, and role r_2. Role r_2 receives the privilege P as the result of the delegation chain. The length of the chain is two.

Decentralized role-based delegation allows users from administratively independent domains to be dynamically joined according to the needs of the tasks. We have also explored the applications of RBCD for efficient and flexible trust establishment

in decentralized and pervasive environments in [129].

2.4 RBCD Example

In this section, we describe a delegation example for the role-based cascaded delegation model. Suppose a collaboration project is established between a hospital H and a medical school M. To facilitate the collaboration, the hospital initiates a delegation chain and delegates its role $H.guest$ to the affiliated role $M.professor$ at the medical school. Hospital H is the administrator of the role $H.guest$. The delegation is expressed in the partial delegation credential (2.1), using the notation described in Section 2.2.

$$H \xrightarrow{H.guest} M.professor \tag{2.1}$$

In credential (2.1), hospital H is the original issuer, $H.guest$ is the delegated privilege, and $M.professor$ is the role that receives the delegation.

The hospital H allows members of the role $M.professor$ to further delegate $H.guest$ role to whomever they deem necessary to accomplish the project. Bob is a professor at M and has an affiliated role credential (2.2).

$$M \xrightarrow{M.professor} \text{Bob} \tag{2.2}$$

For a task in the collaboration project, Bob subcontracts to a lab L. Lab L is independent from school M and is unknown to the hospital H. Lab L defines a research assistant role $L.assistant$. In order for members of the role $L.assistant$ to work on the task and utilize the resources of the hospital H, Bob delegates the role $H.guest$ to the affiliated role $L.assistant$. In our role-based cascaded delegation model, Bob issues a partial delegation credential (2.3) by extending the delegation credential (2.1) to role $L.assistant$.

$$\begin{aligned}(H \xrightarrow{H.guest} M.professor), (M \xrightarrow{M.professor} \text{Bob}),\\ (\text{Bob} \xrightarrow{H.guest} L.assistant)\end{aligned} \tag{2.3}$$

Credential (2.3) also includes Bob's role credential (2.2) for proving that he is allowed to delegate $H.guest$. (2.3) is a partial delegation credential for role $L.assistant$.

Recall that (2.3) is different from the linked role in *RT* framework [89], as the role *H.guest* is delegated, not *M.professor*. Alice is a research assistant in lab L, and has an affiliated role credential (2.4) issued by lab L to prove this role membership.

$$L \xrightarrow{L.assistant} \text{Alice} \tag{2.4}$$

(2.4) is equivalent to the role membership representation below, as in *RT* framework. (2.5) is read as Alice has a role of *L.assistant*.

$$L.assistant \leftarrow \text{Alice} \tag{2.5}$$

Because (2.4) is issued by the lab L, the role *L.assistant* is Alice's affiliated role. To prove that she has the hospital's delegated *guest* role, Alice obtains the delegation credential (2.3) for role *L.assistant* from a credential server, and aggregates it with her affiliated role credential (2.4). This delegation generates credential (2.6).

$$\begin{aligned} &(H \xrightarrow{H.guest} M.professor), \quad (M \xrightarrow{M.professor} \text{Bob}), \\ &(\text{Bob} \xrightarrow{H.guest} L.assistant), \quad (L \xrightarrow{L.assistant} \text{Alice}) \end{aligned} \tag{2.6}$$

Credential (2.6) and the identity signature of Alice yield a complete delegation credential for Alice. For verification, the hospital H does not need to discover the delegation chain that connects Alice with role *H.guest*, because this information is contained in credential (2.6). Furthermore, the lab administrator does not have to participate in the verification of Alice's role membership as this information is also in (2.6). The hospital makes the authorization decision by verifying each component of credential (2.6) and Alice's identity signature. Note that the hospital does not need to have prior knowledge of or trust relationship with lab L. This independent verifiability enables a cross-domain authorization chain to be easily verified.

We allow actions to be delegated, as well as roles. For example, the hospital may delegate the read access of a database *db* (*Read db*) to role *M.professor*, which is expressed in (2.7).

$$H \xrightarrow{(Read\ db)} M.professor \tag{2.7}$$

2.5 Role-based Cascaded Delegation Protocol

In this section, we first describe the role-based cascaded delegation protocol and then show an efficient realization of this protocol using the HCBE scheme [61]. In what follows, a role r represents an affiliated role.

The role-based cascaded delegation protocol defines four operations: RBCD_Initiate, RBCD_Extend, RBCD_Prove, and RBCD_Verify. In our protocol description, delegation credentials once issued are stored in public credential servers that can be queried by anyone. The credential servers (See also Section 2.7.2) may be simple LDAP servers. Because of our security guarantees (See Section 2.7.1, adversaries cannot use the credentials on the servers to forge authorization.

- RBCD_Initiate(P_{D_0}, s_{D_0}, $D_0.priv$, $A_1.r_1$, P_{A_1}): This operation is run by the administrator D_0 of a privilege $D_0.priv$ to delegate $D_0.priv$ to an affiliated role $A_1.r_1$. This operation initiates a delegation chain for privilege $D_0.priv$. Inputs are the public key P_{D_0} of entity D_0, the corresponding private key s_{D_0}, the delegated privilege $D_0.priv$, the role name $A_1.r_1$, and the public key P_{A_1} of role administrator A_1. Recall that only affiliated roles can receive delegations, as discussed in Section 2.2.1. The output is a partial delegation credential C_1 for the role $A_1.r_1$, represented as

  $$D_0 \xrightarrow{D_0.priv} A_1.r_1.$$

  The statement of C_1 includes the public key P_{D_0}, the privilege $D_0.priv$, and information about the role $A_1.r_1$ such as the role name and the public key of the administrator A_1. The delegation certificate is signed using the private key s_{D_0}. D_0 stores C_1 on a credential server.

  Note that if the last argument is the public key of an individual, this operation can also be used for generating role certificates. Role certificate is given to the corresponding role member.

- RBCD_Extend ($s_{D_n}, D_0.priv, C_n, R_{D_n}$, $A_{n+1}.r_{n+1}$, $P_{A_{n+1}}$):

  This operation is run by an intermediate delegator D_n, who is a member of an affiliated role $A_n.r_n$, to extend the delegation of privilege $D_0.priv$ to the role $A_{n+1}.r_{n+1}$. The inputs are the private key s_{D_n} of the delegator D_n, the

delegated privilege $D_0.priv$, the partial delegation credential C_n that delegates the privilege $D_0.priv$ to the role $A_n.r_n$, the role credential R_{D_n} of the delegator D_n, the role name $A_{n+1}.r_{n+1}$, and the public key $P_{A_{n+1}}$ of role administrator A_{n+1}. Credential C_n is retrieved from a credential server. The partial delegation credential C_n is a function of the preceding extension and role credentials, which are denoted as:

$$\begin{array}{cc} & (D_0 \xrightarrow{D_0.priv} A_1.r_1), \\ (A_1 \xrightarrow{A_1.r_1} D_1), & (D_1 \xrightarrow{D_0.priv} A_2.r_2), \\ \multicolumn{2}{c}{\ldots} \\ (A_{n-1} \xrightarrow{A_{n-1}.r_{n-1}} D_{n-1}), & (D_{n-1} \xrightarrow{D_0.priv} A_n.r_n) \end{array}$$

where D_0 represents the resource owner, and $A_i.r_i$ is the role that is delegated the privilege $D_0.priv$ by an entity D_{i-1} who has the affiliated role $A_{i-1}.r_{i-1}$, for $i \in [1, n]$.

An extension credential denoted by $D_n \xrightarrow{D_0.priv} A_{n+1}.r_{n+1}$ is generated as an intermediate product of the operation RBCD_Extend. Its statement contains information about the delegated privilege $D_0.priv$ and the role $A_{n+1}.r_{n+1}$. It is signed with the private key s_{D_n}. The final output of this operation is a partial delegation credential C_{n+1}, which is a function of the credential C_n, the role credential R_{D_n} denoted by $A_n \xrightarrow{A_n.r_n} D_n$, and the extension credential described above.

Credential C_{n+1} may simply be delegation credential C_n together with two individual credentials. Alternatively, D_n can compute a delegation credential for the role $A_{n+1}.r_{n+1}$ as in existing cascaded delegation protocols [50, 95], and also passes down his role credential to members of the role $A_{n+1}.r_{n+1}$. In comparison, our realization using HCBE [61] scheme provides a more efficient approach.

- RBCD_Prove($s_{D_n}, D_0.priv, R_{D_n}, C_n$):

  This operation is performed by the requester D_n who wants to exercise privilege $D_0.priv$. D_n is a member of the affiliated role $A_n.r_n$. The requester D_n uses the partial delegation credential C_n and D_n's affiliated role credential R_{D_n}, denoted by $A_n \xrightarrow{A_n.r_n} D_n$, to prove that he is authorized the privilege $D_0.priv$.

The inputs are the private key s_{D_n} of the requester D_n, the privilege $D_0.priv$, the affiliated role credential R_{D_n} of the requester, and the delegation credential C_n. Credential C_n is retrieved by the requester from a credential server. The operation produces a proof F, which contains delegation statements and corresponding signatures for verification. The private key s_{D_n} is for proving the authenticity of the public key P_{D_n} that appears on the role credential R_{D_n} of the requester.

- RBCD_VERIFY(F):

  This operation is performed by the resource owner D_0 to verify that the proof F produced by the requester D_n correctly authenticates the delegation chain of privilege $D_0.priv$. D_n is a member of the role $A_n.r_n$. The input is a proof F that is computed by the requester D_n. F contains signatures and a string tuple $[D_0.priv, P_{D_0}, A_1.r_1, P_{A_1}, P_{D_1}, \ldots, P_{D_{n-1}}, A_n.r_n, P_{A_n}, P_{D_n}]$ that consists of the components of a delegation chain for requester D_n. In the string tuple, $D_0.priv$ is the delegated privilege, for $i \in [1, n]$ $P_{D_{i-1}}$ is the public key for the delegator D_{i-1} whose affiliated role is $A_{i-1}.r_{i-1}$, $A_i.r_i$ is the role that receives the delegation from D_{i-1}, P_{A_i} is the public key of role administrator A_i, and P_{D_n} is the public key of the requester. The verifier checks whether the signatures in F correctly authenticates the delegation chain. This process includes the authentication of each delegation extension $D_{i-1} \xrightarrow{D_0.priv} A_i.r_i$, and entity D_i's affiliated role membership $A_i \xrightarrow{A_i.r_i} D_i$, for all $i \in [1, n]$. F also contains the proof of possession of private key s_{D_n} that corresponds to public key P_{D_n}. D_n is granted $D_0.priv$ if the verification is successful, and denied if otherwise.

Our role-based cascaded delegation model supports independently-verifiable decentralized role-based delegation. Recall that independently-verifiable decentralized role-based delegation is defined in Section 2.2 as the ability for a member of role r to delegate r to other roles or entities, and in addition the delegation credential can be independently verified by any third-party without the participation of the administrator of role r. In RBCD, RBCD_EXTEND is performed by a valid member of role r to delegate r to others. The partial delegation credential generated contains the role credentials of all delegators on the delegation chain. Therefore, the verification of the

delegation credentials does not require any role administrators, and can be performed by anyone.

Affiliated role credentials can be issued using RBCD_Initiate operation by the administrator of a role. RBCD_Extend operation is used to issue delegated role credentials. The delegation chain of a privilege grows at each delegation extension. The verifier may perform revocation checking at the RBCD_Verify operation. Delegation revocation is discussed in Section 2.7. In the next section, we describe a realization of cascaded delegation using the Hierarchical Certificate-based Encryption [61], which allows aggregation of multiple credentials into one credential.

2.6 Realization

Role-based cascaded delegation can be implemented in a straightforward manner using the RSA signature scheme [100]. At each delegation, the delegator D computes an RSA signature on the delegation statement, and issues it to delegatees along with D's role signature (also an RSA signature). The delegation chain verification consists of verifying each of the above signatures.

We present a more efficient realization of role-based cascaded delegation using the Hierarchical Certificate-based Encryption (HCBE) [61] scheme. In HCBE, each entity has a public/private key pair generated on his own. A member of an affiliated role has an affiliated role credential, which contains a signature signed by the administrator of the role. The delegation credential in this protocol consists of an aggregate signature and a string tuple.

In RBCD, a delegator issues a partial delegation credential to a role, which is not valid until a member of the affiliated role adds in his role credential and identity information. The complete delegation credential of an entity is computed by the entity, using the partial delegation credential obtained through credential servers, his role credential, and his secret personal information. Each member of an affiliated role has a *unique* complete delegation credential, however, the delegator only needs to generate *one* partial delegation credential, which does not require the knowledge of the members of that affiliated role. This feature makes our protocol scalable. Any member of that affiliated role can further delegate the privilege to other affiliated roles,

without any assistance from administrators. The public information of intermediate delegators is traceable. The affiliated role membership of all the delegators on a delegation chain can be proved, however, the signatures on their role credentials are not revealed to anyone.

A delegation credential of an entity corresponds to a delegation chain, and has two components: one aggregate signature of constant size and a string tuple. The string tuple defines the delegation chain, and its size is linear in the length of the chain. The signature is used for authentication of the chain. The aggregate signature [23] in the HCBE scheme is an ordinary sized signature that is the aggregation of multiple signatures, which may include signatures from delegators, role administrators, and the requester. To request a service, the requester uses his private key to sign a statement which is chosen by the verifier, and aggregates this signature with signatures from his role credential and the partial delegation credential obtained from a credential server. To verify the delegation chain, one simply verifies that aggregate signature submitted by the requester.

Our role-based cascaded delegation protocol has five operations, which make use of the algorithms in the HCBE scheme [61]. Alternatively, one can use operations in the aggregate signature scheme [23] for generating and verifying delegation credentials. We choose to use HCBE for the presentation, because its operations have intuitive meanings that are similar to our needs.

RBCD_Setup: This operation outputs the system parameters, public/private keys, and role credentials that will be used in the system.

- The root of the system calls HCBE_Setup and obtains a set of public parameters denoted as *params*. Among other parameters in *params*, including collision-resistant hash functions H and H', a special constant π, and a bilinear map $\hat{e}$ [22].

- Each entity (organization or individual) D chooses a secret s_D as his private key, and computes the product $s_D\pi$ as its public key P_D.

- An organization A with the private key s_A certifies entities who have $A.r$ as an affiliated role. For each entity D who has the affiliated role $A.r$ and the public key P_D, organization A computes a role signature R_D by running

HCBE_CERT_OF_CA($s_A, P_D \| A.r$), where $\|$ denotes string concatenation. The output signature, representing the role assignment $A \xrightarrow{A.r} D$, is given to entity D for proving the affiliated role membership.

RBCD_INITIATE: Resource owner D_0 delegates the privilege $D_0.priv$ to members of an affiliated role $A_1.r_1$. The private key s_{D_0} corresponds to the public key P_{D_0} of entity D_0. Entity D_0 does the following.

- Set the string $info_1 = P_{D_0} \| D_0.priv \| A_1.r_1 \| P_{A_1}$. String $info_1$ contains the public key P_{D_0} of the owner of the delegated privilege, the delegated privilege $D_0.priv$, the role $A_1.r_1$ that receives the privilege, and the public key P_{A_1} of the administrator of the role $A_1.r_1$. Run HCBE_CERT_OF_CA($s_{D_0}, info_1$), which outputs an extension signature X_1. Define a string tuple $chain_1$ as [$D_0.priv$, P_{D_0}, $A_1.r_1$, P_{A_1}]. Set the partial delegation credential C_1 for the role $A_1.r_1$ as ($X_1, chain_1$). Credential C_1 is put on a credential server.

RBCD_EXTEND: An entity D_i, whose role is $A_i.r_i$, further delegates $D_0.priv$ to role $A_{i+1}.r_{i+1}$. D_i uses his private key s_{D_i}, his role signature R_{D_i}, and the delegation credential C_i of the role $A_i.r_i$ to compute a partial delegation credential C_{i+1}. Entity D_i does the following.

- Parse the credential C_i as ($S_{Agg}, chain_i$), where S_{Agg} is the aggregate signature of credential C_i and $chain_i$ is the corresponding string tuple. Signature S_{Agg} is a function of preceding extension and role signatures on the delegation chain. String tuple $chain_i$ contains the components of the delegation chain. Set the string $info_{i+1} = P_{D_0} \| D_0.priv \| A_{i+1}.r_{i+1} \| P_{A_{i+1}}$, where P_{D_0} is the public key of the resource owner of the delegated privilege, $D_0.priv$ is the delegated privilege, $A_{i+1}.r_{i+1}$ is the role that receives the privilege, and the public key $P_{A_{i+1}}$ of the role administrator A_{i+1}. Run HCBE_AGGREGATE($s_{D_i}, info_{i+1}, R_{D_i}, S_{Agg}$), which outputs an aggregate signature S'_{Agg}.

- Define the string tuple $chain_{i+1}$ of credential C_{i+1} as the string tuple $chain_i$ appended with public key P_{D_i}, the role name $A_{i+1}.r_{i+1}$, and the public key $P_{A_{i+1}}$. Set credential C_{i+1} as ($S'_{Agg}, chain_{i+1}$). The partial delegation credential C_{i+1} for the role $A_{i+1}.r_{i+1}$ is put on a credential server.

RBCD_PROVE: The requester D_n with the role signature R_{D_n} and delegation credential C_n wants to use the delegated privilege $D_0.priv$. D_n is given a random message T by the verifier D_0. The message T contains some random information to prevent a replay attack. D_n does the following.

- Parse the credential C_n as $(S_{Agg}, chain_n)$, where S_{Agg} is the aggregate signature of C_n and $chain_n$ is the string tuple. Run HCBE_AGGREGATE$(s_{D_n}, T, R_{D_n}, S_{Agg})$, where s_{D_n} is the private key of D_n. HCBE_AGGREGATE outputs an aggregate signature S'_{Agg}. Set the string tuple $chain'_n$ to be $chain_n$ appended with the public key P_{D_n} of D_n. Set the proof F to be $(S'_{Agg}, chain'_n, T)$, which is sent to the verifier D_0.

RBCD_VERIFY: The verifier D_0 verifies the proof F submitted by the requester D_n as follows.

- Parse F as $(S'_{Agg}, chain'_n, T)$, where S'_{Agg} is an aggregate signature, $chain'_n$ is a string tuple, and T is a message. Parse the string tuple $chain'_n$ as $[D_0.priv$, P_{D_0}, $A_1.r_1$, P_{A_1}, ..., $A_n.r_n$, P_{A_n}, $P_{D_n}]$, where for $i \in [0, n-1]$ P_{D_i} is the public key of delegator D_i whose affiliated role is $A_i.r_i$, $A_{i+1}.r_{i+1}$ is the role receiving the delegation from D_i, $P_{A_{i+1}}$ is the public key of role administrator A_{i+1}, and P_{D_n} is the public key of the requester.

- Encrypt a message M in HCBE as follows. Choose a random number r. Set the ciphertext $Ciphertext = [r\pi, V]$, where π is one of the public parameters, $V = M \oplus H'(g^r)$, where $g = g_1 g_2 g_3$ is a product of the following: $g_1 = \hat{e}(P_{D_n}, H(T))$, $g_2 = \Pi_{i=1}^{n} \hat{e}(P_{A_i}, H(P_{D_i} \| A_i.r_i))$, $g_3 = \Pi_{i=0}^{n-1} \hat{e}(P_{D_i}, H(P_{D_0} \| D_0.priv \| A_{i+1}.r_{i+1} \| P_{A_{i+1}}))$.

  The value g is the product of multiple bilinear map functions [22] whose inputs are the public key of a signer and the hash digest of the signed message. H and H' are the two hash functions in the system parameters $params$. $\oplus$ denotes bit-wise XOR operation. T is the message that D_n signs in RBCD_PROVE.

- Run HCBE_DECRYPTION$(Ciphertext, S'_{Agg})$ to decrypt ciphertext $Ciphertext$ using S'_{Agg}. Compare the output M' of the decryption with the original message M. The request is granted if $M = M'$, denied if otherwise.

The correctness of the protocol can be directly deduced from the correctness of HCBE and is not shown here.

A delegation to the intersection of roles [89], for example $A_1.r_1 \cap A_2.r_2$, may be realized by extending one delegation to a string that represents an intersection of roles, rather than one role. To extend or prove such a delegation, an entity needs to aggregate two, rather than one, role signatures into a delegation credential. Additional fields can be added by the delegator to a delegation credential to increase the expressiveness, one of them being the expiration date of a delegation. Given a delegation chain defined by the credential, the expiration date of a delegation should be no later than any of the expiration dates of preceding delegations. The delegator may also set restrictions on the level of a delegation, which specifies whether or not the privilege can be further delegated and for how many times, i. e., the length of a delegation chain. This constraint helps improve the accountability, and gives the delegator a tight control over the delegated privileges. The verifier or the delegation receiver should check if all the constraints are satisfied before accepting a credential. Supporting the RBCD model with predicates and constraints was recently presented in [129].

We discuss the security, efficiency, and scalability of role-based cascaded delegation protocol in the next section.

2.7 Analysis

We now analyze the security, efficiency, scalability, and revocation of role-based cascaded delegation.

2.7.1 Security

In this section, we first analyze the security of our role-based cascaded delegation model, and then describe the security of the RBCD realization with aggregate signatures.

The security property of the RBCD model is defined as follows: unauthorized entities cannot access protected resources, and unauthorized entities cannot issue valid delegations. We allow adversaries to do the following: (1) adversaries can observe

communications between delegation participants and between resource owners and requester; (2) adversaries can forge delegation credentials or role credentials; and (3) adversaries can submit access requests. We assume the existence of a signature scheme that is secure against forgery attacks by (probabilistic) polynomial-time adversaries.

Theorem 2.7.1 states the security of the RBCD model.

Theorem 2.7.1 *In role-based cascaded delegation model, given a partial delegation credential for a role r, a polynomial-time adversary cannot forge a valid delegation chain that authorizes the role r to any role or individual.*

The analysis of the theorem is straightfoward as follows. The partial delegation credential is generated by INITIATE or EXTEND operations. A partial delegation credential is issued to roles, rather than to individuals. To use the partial delegation credential for role r to request for access, one needs to have a valid role credential R_r of role r and the private-key corresponding to the public-key stated in R_r. The latter is for signing the challenge nonce from the resource owner. Given any secure signature scheme against polynomial-time adversaries, an adversary cannot forge role credential and the signature on the nonce. Therefore, she cannot use the partial delegation credential to authorize the role r to herself. In addition, an adversary cannot forge valid extension credentials to extend role r, because she is unable to forge a valid role credential of role r that is required in EXTEND operation.

The RBCD realization with aggregate signatures provides strong protection of sensitive signatures because individual signatures can be verified without being disclosed. To extend a delegation, an intermediate delegator aggregates two signatures. One is his role signature signed by a role administrator, and the other is the extension signature signed by the delegator himself. Once the role signature and the extension signature are aggregated with the signature from the previous delegation (the order does not matter), it is impossible for others to find out what the role signature or the extension signature is. Similarly, for a requester, the role signature and the signature on a challenge statement are also protected. This is not achievable in conventional signature schemes, such as RSA [24].

Furthermore, the security of the aggregate signature and HCBE schemes guarantees that an attacker cannot forge a valid aggregate signature consisting of n individual

signatures, even if he possesses $n-1$ of the required private keys [23]. In our delegation model, this implies that one cannot forge any valid delegation credential from existing credentials. Although signature verification can be performed by anyone, an adversary cannot derive any signature nor secret key of the preceding delegators from the aggregate signature that is issued to him. HCBE also guarantees that collusions between users do not give them any information more than what they have already known.

2.7.2 Scalability

The abstraction of roles in role-based cascaded delegation greatly reduces the potential for a large number of delegation credentials, and makes the model scalable. Because the partial delegation credentials issued by the delegators cannot be directly used for accessing resources, they may be stored at credential servers so that members of a role can query the server to retrieve the partial credential. Thus, our implementation scales up to a large number of credential receivers. Also, the delegation is decentralized. Individuals, who have qualified roles, can make delegations of the roles without the assistance of administrators. In collaboration environments where coalitions are formed dynamically, this feature greatly facilitates resource sharing. Note that our model does not require the existence of public-key infrastructure.

2.7.3 Efficiency

We analyze the efficiency of RBCD model, and compare its realizations with RSA and aggregate signatures. The size of delegation credentials in our model is formalized below.

Theorem 2.7.2 *In role-based cascaded delegation, the size of a partial or complete delegation credential is linear in the length of a delegation chain, which is the number of delegation transactions associated with the delegation credential.*

This complexity is because at each delegation transaction, one extension credential and one role credential are accumulated to existing delegation credentials.

Although the asymptotic sizes of delegation credentials in different RBCD realizations are the same, the implementation using HCBE and aggregate signatures can

have significant advantages in delegation efficiency, compared to an implementation using conventional credentials. We compare our HCBE-based realization with the realization using the RSA signature scheme [100] described at the beginning of Section 2.6. We consider a 1024-bit modulus RSA scheme, in which the size of the public key is slightly larger than 1024 bits and the size of a signature is 1024 bits long.

For the same level of security as 1024-bit modulus RSA, the signatures and public keys in the aggregate signature scheme can be as short as 170-bit long [26]. Observe that at each delegation extension of RBCD, the following information needs to be added to the delegation credential: the public key of the delegator, the role name of recipients, the public key of the role administrator, the signature on the role credential of the delegator, and the extension signature generated by the issuer. The analysis also applies to the AGGREGATE operation performed by the requester. Therefore, to authenticate a delegation chain of length n (i.e. having n delegations), the information required by the verifier includes the delegated privilege, the public keys of n delegators and n role administrators, n role names, the public key of the requester, along with $2n + 1$ digital signatures.

Suppose the length of a role name is 100 bits and the delegated privilege has the same size as a role name. The total size of the credential in our HCBE realization is $170 + 170(2n + 1) + 100(n + 1) = 440n + 440$ bits. For the RSA signature scheme, such a delegation credential contains $2n$ additional signatures, and the total size is at least $1024(2n + 1) + 1024(2n + 1) + 100(n + 1) = 4196n + 2148$ bits.

For example, consider a delegation chain of length 20. The size of the delegation credential in RSA is more than 86 Kbits, while in the HCBE realization it is about 9.2 Kbits. Smart cards with a microprocessor typically have 32 KBytes (256 Kbits) EEPROM storage. Thus, our approach has a clear advantage in terms of the number of credentials that can be stored by smart cards and similar devices. For small mobile devices with limited communication bandwidth, this saving in the credential size allows the information to be transmitted faster. The above analysis also applies to the EXTEND operation.

For a 20 Kbits per second connection and a delegation chain of length 20, the time for transmitting the entire RSA credentials to the verifier in the PROVE operation takes $(4196 \times 20 + 2148)/20000 = 4.30$ seconds. The time in our HCBE realization

takes $(440 \times 20 + 440)/20000 = 0.46$ seconds. In addition, generating a signature in HCBE scheme requires only 3.57 ms on a 1 GHz Pentium III, and is faster than generating a signature in the RSA scheme, which requires 7.90 ms for a 1007-bit private key on the same machine [11].

The running time for verifying an aggregate signature associated with a delegation chain is linear in the number of single signatures aggregated, i.e., the length of the chain. The verification of a signature in the HCBE scheme is slow (about 50 ms on a 1 GHz Pentium III) compared to RSA signature verification (0.40 ms on the same machine for a 1007 bits private key) [11]. Nevertheless, in our protocol only the servers of resource owners, which are typically powerful, have to performs delegation chain verifications.

Table 2.1 summarizes the analysis above.

Chain length $n = 20$	Credential size	Transmission (20 Kbit/s)	Signing [11]	Verifying [11]
RBCD using RSA	86 Kbits	4.3s	7.9ms	0.4ms
RBCD using Agg. Sig.	9.2 Kbits	0.46s	3.57ms	50ms

Table 2.1: Efficiency comparisons between RBCD realizations using RSA signatures and bilinear-map based aggregate signatures

2.7.4 Delegation renewal and revocation

At each delegation extension, the issuer can set an expiration date for the delegation, which may be earlier than the expiration dates of preceding delegations on the chain. For a delegation credential to be considered valid, none of the expiration dates has passed. Intermediate delegators may issue delegations with a short validity period, and then periodically renew them. Delegation renewal can be done in a hierarchical fashion as follows. To renew a delegation, a delegator E puts the renewed partial delegation credential on credential servers. Intermediate delegators that succeed to E may retrieve the renewed credential and update the corresponding delegations that are issued by them.

Delegation revocation before expiration can be handled by maintaining a revocation service, which can be efficiently achieved using the authenticated dictionary technique (see, e.g., [49, 69, 70, 94]). An authenticated dictionary is a system for

distributing data and supporting authenticated responses to queries about the data. The data originates at a secure central site (the repository) and is distributed to servers scattered around the network (responders). The responders answer queries about the data made by clients on behalf of the repository and provide a proof of the answer.

The roles or public keys whose delegated privileges are revoked are put on the repository of the revocation service by the resource owner. Before verifying the credential signatures in the VERIFY operation, the resource owner queries the revocation service to ensure that no public key whose delegated privileges are revoked appears on the delegation credential. Similarly, the revocation of affiliated role memberships can also be supported using a revocation service, which the verifier queries in the VERIFY operation to ensure the validity of the affiliated role memberships of intermediate delegators.

2.8 Related Work

The PolicyMaker [18] and KeyNote [17] are the first trust management systems that authorize decentralized access by checking a proof of compliance. The system puts all the policy and credential information into signed certificates that are programmable. Certificates in PolicyMaker are generalized as they consist of programs written in a general programming language. SPKI/SDSI (Simple Public Key Infrastructure/Simple Distributed Security Infrastructure) is a public-key infrastructure emphasizing decentralized name space and flexible authorization [43, 53]. The owner of each public key can create a local name space relative to that key. These name spaces can be linked together to enable chains of authorization and define groups of authorized principals. To access a protected resource, a client must show a proof that takes the form of a certificate chain proving that the client's public key is one of the groups on the resource's ACL, or that the client's public key has been delegated authority from a key in one of the groups on the resource's ACL. Due to the flexible naming and delegation capabilities of SPKI/SDSI certificates, finding such a chain can be nontrivial.

Compared to RBCD, PolicyMaker, KeyNote, and SPKI/SDSI do not define role

abstractions, and thus delegations can only be issued to individuals. The use of roles makes authorization scalable, and in the meantime, the role-based delegation mechanism is more complex as demonstrated in our work. In addition, these systems assume that all certificates are centrally stored, which may not be realistic in decentralized environments. In comparison, we address this issue with a simple accumulation approach by having delegators to pass down relevant credentials.

The *RT* framework is a family of Role-based Trust management languages for representing policies and credentials in decentralized authorization [89]. Compared to our work, the work of *RT* focuses on the high-level expressiveness aspect of trust management, and does not address the cryptographic verification problem of authorization chains as studied in our work. Our delegation mechanism is general and can be incorporated into existing role-based trust management systems such as *RT* to instantiate a concrete delegate mechanism. Details of how this incorporation is done is out of the scope of this work.

As we said earlier in the introduction, our role-based delegation can be simplified to support individual delegation, i.e., a role member further extends his or her delegated privileges to another individual. Therefore, TM systems such KeyNote, PolicyMaker, and SPKI/SDSI can also utilize our protocol to instantiate their delegation mechanisms.

QCM [75] and SD3 [79] are two trust-management systems that consider distributed storage of credentials. A limitation of the approach in QCM and SD3 is assuming that issuers initially store all the credentials, which may be impractical for some applications. This limitation was addressed by Li *et al.* [90], who presented goal-directed credential chain discovery algorithms that support a more flexible distributed storage scheme in which credentials may be stored by their issuer, their recipient (also called their "subject"), or both. The algorithms dynamically search for relevant credentials from remote servers to build a proof of authorization. While storing credentials with their issuers or recipients supports flexible delegation models, in many cases such flexibility is unnecessarily costly. The discovery algorithms require delegation issuers or their responders (credential servers) to participate in the computation. Role-based cascaded delegation can be integrated with the credential chain discovery algorithms to reduce the communicational and computational costs to a

certain degree [129]. This is because part of the target authorization chain is already captured in RBCD's delegation credentials and does not need to be discovered.

There are several cryptographic cascaded delegation [112] schemes for the proxy authentication and authorization, including nested signature schemes [118], delegation keys [95], and a combined approach [50]. These schemes do not provide the support for delegations to roles, and the delegation credentials are not as compact as ours, as is explained in the following. Nested signatures define the order of delegations on a delegation path. They are used to prevent the attacker to construct another delegation path using one of the delegation credential [118]. The size of delegation credential is linear to the number of entities on a delegation chain, and verification of signatures is done sequentially. Cascaded delegation is also implemented by binding two delegation credentials using delegation keys [95]. Applying this scheme to role-based delegation means sharing secret group key among members of a role, which may cause accountability problem. The hierarchical delegation protocol by Ding *et al.* [50] combines the nested signature scheme and delegation public/private key approach. It is based on Schnorr signature scheme [106], self-certified public keys [62], and the concept of hierarchical key generation [60]. Compared to our realization using HCBE, their delegation and verification algorithms require more computations. In their scheme, to verify one hierarchical delegation credential of length l, a verifier has to compute and verify l public delegation keys (different from public keys in conventional PKI). In addition, at each delegation the delegation receiver is required to perform a number of computations. In our scheme, a delegated entity is not required to perform any computation.

The security framework for Java-based computing environment in [112] uses roles in chained delegations to simplify the management of privileges. However, their delegations are made to individuals rather than to roles. The framework does not support tracing the delegation credentials of intermediate entities on the delegation chain, therefore does not support the verification of delegation chains. Their term cascaded delegation has different meanings from ours, and refers to delegations where all the privileges of preceding entities on the chain are inherited by the delegatee. In our model, only the specified privilege is delegated throughout a delegation chain.

Permission-based delegation model (PBDM) built on RBAC supports user-to-user,

role-to-role delegations [135]. A delegator creates one or more temporary delegation roles and assigns delegatees to particular roles. Delegations in PBDM requires changes of role hierarchies by the proper authority, for example, a project leader who has write access to the role assignment and access policies. PBDM does not address decentralized delegation, which is our main focus.

X-GTRBAC Admin [16] is an administration model for policy administration within a large enterprise. It specifies the user-to-role and permission-to-role assignments in the XML-based Generalized Temporal Role Based Access Control (X-GTRBAC) framework [80]. X-GTRBAC Admin supports decentralized administration by distributing assignment tasks to multiple domains within the enterprise while enforcing temporal constraints. In comparison, our RBCD models aim at the decentralized trust management among members of independent organizations. Therefore, X-GTRBAC Admin is complementary to RBCD models.

Shehab, Bertino, and Ghafoor recently propose a distributed secure interoperability framework for mediator-free collaboration environments [109]. They define secure access paths for dynamic collaboration environment, and also give a path authentication technique for proving path authenticity. Their idea of exploring trust paths in multi-domain environment is similar to the authentication of delegation chains in RBCD. The main difference of their work from ours is that they focus on the domain-level authentication, as opposed to authentication of individual role members.

2.9 Conclusions

We have studied cross-domain role-based delegation problem for information sharing where there is no central administrator. The main challenge addressed in this work is the verification of role-based authorization chains in decentralized environments, which has not been much studied in existing literatures. We have presented a role-based cascaded delegation model and its associated cryptographic operations for the purpose of convenient verification of delegation chains. RBCD enables a role member to create delegations based on the need of collaboration, yet in the meantime a delegation chain can be verified by anyone without the participation of role administrators. Our protocol is general and can be realized by any signature scheme.

We have described a specific realization with hierarchical certificate-based encryption scheme that gives delegation compact credentials.

Chapter 3

Anonymous Cascaded Authorization: Model and Protocol

A preliminary version of this paper was published in the Proceedings of the Seventh Annual IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW '06) [128].

3.1 Introduction

Authorization is an important concept of the resource sharing in open and collaborative environments such as Grid computing [98] or the Internet. In role-based trust management [89, 114], privileges are associated with roles and each user is assigned one or more roles. Role members prove their memberships with digital credentials and public-key signatures. Role-based delegation is important in decentralized role-based trust management for transferring privileges and sharing resources among role members that are initially unknown to each other. A delegation credential is a digital certificate signed by a delegator on a statement that gives authorizations to delegatees. In role-based delegation models [90, 114], a privilege can be delegated to another role, and then any member of the role can pass that privilege onto other roles. Besides privileges, a role, which represents a collection of privileges, can be delegated as well. We first illustrate in Example 3.1.1 a simple multi-step delegation scenario that transfers rights among roles within one administrative domain. Then we show

in Example 3.3.1 a more complex cross-domain role-based delegation.

Example 3.1.1 *A hospital has roles* Doctor*,* Nurse*, and* Intern*. The hospital permits all doctors to access a medical storage room. Bob is a doctor and has a doctor role credential issued by the hospital. When Bob is out of town, he authorizes his nurses to access the storage room by issuing the nurses a delegation credential. Alice is Bob's nurse and has a nurse role credential. She has short-term interns who also need to access the storage room. Then Alice passes the access privilege onto her interns by creating another delegation credential. The two-step delegation chain gives the authorization to interns to access the storage room, which consists of the two delegation credentials and Bob and Alice's role credentials. The role credentials show the delegators have the proper roles to issue the delegation.*

Decentralized delegation is to transfer privileges across different administrative domains, which is important to facilitate information and resource sharing in a collaboration. We give a more complex cross-domain role-based delegation in Section 3.3.

For privacy concerns, the identity of a user or an authorizer may be sensitive information in e-commerce, e-medicine, or peer-to-peer file-sharing (e.g., Kazaa) applications. An authorizer may not want to reveal his or her identity and role membership at each authorization or authentication. There has been a significant amount of work on trust negotiation frameworks [119, 132], whose aim is to strategically control the release of sensitive credentials to unknown parties. In addition, organizations may want to hide their internal structures from the outside world.

To address these privacy concerns, an *anonymous role-based delegation* protocol can be implemented with group signatures, in which a signature proves the membership of a signer without revealing the identity [40, 12]. The anonymous signing feature of group signatures is particularly suitable for role-based delegation, because what is essential for verifying a delegation credential is the proof of the delegator's role membership, rather than his or her identity. A role-based delegation protocol implemented using group signature schemes is not only scalable due to the use of roles, but also has strong privacy protection provided by the group signature schemes.

A practical concern about group signatures is their efficiency in a distributed environment. Next, we introduce the technique of aggregate signatures and explain

the need for a signature scheme that supports both anonymous signing and signature aggregation.

3.1.1 Credential size and aggregate signatures

Lengthy digital credentials are inefficient to transmit and store. In decentralized trust management systems [90, 114], a delegation chain represents how trust or a delegated privilege is transferred from one user to another. The chain contains a sequence of delegation credentials that connects unknown entities and resource owners. The number of credentials required to authenticate a delegation chain is linear in the length of the chain. Credentials associated with a delegation chain need to be compact, because mobile devices may have limited storage units and bandwidth.

Aggregate signatures [23, 92] are an effective solution for shortening credential size. Namely, multiple signatures on different messages can be aggregated into one signature of constant size. An interesting question is how to obtain an aggregate signature scheme that supports anonymous signing in role-based authorization. In on-line banking applications for example, certain transaction can be approved only if it is signed sequentially by a member of the role *cashier*, a member of the role *accountant*, and a member of the role *manager*. Each signature can be generated without disclosing the signer's identity for privacy protection, and then be aggregated to existing ones.

Existing group signatures do not support signature aggregation. In this work, we present an *anonymous-signer aggregate signature* scheme. that satisfies properties of unforgeability, anonymity, traceability, exculpability, unlinkability, collusion-resistance, correctness, and aggregation (See Section 3.4.3 for definitions). We achieve these properties by designing the signing key such that it is random, yet contains the long-term private key of a role member. Even a role manager cannot sign on behalf of a role member because the manager does not know the long-term private key of that user. We are able to achieve this by leveraging properties of a bilinear map, which was first used in the identity-based encryption scheme of Boneh and Franklin [22].

3.1.2 Our contributions

We present an anonymous-signer aggregate signature scheme. In our scheme, a role member u has a long-term public and private key pair. In addition, u computes a set of one-time secret signing keys from his private key. Then, the public information associated with these one-time signing keys are certified by the role manager. A role manager maintains the role by processing newly joined members and opening signatures (revoking the anonymity of signers) as necessary. The resulting certificates are (one-time) *signing permits*. To sign on behalf of a role, a member u first signs with one of the secret signing keys, then that signature is aggregated with the corresponding signing permit. This operation creates a *role signature* in which the signer is anonymous but can be proven to be a member of a role. We introduce a simple yet effective key blinding mechanism that integrates the long-term private key of a signer with a random blinding factor. Using this special signing key, a role member cannot deny a signature when revoked anonymity; yet, the role manager cannot misattribute a signature to any role member. By leveraging signature aggregation [23], the length of a role signature can be as short as 170 bits with security equivalent to a 1024-bit RSA signature. A role signature along with the public information needed for verification is only 510 bits or 64 bytes long. Role members can join and leave at any time, without requiring existing members to perform any update.

In an anonymous-signer aggregate signature scheme, individual role signatures that may be generated by members of different roles can be aggregated into one signature of constant length. Even if a signature is aggregated with other signatures, a role manager can trace the signer and show the proof. The security is based on the security of the aggregate signature scheme [23]. Because of one-time public keys, the asymptotic growth of our signatures is still linear in the number of individual signatures. Nevertheless, signature aggregation can significantly reduce the length of multiple signatures. A discussion on the efficiency of the scheme is given in Section 3.7.

We describe how anonymous-signer aggregate signatures can be used to realize an anonymous and efficient role-based authorization protocol, where a delegator issues delegation credentials and proves role membership without disclosing the identity. Although anonymous RBCD can be realized with any group signature scheme, using our

anonymous-signer aggregate signature scheme allows the compression of delegation credentials and significantly improves the efficiency. Delegation certificates in RBCD are issued to roles, rather than individual role members. For example, a privilege is delegated to the role *doctor* at a hospital.

Note that the RBCD protocol does *not* require a hierarchical generalization of our signature scheme, and does *not* require the (expensive) hierarchical certification of one-time signing keys.

Finally, we point out that anonymous role-based delegation implemented with anonymous-signer aggregate signatures gives rise to a proxy signature scheme for groups, which may be of separate interest. In this scheme, u delegates his signing power to a certain group G of proxy signers by issuing a delegation certificate. Each of the proxy signers can sign anonymously on behalf of u, provided that the proxy is a valid group member. The anonymity can be revoked by the manager of group G. The signature from a proxy signer needs to demonstrate the group membership of the proxy, and that group G is authorized by u. Note that u is not the manager of group G. Indeed, u can be anyone outside group G. Our proxy signature scheme for groups is scalable, and is particularly suitable for role-based systems [103]. For example, Central Bank needs to delegate the signing power to all members of role *clerk* at a local bank. To do this, Central Bank just needs to generate one proxy signature for the role *clerk*, instead of issuing one for each role member. The ability to aggregate multiple such proxy signatures into one make this scheme efficient in pervasive computing environment. We omit the details of our proxy signature scheme for groups here, as it can be easily derived from our anonymous RBCD protocol.

3.1.3 Organization

In Section 3.2, we give an overview of the aggregate signature by Boneh *et al* [23]. A cross-domain role-based delegation example is given in Section 3.3. The definition and construction of our anonymous-signer aggregate signature scheme are given in Section 3.4. We prove the security properties in Section 3.5. In Section 3.6, we introduce the anonymous role-based cascaded delegation protocol. The analysis of the anonymous role-based cascaded delegation protocol is given in Section 3.7. Related work is described in Section 3.8. Conclusions are given in Section 3.9.

3.2 Preliminaries

Here, we describe the aggregate signature scheme [23] that is used to construct our signature schemes. The aggregate signature scheme by Boneh, Gentry, Lynn, and Shacham (BGLS scheme for short) supports aggregation of multiple signatures on distinct messages from distinct users into one signature [23]. It uses bilinear maps [22] and works in any group where the decision Diffie-Hellman problem (DDH) is easy, but the computational Diffie-Hellman problem (CDH) is hard. Such groups are referred as gap groups [96] and are explained further in Section 3.4.1. The BGLS scheme comprises five algorithms: BGLS_KeyGen, BGLS_Sign, BGLS_Aggregate, BGLS_Verify, and BGLS_Agg-Verify. The first three algorithms are defined the same as in ordinary signature schemes; BGLS_Aggregate merges multiple signatures into one signature of constant length; BGLS_Agg-Verify verifies aggregate signatures.

Informally, the security of aggregate signature schemes is equivalent to the nonexistence of an adversary capable of existentially forging an aggregate signature [23]. Here, existential forgery means that the adversary attempts to forge an aggregate signature by some set of users, on messages of her choice. The formal proof of security defines an aggregate chosen-key security model, where the adversary is given a single public key, and her goal is the existential forgery of an aggregate signature. The adversary is given the power to choose all public keys except the challenge public key, and she is also given access to a signing oracle on the challenge key [23]. We refer readers to the paper of BGLS scheme [23] for further details.

Our anonymous-signer aggregate signature scheme is constructed based on the aggregate signature scheme [23]. We do not claim our scheme as a general group signature scheme, although it has the key properties of a group signature scheme. To distinguish from the naming conventions of group signatures, we use *role*, *role member*, *role manager*, and *role signature* in our scheme, which are equivalent to *group*, *group member*, *group manager*, and *group signature* in a group signature scheme, respectively. A role represents a number of individuals having certain attributes, each of them being a role member. The role is administrated by the role manager. A role signature is a signature signed by a role member on behalf of a role.

3.3 An Example of Cross-Domain Role-based Delegation

Example 3.3.1 is a multi-step role-based delegation that transfers rights among roles across different administrative domains in a collaboration. Example 3.3.1 is conceptually more complex than Example 3.1.1 in the introduction.

Example 3.3.1 [**Scenario**] *Suppose that a hospital has a collaborative project with members of the role* Staff *in a lab. The collaboration requires* Staff *to access certain resources (e.g., medical databases) at the hospital. Also suppose that the lab further subcontracts a part of the project to a company. This subcontract requires a role* Contractor *at the company to also access the resources at the hospital. Therefore, in this example, a two-step delegation is needed to transfer privileges first to the role* Staff *and then to the role* Contractor*. Note that there is no single administrative authority and the three organizations are autonomous.*

Suppose the priviledges (e.g., accessing medical databases) required in the project are all associated with the role guest at the hospital. Therefore, when the role guest is delegated to the role Staff*, all members of the role* Staff *at the lab are authorized the privileges associated with role guest and thus can access the required data. Furthermore, a member of role* Staff *needs to extend the role guest to members of role* Contractor*, so that the collaborators at the company can share the resources as well. The rights are transferred by delegation as follows.*

[**Delegation step 1**] *An administrator at the hospital delegates the role guest to the lab's role* Staff *in a credential C. This delegation means that a member of* Staff *at the lab is also a member of guest at hospital, and can access resources that are associated with the role guest. John is a member of the role* Staff *and has the corresponding role credential R. Therefore, John now is delegated the hospital's role guest.*

[**Delegation step 2**] *To transfer the access privileges associated with role guest to* Contractor *at the company, John (or any authorized* Staff *member) further delegates the role guest to the role* Contractor *in a credential C'. This delegation means that a member of role* Contractor *at the company is also a member of guest at the hospital.*

[**Delegation chain for** Contractor] *Credentials C, R, and C' constitute the delegation credential that authorizes the role* Contractor*. Note that the role credential R proves that John is indeed a member of* Staff *and thus is entitled to issue delegations.*

[**Accessing resource**] *When a member of* Contractor *at the company requests to access the shared resources at the hospital, he or she presents the delegation chain shown above along with the proof of* Contractor *membership.*

3.4 Anonymous-signer aggregate signature scheme

We present our anonymous-signer aggregate signature scheme. First, we list the number theoretic assumptions needed in our scheme, and then describe the algorithms.

3.4.1 Assumptions

Similar to the aggregate signature scheme [23], our anonymous-signer aggregate signature scheme uses bilinear maps and works in gap groups [26, 96], which is explained next. Let G_1 and G_2 be two cyclic groups of some large prime order q. We write G_1 additively and G_2 multiplicatively.

Computation Diffie-Hellman (CDH) Problem: Given a randomly chosen $P \in G_1$, aP, and bP (for unknown randomly chosen $a, b \in \mathbb{Z}_q$), compute abP.

Decision Diffie-Hellman (DDH) Problem: Given a randomly chosen $P \in G_1$, aP, bP, and cP (for unknown randomly chosen $a, b, c \in \mathbb{Z}_q$), decide whether $c = ab$. (If so, (P, aP, bP, cP) is called a valid Diffie-Hellman tuple.)

We call G_1 a gap group, if the DDH problem can be solved in polynomial time but no probabilistic algorithm can solve the CDH problem with non-negligible advantage within polynomial time. As observed in the aggregate signature scheme [23], general gap groups are insufficient for constructing efficient aggregate signatures, therefore our scheme also makes use of bilinear maps. We refer the readers to papers by Boneh and Franklin [22] for examples and discussions of groups that admit such pairings.

Reverse Computation Diffie-Hellman (RCDH) Problem: Given a randomly chosen $P \in G_1$, aP, and bP (for unknown randomly chosen $a, b \in \mathbb{Z}_q$), compute cP such that $aP = bcP$.

RCDH problem has been shown to be equivalent to CDH problem by Chen, Zhang, and Kim [42], which is shown briefly as follows for completeness. Suppose one can solve CDH problem in G_1 on (P, aP, bP), then one can obtain abP. Let $Q = bP$. Then $P = b^{-1}Q$, $aP = ab^{-1}Q$, and $abP = aQ$. This means that we can obtain aQ from $(Q, b^{-1}Q, ab^{-1}Q)$. Thus solves RCDH problem. Given (P, aP, bP), suppose one can solve RCDH problem in G_1. Then one can first obtain $b^{-1}P$ from (P, bP) because $P = bb^{-1}P$. Then we can solve RCDH problem on $(P, aP, b^{-1}P)$ to obtain abP, as $aP = (ab)b^{-1}P$. This means that we obtain abP and thus solve CDH problem.
Admissible pairings: Following Boneh and Franklin [22], we call $\hat{e}$ an admissible pairing if $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a map with the following properties:

1. Bilinear: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in G_1$ and all $a, b \in \mathbb{Z}$.

2. Non-degenerate: The map does not send all pairs in $G_1 \times G_1$ to the identity in G_2.

3. Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in G_1$.

Admissible pairing has been used to construct a number of encryption and signature schemes [22, 126], and most recently in a broadcast encryption scheme with short ciphertexts and private keys [25].

3.4.2 Operations

An anonymous-signer aggregate signature scheme consists of AA_Setup, AA_Join, AA_Sign, AA_Aggregate, AA_Verify, and AA_Open algorithms.

AA_Setup: On input a security parameter k, a probabilistic algorithm outputs a role public key P_{A_1}. Each entity (role manager and role member) also chooses his or her public/private keys.
AA_Join: A protocol is run between the role manager A_1 and a user that results in the user becoming a new role member. The outputs of the user are membership certificates and membership secrets.
AA_Sign: An algorithm takes as inputs a role public key, a membership secret, a membership certificate, and a message M_1, and outputs a role signature on M_1.

AA_Aggregate: This deterministic algorithm takes as inputs a number of role signatures and returns one aggregate signature S_{Agg}.
AA_Verify: An algorithm takes as inputs the role public keys $P_{A_1}, \ldots, P_{A_n}$, the aggregate signature S_{Agg}, and the messages $M_1, \ldots, M_n$. n is the number of signatures in the aggregation. P_{A_i} is the role public key of role manager A_i whose member signs message M_i in S_{Agg}, for $i \in [1, n]$. The output is 1 or 0.
AA_Open: The deterministic algorithm takes as inputs the message $M_1, \ldots, M_n$, the signature S_{Agg}, and role manager A_1's secret information to return the identity of the signer on message M_1.

A secure anonymous-signer aggregate signature scheme must satisfy the following properties. We give formal definitions for these security properties in the next section.

Correctness: Signatures produced by a role member using AA_Sign must be accepted by AA_Verify, and the AA_Open recovers the identity of the signer.
Unforgeability: Only valid role members can sign messages on behalf of the role. In particular, for an anonymous-signer aggregate signature S that is aggregated from n individual role signatures, even if an adversary knows $n - 1$ of them, she cannot successfully forge S.
Anonymity: Given a valid signature, it is computationally hard to identify the signer for anyone except the role manager.
Unlinkability: Deciding whether two different valid signatures were computed by the same role member is computationally hard for anyone except the role manager.
Traceability: The role manager is always able to open a valid signature and identify the signer.
Exculpability: Even if the role manager and members collude, they cannot sign on behalf of a non-involved member.
Coalition-resistance: A colluding subset of role members cannot produce a valid signature that the role manager cannot open.
Aggregation: Multiple signatures signed on different messages by different signers can be aggregated into one signature of constant length, and the aggregation can be performed by anyone.

We achieve the exculpability property because the manager cannot frame the member. A role member cannot deny his signature to the role manager because the

manager possesses a proof that binds the signature to the role member's long-term public key. The signature itself only serves as a partial proof. Only the role manager can revoke the anonymity and this can be done any time without any restriction.

3.4.3 Formal Definitions of Security Properties

Here, we give formal definitions in game models for properties of unforgeability, anonymity, traceability, and exculpability for an anonymous-signer aggregate signature scheme. We will show later in Section 3.5 that unlinkability and collusion-resistance are implied by our definitions. Our anonymity and traceability definitions follow the definitions by Bellare, Micciancio, and Warinschi who gave the first formal treatment of group signatures [12]. We do not give game-based security definitions for properties of correctness and aggregation as the definitions in the above section are sufficiently clear.

For unforgeability definition, the challenge key corresponds to the role public key. For anonymity definition, there are two challenge keys that correspond to two users' public keys, and we allow the adversary to adaptively choose both targets. We allow adversary to choose messages and query for opening signatures on the challenge public key(s). Similar to aggregate signature security [23], for a signature aggregated from n role signatures, the adversary is free to choose $n-1$ of the signing keys in all our definitions.

UNFORGEABILITY **Setup:** The challenger chooses a role manager's public key P_{A_1} by random and gives the adversary P_{A_1}. The challenger keeps the corresponding secret key. The challenger also gives the adversary public/private key pairs of all role members.

Join query: The adversary adaptively requests to join the role by asking for membership certificates of users of her choice. The challenger uses role manager's secret key to generate role certificates.

Hash query: The adversary requests the hash of a message of her choice.

Open query: The adversary requests to open anonymous-signer aggregate signatures of her choice.

Unforgeability response: The adversary outputs an anonymous-signer aggregate signature σ along with verification keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, and messages

$M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried and (2) all messages are distinct [1]. $P_{A_1}, \ldots, P_{A_n}$ correspond to the role public keys that are needed to verify the n signatures in the aggregation. The adversary breaks the unforgeability if σ can be verified.

ANONYMITY **Setup:** Same as in the unforgeability definition. In addition, the adversary chooses a message M at this phase.

Join, Hash, Open queries: Same as in the unforgeability definition.

Anonymity challenge: Once the adversary decides the query phase is over, she outputs two users' public keys P_u^0, P_u^1. The challenger picks a random bit $b \in \{0, 1\}$ and computes a challenge role signature ρ on M with the secret key corresponding to P_u^b. The adversary's task is to guess which user generates ρ.

The adversary can continue to submit more open queries on signatures other than ρ.

Anonymity response: The adversary outputs a guess b' and wins if $b' = b$.

TRACEABILITY **Setup:** The challenger gives the role manager's public key P_{A_1} to the adversary as in unforgeability definition. The challenger gives the public keys of all role members to the adversary.

Key extract query: The adversary obtains private keys of users in a set C of her choice.

Join, Hash, Open queries: Same as in the unforgeability definition.

The order of the above queries is up to the adversary.

Traceability response: The adversary outputs an anonymous-signer aggregate signature τ along with $P_{A_1}, \ldots, P_{A_n}$, and messages $M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried and (2) all messages are distinct. The adversary wins if τ can be verified and the signer associated with role manager P_{A_1} is opened to $\perp$ or to a user not in set C.

Compared to the "full-traceability" definition by Bellare, *et al* [12], our definition for traceability is weaker as we do not give the adversary the private key of the group manager. In our exculpability definition, an adversary is allowed to have the role manager's secret key, which means that we consider the case of a malicious role

[1] This restriction is inherited from the aggregate signature scheme (See explanation on page 6 of BGLS paper [23])

manager. The exculpability adversary is given the challenge that is the public key of a target role member.

EXCULPABILITY **Setup:** The challenger chooses a role manager's public key P_{A_1} and a challenge public key P_u by random. P_u is the public key of the target role member that the adversary needs to attack. Let s_{A_1} be the private key corresponding to P_{A_1}. P_{A_1}, s_{A_1}, and P_u are given to the adversary. The challenger keeps the private key of the target user. The challenger also gives the adversary the public/private keys of all the other role members.

Exculpability response: The adversary outputs an anonymous-signer aggregate signature ϕ along with $P_{A_1}, \ldots, P_{A_n}$, and messages $M_1, \ldots, M_n$. The restrictions are that (1) message M_1 has not been queried in the Sign queries and (2) all messages are distinct. The adversary wins if ϕ can be verified and the signer associated with role manager P_{A_1} is opened to the target role member with public key P_u.

Our exculpability definition is restrictive in that it does not allow an adversary to issue queries on the target. Ideally, an adversary may obtain from the challenger signatures of the target on messages of the adversary's choice. Note that the adversary can generate and open signatures of other role members on her own as she is given the private keys of the rest of group.

3.4.4 Construction

One can construct a naive aggregate group signature scheme from BGLS scheme [23] and one-time signing keys as follows. In a naive scheme, a group member generates a public/private key pair (PK, SK), by running the key generation algorithm of BGLS aggregate signature scheme. The group manager signs (with the group master secret) the public key, and sends the certificate [2] $Cert$back to the group member. To produce a signature on message M, the group member signs M with the private key SK to create signature Sig as in the aggregate signature scheme, and sends $(M, Sig, PK, Cert)$ to the verifier. Signature Sig can be aggregated with other signatures of this scheme as in BGLS aggregate signature scheme. However, the above scheme cannot prevent the group manager from misattributing signatures. The group manager first runs the

[2] This certificate is issued by a group manager for proving group membership of a member. It is different from a CA certificate, which certifies the validity of a public key.

key generation algorithm of BGLS aggregate signature scheme to obtain a key pairs (PK^*, SK^*). He signs public key PK^* using the group master secret and generates a certificate $Cert^*$ for PK^*. The group manager can then sign a message with private key SK^*, and misattribute the signature to *any* group member. The innocent group member does not have any proof that can be used to deny the signature.

We overcome the above problem by designing signing keys that are both unlinkable and tied to the *long-term* private key of a signer. In our protocol, a role member generates a one-time signing key based on both a long-term private key of and a short-term secret. The signing keys are then certified by the role manager. The long-term public key of the role member is certified by a Certificate Authority (CA), which serves as a trusted entity. Misattributing a signature to others is impossible, even for the manager, because a role member can prove that the signature does not correspond to his CA-certified public key. The underlying bilinear pairing allows us to achieve this property.

The notation of our anonymous-signer aggregate signature scheme is shown in Table 3.1. The last three items in Table 3.1 refer to the k-th signature in an (aggregate) signature aggregated from n ($k \leq n$) individual signatures.

Notation: For a role member u, s_u represents his private key, P_u represents his long-term public key, $K_{u,i}$ represents his i-th signing public key, and $X_{u,i}$ represents the corresponding i-th secret signing factor. For a role manager A, s_A represents his private key, P_A represents his long-term public key. $S_{u,i}$ is the i-th signing permit generated by the role manager for member u. When referring to an aggregate signature, K_{u_k,i_k} represents the signing public key associated with the k-th signature in the aggregate signature. Similarly, P_{u_k} and X_{u_k,i_k} represent the long-term public key and the secret signing factor of the k-signer in an aggregate signature, respectively. See also Table 3.1.

AA_Setup: This operation outputs the system parameters and public/private keys of users that will be used in the system.

- A trusted third party chooses a set of public parameters $params = (G_1, G_2, \hat{e}, \pi, H)$, where G_1, G_2 are groups of a large prime order q, G_1 is a gap group, $\hat{e} : G_1 \times G_1 \rightarrow G_2$ is a bilinear map, π is a generator of G_1, and $H : \{0,1\}^* \rightarrow G_1$ is a collision-resistant hash function, viewed as a random oracle [13].

u	A role member
s_u	Private key of u
P_u	Long-term public key of u
$K_{u,i}$	i-th signing key
$X_{u,i}$	i-th secret signing factor
$S_{u,i}$	i-th signing permit of u
A	A role manager
s_A	Private key of A
P_A	Long-term public key of A
K_{u_k,i_k}	The signing public key for the k-th signature
P_{u_k}	Long-term public key of k-th signer
X_{u_k,i_k}	Secret signing factor of k-th signer

Table 3.1: Notation for anonymous-signer aggregate signature scheme.

- Each role member chooses a secret s_u as his private key, and computes the product $s_u\pi$ as its public key P_u. Similarly, the role manager chooses his secret key s_A, and computes the public key $P_A = s_A\pi$. These are the *long-term* public keys, and are certified by a Certificate Authority (CA) using any secure signature scheme. The certification binds a long-term public key with its owner. The public key certificate of a member is used for repudiating misattributed signatures, and is different from the *one-time signing permits* below. The CA can be any trusted third party, but cannot be the same as the role manager. [3]

AA_Join: A role member u obtains one or more *one-time signing permits* from the role manager. Each permit certifies u's one-time signing key information, and is used for issuing role signatures. The following shows how the signing permits are generated.

- u randomly chooses l number of secrets $x_1, \ldots, x_l$. u computes one-time signing factors $X_{u,1} = x_1\pi, \ldots, X_{u,l} = x_l\pi$ and one-time signing public keys $K_{u,1} = s_u x_1\pi, \ldots, K_{u,l} = s_u x_l\pi$. Keys $P_u, X_{u,i}$, and $K_{u,i}$ are sent to the role manager in a secure channel [4], for all $i \in [1, l]$.

- The role manager tests if $e(K_{u,i}, \pi) = e(P_u, X_{u,i})$ for all $i \in [1, l]$. Recall $K_{u,i} = s_u x_i\pi$, $P_u = s_u\pi$, and $X_{u,i} = x_i\pi$. If the test fails, the protocol terminates.

[3] For simplicity, we assume that at a given time each user has only one long-term public key.

[4] $X_{u,i}$ needs to be kept secret because it can be used to identify the signer along with public information P_u and $K_{u,i}$.

The role manager makes sure that the one-time signing public keys submitted by u and on the manager's record are all unique. This check is necessary for the traceability requirement, otherwise colluding members can submit identical signing keys by manipulating their private keys and signing factors. Finally, the role manager runs BGLS_Sign on inputs s_A and strings *roleinfo* $\parallel K_{u,i}$ to obtain $S_{u,i} = s_A H(\textit{roleinfo} \parallel K_{u,i})$ for all $i \in [1, l]$. $S_{u,i}$ is the i-th *one-time signing permit* for u, and is given to u. The role manager adds tuples $(P_u, X_{u,i}, K_{u,i})$ to its record for all $i \in [1, l]$.

AA_Sign : A role member u first computes a signature S_u on a message M on behalf of the role, by running BGLS_Sign on inputs $s_u x_i$ and M, where $s_u x_i$ is one of his one-time signing secrets. Then, u calls algorithm BGLS_Aggregate to merge signature S_u with his one-time signing permit $S_{u,i}$ corresponding to the secret $s_u x_i$. This gives the role signature, which is returned with the public key P_A of the role manager and the key $K_{u,i}$. The details are as follows.

- u runs BGLS_Sign with secret key $s_u x_i$ and message M, and obtains a signature $S_u = s_u x_i H(M)$.
- u aggregates the signature S_u with one-time signing permit $S_{u,i}$ associated with secret $s_u x_i$. This is done by running BGLS_Aggregate, which returns a signature $S_g = S_u + S_{u,i}$. Recall that $S_{u,i} = s_A H(\textit{roleinfo} \parallel K_{u,i})$. S_g is output as the role signature. u also outputs public key P_A and one-time signing public key $K_{u,i}$.

AA_Aggregate: Same as in algorithm BGLS_Aggregate. It takes as inputs n number of role signatures S_{g_k} and corresponding values P_{A_k} and K_{u_k,i_k} for all $k \in [1, n]$. Set $S_{Agg} = \sum_{k=1}^{n} \mathrm{S}_{g_k}$. S_{Agg} is output as the anonymous-signer aggregate signature. The associated keys P_{A_k} and $K_{u_k,i_k} = s_{u_k} x_{i_k} \pi$ for all $k \in [1, n]$ are also returned.

Note that k-th role manager's public key P_{A_k} for $k \in [1, n]$ does not need to be the same. In other words, signatures from roles of different organizations can be aggregated.

AA_Verify: This algorithm calls algorithm BGLS_Agg-Verify with the following inputs: an anonymous-signer aggregate signature S_{Agg}, public key P_{A_k}, and one-time signing public key K_{u_k,i_k} for all $k \in [1, n]$. n is the number of signers on the authorization chain.

- For $1 \leq k \leq n$, compute the hash digest $H(M_k)$ of message M_k and $h_k = H(roleinfo_k \parallel K_{u_k,i_k})$ of the statement on one-time signing permit.
- S_{Agg} is accepted, if $\hat{e}(S_{Agg}, \pi) = \Pi_{k=1}^{n} \hat{e}(P_{A_k}, h_k)\hat{e}(K_{u_k,i_k}, H(M_k))$; rejected if otherwise.

The correctness of the verification algorithm in our anonymous-signer aggregate signature scheme is shown as follows.

$$\begin{aligned}
\hat{e}(S_{Agg}, \pi) &= \hat{e}(\sum_{k=1}^{n} S_{g_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{g_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{u_k} + S_{u_k,i_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(S_{u_k}, \pi)\hat{e}(S_{u_k,i_k}, \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(s_{u_k} x_{i_k} H(M_k), \pi)\hat{e}(s_{A_k} H(roleinfo_k \parallel K_{u_k,i_k}), \pi) \\
&= \Pi_{k=1}^{n} \hat{e}(H(M_k), s_{u_k} x_{i_k} \pi)\hat{e}(h_k, s_{A_k}\pi) \\
&= \Pi_{k=1}^{n} \hat{e}(H(M_k), K_{u_k,i_k})\hat{e}(h_k, P_{A_k})
\end{aligned}$$

Opening of the signature by the role manager correctly identifies the signer, which is shown next.

AA_Open: Given an anonymous-signer aggregate signature S_{Agg} and its associated public information including P_{A_k} and K_{u_k,i_k} for $k \in [1, n]$, a role manager first verifies signature S_{Agg}. If it is valid, a role manager can easily identify a role member's public key P_{u_k} from K_{u_k,i_k}, by consulting the role record. The role member cannot deny his signature because the role manager can provide a proof, i.e. by showing $\hat{e}(K_{u_k,i_k}, \pi) = \hat{e}(P_{u_k}, X_{u_k,i_k})$, that the signature is associated with the member's public key.

Theorem 3.4.1 *The communication complexity of* AA_Join *algorithm is* $O(l)$*, where* l *is the number of one-time signing keys certified. The computational complexity of the* AA_Verify *algorithm is* $O(n)$*, where* n *is the number of signatures aggregated.*

3.5 Security

We have shown that our anonymous-signer aggregate signature scheme satisfies the correctness requirement. It also clearly satisfies the aggregation property. In this section, we prove the security properties for our anonymous-signer aggregate signature scheme.

Theorem 3.5.1 *Our anonymous-signer aggregate signature scheme is as secure as the BGLS aggregate signature scheme against existential forgery attacks.*

Proof: There are three parties involved in this proof: a challenger, adversary $\mathcal{A}$, and adversary $\mathcal{B}$. If adversary $\mathcal{B}$ has a non-negligible advantage over the unforgeability of our anonymous-signer aggregate signature scheme, then $\mathcal{A}$ uses $\mathcal{B}$ to break the BGLS aggregate signature scheme by answering the challenge posed by the challenger. The challenger chooses a public key P_{A_1} by random and gives P_{A_1} to $\mathcal{A}$ as the BGLS challenge. The challenger keeps the corresponding secret key s_{A_1}. $\mathcal{A}$ then interacts with $\mathcal{B}$ as follows.

Setup: $\mathcal{A}$ gives $\mathcal{B}$ the challenge key P_{A_1}. $\mathcal{A}$ generates a set of public/private key pairs and gives them to $\mathcal{B}$ as the keys of all role members.

Join query: $\mathcal{A}$ answers $\mathcal{B}$'s join query by submitting it to the challenger as follows. Suppose $\mathcal{B}$'s query is to join a user with public key P_u, one-time signing factors X_u, and signing key K_u. $\mathcal{A}$ tests whether $e(P_u, X_u) = e(K_u, \pi)$ holds. If yes, then $\mathcal{A}$ asks the challenger to sign ($roleinfo \parallel K_u$) with secret s_{A_1}. $\mathcal{A}$ passes the signature to $\mathcal{B}$ as the signing permit. $\mathcal{A}$ also keeps the tuple (P_u, K_u, X_u) for the record.

Hash query: $\mathcal{A}$ simply uses a collision-resistant hash function to compute the hash of messages of $\mathcal{B}$'s choice.

Open query: $\mathcal{B}$ requests to open an anonymous-signer aggregate signature of her choice. $\mathcal{A}$ can easily identify the signer's P_u by looking up the signing key K_u in $\mathcal{A}$'s record.

Unforgeability response: $\mathcal{B}$ outputs an anonymous-signer aggregate signature σ along with verification keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, strings $roleinfo_1$, ..., $roleinfo_n$, and messages $M_1, \ldots, M_n$. $P_{A_1}, \ldots, P_{A_n}$ correspond to role public keys that are needed to verify the n signatures in the aggregation. $\mathcal{A}$ makes sure that

(1) signing key K_1 has not been queried, (2) *roleinfo*$_1 \parallel K_1$, ..., *roleinfo*$_n \parallel K_n$ are distinct, and (3) messages $M_1, \ldots, M_n$ are distinct. $P_{A_1}, \ldots, P_{A_n}$ correspond to the role public keys that are needed to verify the n signatures in the aggregation.

$\mathcal{A}$ passes σ to the challenger, along with keys $P_{A_1}, P_{A_2}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, and messages *roleinfo*$_1 \parallel K_1$, ..., *roleinfo*$_n \parallel K_n$, $M_1, \ldots, M_n$.

If $\mathcal{B}$ breaks the unforgeability, i.e., σ can be verified correctly, with advantage ϵ, then $\mathcal{A}$ breaks BGLS with advantage ϵ. □

Next we show that our signature scheme satisfies the anonymity property.

Theorem 3.5.2 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups preserves anonymity in the random oracle model.*

Proof: We first design a new game, called random-x game, that is secure based on randomness, i.e., the adversary's advantage over random guessing is negligible. Then we reduce the anonymity game to the random-x game, i.e., breaking anonymity means that breaking the random-x game. Random-x game is as follows.

RANDOM-x GAME

The challenger chooses public parameters: a gap group G_1 of prime order q, a generator π of G_1. The challenger also generates a set of public/private key pairs in the form of $P_u = s_u\pi$ and s_u. These public/private key pairs are given to the adversary.

Query phase: The adversary chooses two public keys P_u^0 and P_u^1, and gives them to the challenger. The challenger picks a random secret x, a random bit b, and computes $K_u^b = s_u^b x\pi$. The adversary is given K_u^b, $x\pi$, and b, so that she learns K_u^b is computed from s_u^b.

When the adversary decides the query phase is over, she outputs two public keys P_u^{*0} and P_u^{*1} to be challenged on.

Challenge phase: The challenger picks a random secret x, a random bit b, and computes $K_u^{*b} = s_u^{*b} x\pi$, where $P_u^{*b} = s_u^{*b}\pi$. The adversary is given K_u^{*b}. The adversary's task is to guess whether b is 0 or 1.

The adversary submits more queries. Finally, the adversary outputs her guess b', and wins if $b' = b$. Because x is randomly chosen, the adversary does not have the advantage over random guessing, thus has negligible advantage in the random-x game. Note that the adversary does not know $x\pi$ of her challenge.

Assume that an adversary $\mathcal{A}$ can break the anonymity of our anonymous-signer aggregate signature scheme. Then an adversary $\mathcal{B}$ can use $\mathcal{A}$ to gain non-negligible advantage in the random-x game as follows. We model hash function H as a random oracle.

Setup: $\mathcal{B}$ first obtains the public parameters from her challenger in the random-x game. $\mathcal{B}$ then chooses a role manager's secret key s_A by random and computes the public key $P_{A_1} = s_A\pi$. $\mathcal{B}$ gives $\mathcal{A}$ P_{A_1}, and keeps s_A. $\mathcal{B}$ also gives $\mathcal{A}$ the public/private key pairs of all role members that $\mathcal{B}$ obtains from her challenger. $\mathcal{A}$ outputs a message M that $\mathcal{A}$ wishes to be challenged on.

Join query: $\mathcal{A}$ adaptively requests to join the role by asking for membership certificates of users of her choice. $\mathcal{B}$ uses s_A to generate role certificates. B makes sure that one-time signing public keys on her record are all unique.

Hash query: For messages other than M, $\mathcal{B}$ picks a random value in $\mathbb{Z}_q$ and returns it as the hash value. For message M, $\mathcal{B}$ picks a random r and returns $r\pi$ as its hash.

Open query: $\mathcal{B}$ also answers $\mathcal{A}$'s requests of opening anonymous-signer aggregate signatures. $\mathcal{B}$ can do this because the signing keys and the associated (long-term) public keys are recorded during the join query phase.

Anonymity challenge: Once $\mathcal{A}$ decides the query phase is over, $\mathcal{A}$ outputs two targets' public keys P_u^{*0}, P_u^{*1} in the random-x game. Then $\mathcal{B}$ outputs these two keys to her challenger as her targets. $\mathcal{B}$'s challenger picks a random bit $b \in \{0, 1\}$ and a random x, and computes $K_u^{*b} = xP_u^{*b} = s_u^{*b}x\pi$. K_u^{*b} is given to $\mathcal{B}$ as the challenge (in the random-x game). Next, $\mathcal{B}$ needs to generate a challenge role signature ρ on M for $\mathcal{A}$ using K_u^{*b} as the signing key even though $\mathcal{B}$ does not know $s_u^{*b}x$. (The trick is in the hash of message M.) $\mathcal{B}$ computes signature $\rho = s_A H(\mathit{roleinfo} \parallel K_u^{*b}) + rK_u^{*b}$. ρ can be correctly verified: $e(\rho, \pi) = e(H(\mathit{roleinfo} \parallel K_u^{*b}), P_A)e(r\pi, K_u^{*b}) = e(H(\mathit{roleinfo} \parallel K_u^{*b}), P_A)e(H(M), K_u^{*b})$.

$\mathcal{B}$ answers more open queries from $\mathcal{A}$ on signatures other than ρ as before.

Anonymity response: $\mathcal{A}$ outputs a guess b'. $\mathcal{B}$ outputs b' as her guess in the random-x game. If $\mathcal{A}$ has non-negligible advantage ϵ in breaking the anonymity of our anonymous-signer aggregate signature scheme, then $\mathcal{B}$ has advantage ϵ in breaking the random-x game. □

Anonymity as defined in Section 3.4.3 naturally implies unlinkability. In fact, they are technically the same property as observed in [12]. We summarize the unlinkability property in the following corollary.

Corollary 1 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the unlinkability requirement in the random oracle model.*

Next we show the traceability of our signature scheme. Intuitively, traceability means that the role manager is always able to open a valid signature and identify the signer.

Theorem 3.5.3 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the traceability requirement in the random oracle model under the CDH assumption.*

Proof: We prove traceability by contradiction. We show that if an adversary has non-negligible advantage in the traceability game, then there is a contradiction.

Setup, **Join query**, **Hash query**, and **Open query:** The challenger performs as in the unforgeability proof for Theorem 3.5.1. In addition, in **Join query**, the challenger makes sure that the one-time signing public keys are all distinct. At the end of join queries, the challenger has recorded a list of tuples (P_u, K_u, X_u) for the users that have been queried.

Traceability response: The adversary outputs a signature τ along with keys $P_{A_1}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, strings $roleinfo_1$, ..., $roleinfo_n$, and messages $M_1, \ldots, M_n$. As defined, τ should satisfy the following restriction: (1) signing key K_1 has not been queried, (2) $roleinfo_1 \parallel K_1$, ..., $roleinfo_n \parallel K_n$ are distinct, and (3) messages $M_1, \ldots, M_n$ are distinct.

Suppose the adversary wins. This means that (i) τ can be verified and (ii) the signer associated with role manager P_{A_1} is identified to $\perp$ after opening τ. In our signature scheme, (ii) means that $e(K_1, \pi) \neq e(P_u, X_u)$ for all users on the challenger's record. However, (i) means that τ is correctly formed, in particular, (i) means that τ contains a valid signing permit (in the form of $s_{A_1} H(roleinfo_1 \parallel K_1)$) for signing public key K_1. We have shown in Theorem 3.5.1 that the signing permit cannot be forged, which implies that the adversary obtains it from the challenger. Thus the

challenger must have a unique tuple corresponding to K_1 on the record. We reach a contradiction. □

Next, we prove the exculpability of our signature scheme [5].

Theorem 3.5.4 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups satisfies the exculpability requirement in the random oracle model under the CDH assumption.*

Proof: There are three parties involved in this proof: a challenger, adversary $\mathcal{A}$, and adversary $\mathcal{B}$. If adversary $\mathcal{B}$ has a non-negligible advantage over our anonymous-signer aggregate signature scheme, then $\mathcal{A}$ uses $\mathcal{B}$ to solve the RCDH problem (defined Section 3.4.1).

Setup: $\mathcal{A}$'s challenger chooses s and x by random and gives $\mathcal{A}$ (π, $s\pi$, $sx\pi$) as the challenge. Denote $s\pi$ by P_u and $sx\pi$ by K_1. They correspond to the long-term public key and one-time signing public key of a role member u, respectively. Adversary $\mathcal{A}$ also chooses a role manager's private key s_{A_1} by random and computes the corresponding public key $P_{A_1} = s_{A_1}\pi$.

$\mathcal{A}$ gives $\mathcal{B}$ keys P_{A_1}, s_{A_1}, P_u, and K_1. $\mathcal{A}$ also gives $\mathcal{B}$ the public/private keys of all the other role members that $\mathcal{A}$ generates on her own. Adversary $\mathcal{B}$'s task is to use K_1 as one of the signing public keys to create an anonymous-signer aggregate signature misattributing role member u.

Exculpability response: Adversary $\mathcal{B}$ outputs an anonymous-signer aggregate signature ϕ along with $P_{A_1}, \ldots, P_{A_n}$, $K_1, \ldots, K_n$, and messages $M_1, \ldots, M_n$. The restriction is that all messages are distinct. If adversary $\mathcal{B}$ wins then ϕ can be verified and the signer associated with role manager P_{A_1} is opened to the target role member with public key P_u. The latter means that $\mathcal{B}$ can provide the proof X that satisfies $e(s\pi, X) = e(K_1, \pi)$. Because of the bilinearity of e, X must be $x\pi$. Therefore, $\mathcal{A}$ outputs X as her answer and solves the RCDH problem on $sx\pi$ and $x\pi$. □

Our definitions of unforgeability, anonymity, traceability, and exculpability allow the adversary to have the private keys of any number of users except the target(s), therefore our signature scheme is naturally collusion-resistant.

[5] Our traceability does not directly imply exculpability as in [12] because the traceability definition does not give the adversary the private key of the role manager.

Corollary 2 *Our anonymous-signer aggregate signature scheme from bilinear pairings in gap groups is collusion-resistant in the random oracle model under the CDH assumption.*

3.6 Anonymous role-based cascaded delegation protocol

In this section we first briefly introduce the role-based cascaded delegation (RBCD) protocol [114, 129]. A large amount of work has been done on trust management and distributed authorization systems [2, 8, 17, 43, 89]. Among them, role-based cascaded delegation [114] is an efficient role-based trust management model that supports administrator-free delegation. Administrator-free delegation allows an individual role member to issue delegations without the participation of a role manager. It enables flexible and dynamic authorization in a decentralized environment. Using predicates and constraints, it is also possible to restrict the scope of the delegation, e.g., prevent further delegation [129]. RBCD comprises four operations: INITIATE, EXTEND, PROVE, and VERIFY. INITIATE and EXTEND are used by a resource owner and an intermediate delegator, respectively, to delegate a privilege to a role. PROVE is used by a requester to produce a proof of a delegation chain that connects the resource owner with the requester. VERIFY decides whether the requester is granted the access based on the proof.

In RBCD [114], a delegation credential includes role membership certificates of each intermediate delegator, and delegation extension credentials that are proofs of delegation transactions signed by delegators. Credentials associated with a delegation chain are transmitted to delegated role members at each delegation transaction. Therefore, for a delegation chain of length n, the number of certificates required to verify the delegation path is $2n$. Here, we use our signature scheme to extend the original RBCD protocol to support the anonymity of delegators.

Next we define anonymous role-based cascaded delegation and then describe how it is realized using anonymous-signer aggregate signatures. Delegation credentials generated in our signature scheme are efficient to store and transmit, which is important in ubiquitous computing. Similar to definitions in the original RBCD protocol

[114], a privilege represents a role membership or a permission for an action such as accessing a database. Anonymous role-based cascaded delegation allows any role member to authorize on behalf of the role without disclosing the individual identity. Recall that a role defines a group of entities having certain qualifications. Role members are managed by a role manager, which is equivalent to a role manager in the anonymous-signer aggregate signature scheme.

3.6.1 Definitions

An anonymous role-based cascaded delegation protocol defines five operations: ARBCD_Initiate, ARBCD_Extend, ARBCD_Prove, ARBCD_Verify, and ARBCD_Open.

ARBCD_Initiate: Same as in RBCD protocol [114], this operation is run by a resource owner to delegate a privilege to a role. It initiates a delegation chain for the privilege. The delegation certificate is signed using the private key of the resource owner on a statement, which includes the delegated privilege, the name of the role, and the public key of the role manager.

ARBCD_Extend: This operation is similar to ARBCD_Initiate, but is run by an intermediate delegator u, who is a member of a role that is delegated a privilege by credential C. The goal is for u to generate a credential C' that extends the privilege to members of another role r. Delegation credential C' includes information of the delegated privilege, the name of role r, and the public key of role r's administrator. In addition, credential C' also contains the delegation credential C that u received, and the proof of u's role membership. C' does not disclose the identity of u.

Credential C' may simply be an accumulation of individual certificates. In comparison, our realization using anonymous-signer aggregate signatures is more efficient.

ARBCD_Prove: A requester u with role r produces a proof, which authenticates the delegation chain connecting the resource owner with u. This includes a proof of u's role membership without disclosing the identity, and the delegation credential that delegates the requested privilege to r.

ARBCD_Verify: This is performed by the resource owner to verify that a proof produced by a requester correctly authenticates the delegation chain of a privilege.

ARBCD_Open: Role manager revokes the anonymity of a delegator by examining signatures on a delegation credential. The identity of the delegator is returned.

3.6.2 Realization

We give an anonymous RBCD protocol using anonymous-signer aggregate signatures. Compared to the original RBCD protocol [114], a one-time signing secret key instead of the delegator's private key is used to sign a credential, and a one-time signing permit instead of a role credential is used to prove role membership.

ARBCD_Setup: A trusted third party runs AA_Setup to set up public parameters *params*, and individuals to choose and certify long-term keys. Then, AA_Join protocol is run between role members and the role manager to set up one-time signing permits. The role manager also keeps a record of signing key information.

ARBCD_Initiate: A resource owner runs the BGLS_Sign to sign a delegation statement that authorizes a certain privilege to a role r. The inputs to BGLS_Sign are the resource owner's private key and the delegation statement that includes the delegated privilege, the role name r, and the role manager's long-term public key. The output is a delegation credential for r.

ARBCD_Extend: Role r is delegated a privilege, and a member u of r wants to further delegate the privilege to a role r'. u first runs algorithm AA_Sign to generate a role signature for r'. The inputs to AA_Sign are u's one-time signing secret key $s_u x_i$, a delegation statement, and u's one-time signing permit S_i corresponding to $s_u x_i$. The delegation statement includes the following information: role name r', the long-term public key of r''s manager, delegated privilege, AA_Sign returns a role signature *Sig*. Then the public signing key $s_u x_i \pi$ is appended to the delegation statement. Finally, delegator u calls AA_Aggregate to aggregate *Sig* with the signature on the delegation credential issued to role r. The resulting aggregate signature S_{Agg} and delegation statements are given to members of role r' as the delegation credential.

ARBCD_Prove: A requester u of role r first runs AA_Sign that uses a one-time signing key to sign a random challenge message T chosen by verifier. The random challenge is to prevent replay attacks and ensure that u possesses the secret signing key. Then, u calls AA_Aggregate to merge the output signature with the signature on the delegation credential issued to role r. The outputs are returned.

ARBCD_Verify: AA_Verify is run to verify the aggregate signatures submitted by the requester against the delegation statements. The request is granted if the signature is accepted, and rejected if otherwise. Note that the delegation statements include the

following public keys required to verify the aggregate signature: (1) public signing keys of intermediate delegators, and (2) long-term public keys of role managers. (1) are for verifying the signatures created in ARBCD_Extend operations; and (2) are for verifying one-time signing permits of intermediate delegators. We assume that the verifier knows the public key of the resource owner, which is needed to verify the signature generated in ARBCD_Initiate.

ARBCD_Open: A role manager runs algorithm AA_Open with a delegation credential, a target signing key $s_u x_i \pi$, and the signing keys record. This returns the public key $s_u \pi$, which identifies the signer.

The security of the above protocol is directly based on the security of the anonymous-signer aggregate signature scheme. This implies that it is infeasible to forge any valid delegation credential even under collusion. The anonymous RBCD protocol satisfies traceability and exculpability requirements, i.e., a role manager can revoke the anonymity of a role member as an intermediate delegator, but cannot frame a role member. Our realization of anonymous RBCD supports delegator anonymity without affecting the performance. It has similar efficiency as in the original RBCD protocol [114]. The time required for signing and verification is the same as in the original RBCD protocol [114]. In anonymous RBCD, role managers need to sign multiple one-time signing permits for role members, which is not required in RBCD. Nevertheless, a single signature is quite fast (3.57 ms on a 1 GHz Pentium III, compared to 7.90 ms for a RSA signature with 1007-bit private key on the same machine [11]). As described in Section 3.1.2, the above protocol gives rise to a proxy signature scheme for groups. Details (definitions, construction, and proof of security) of the proxy signature scheme are omitted here.

3.7 Analysis

For a delegation chain of length n, a delegation credential using our anonymous-signer aggregate signatures can be twenty times shorter than the one using ACJT scheme [6], and five times shorter than the one generated in BBS scheme [20]. For a delegation chain of length twenty, the size of our credential is 1.4 KB, and the one in BBS scheme is 5.2 KB; for a 20 Kbits per second connection, our credential can be transmitted

within 0.5 seconds, and the one using BBS takes 2.1 seconds.

In the anonymous RBCD protocol, a delegation credential generated by INITIATE operation contains a signature, delegator's public key, delegatee's role, the public key of delegatee's role manager, and the delegated privilege. Similarly, we can derive the contents of a delegation credential after $n-1$ extensions. Assume a role or privilege name is expressed in 100 bits and let security requirement equivalent to 1024-bit RSA signature. Using ACJT group signatures, the size of a credential of length n is at least $10944n$ bits. Using BBS group signatures, the size is $2073n$ bits. Using our signature scheme, the size is $540n + 170$. The improvement in credential size is more significant as the length of delegation chain increases.

One-time keys. A major drawback of our anonymous-signer aggregate signature scheme is that signing keys and signing permits are not reusable. To reduce communications between the role manager and members, role members can obtain multiple signing permits $S_1, \ldots, S_n$ at a time, by asking the role manager to certify multiple signing keys $K_{u,1}, \ldots, K_{u,n}$. Similar concepts can be found in the trustee tokens scheme [81] and the secret handshakes protocol [9]. A role manager needs to keep a file for storing one-time signing public keys. However, this does not significantly affect his performance, even though the number of role members is large. For example, for a role that has 100,000 members who obtain 100 one-time signing keys each year for ten years, the total storage space for all the one-time signing keys takes about 6.4 GB and can be easily stored on hard disks. Although file I/O in general can be relatively slow, appending new keys to the file is done off-line and does not affect the performance of users. If a database is used to maintain the keys, operations such as searching a signing key can be very fast as the keys can be indexed.

Remark: Our anonymous RBCD protocol does *not* use the anonymous-signer aggregate signatures in a hierarchical fashion, where a role member in one organization is the role manager of another organization and so on. Instead, signatures to be aggregated are generated by role members belonging to *independent* roles (or organizations), and role members have their signing keys certified independently by their role managers.

Therefore, the anonymous RBCD model using our signature scheme supports

anonymity, exculpability (non-framing), and aggregation, without incurring significant overhead from the use of one-time signing keys. Note that there is a conceptual difference between the BGLS signature scheme and our signature scheme. In their aggregate signature scheme, a verifier is given a signature along with the identities of the parties involved and their respective messages. The verifier can obtain the public keys from CA, and thus in aggregate signatures, the size of public information is reduced. Instead, in our proposed scheme, the verifier can not obtain the one-time signing public keys from a certified directory.

Revocation Role membership revocation before the expiration can be handled by maintaining a revocation service, which can be efficiently achieved using authenticated dictionaries (e.g. [69, 94]). Authenticated dictionary is a system for distributing data and supporting authenticated responses to queries. One-time signing public keys of revoked members are put on the repository of revocation service by a role manager. Before verifying a role signature, the revocation service is queried to ensure that the signature is not generated by a revoked signing key.

Anonymity of Role Manager In our schemes, the public key of the role manager is required to verify role signatures, and therefore known to the public. Nevertheless, it does not mean that the role manager cannot sign messages anonymously. On the contrary, a role manager can run the protocols to certify a set of secret signing keys of his choice and use them as signing keys without disclosing the identity. Therefore, our schemes provide the same signing ability to every role member including the role manager.

3.8 Related Work

Our signature scheme has properties that are related to group signature schemes. Group signatures, introduced by Chaum and van Heijst [40], allow members of a group to sign messages anonymously on behalf of the group. Only a designated group manager is able to identify the group member who issued a given signature. Furthermore, it is computational hard to decide whether two different signatures are issued by the same member. In early group signature schemes [40], group public keys grew with the size of the group and were inefficient.

A group signature scheme with constant-sized public keys was first given in [31], and followed by a number of improvements [6, 20]. Until recently, group signature constructions (e.g., [6, 29]) were mostly based on the strong-RSA assumption, and a group signature typically comprised of multiple elements of RSA-signature length. Recently, bilinear pairing [22] has been used to construct group signature schemes [20, 30, 42], whose security is based on variants of Diffie-Hellman assumptions. The group signature scheme by Boneh, Boyen, and Shacham [20] significantly shortens the signature length, compared to the RSA-based state-of-the-art group signature scheme by Ateniese *et al.* [6]. An identity-based group signature scheme was proposed by Chen, Zhang, and Kim [42], where a group signature cannot be forged even if the private key of a user is known by a third party (i.e., the Private Key Generator in the ID-based systems [22]). Bellare, Micciancio, and Warinschi gave the first formal treatment of group signatures by introducing strong, formal definitions for the core requirements of anonymity and traceability [12]. They also developed a construction of a group signature scheme achieving the requirements based only on the existence of trapdoor permutations.

Most recently, Chase and Lysyanskaya gave an abstract construction of anonymous delegatable credentials based on their construction of signatures of knowledge [36]. Our anonymous role-based cascaded delegation can be implemented using their anonymous delegatable credential system, which allows one to issue delegation credential without revealing his or her identity and the delegation can be further extended to others anonymously. In comparison, we focus on the functionality of signature aggregation in addition to anonymous delegation in our construction.

There has been extensive research on access control models in the past decade [55, 102, 103]. The concept of role-based access control [55, 103] is widely deployed to improve the scalability and efficiency of management. Trust management models are developed for the authorization in distributed systems. A number of such systems have been proposed, for example KeyNote [17], delegation certificates [8], SPKI [43], Delegation Logic (DL) [88], proof-carrying authorization (PCA) [1], *RT* framework [89], and role-based cascaded delegation [114]. Our anonymous role-based delegation protocol and implementation are privacy-enhancing techniques general for any role-based trust management systems.

Hidden credentials system [76] has been proposed to protect sensitive credentials and policies. The main idea of that paper is that when a signature derived from an identity based encryption scheme (IBE) [21, 44, 107] is used to sign a credential, the credential content can be used as a public encryption key such that the signature is the corresponding decryption key. Hidden credentials can be used in such a way that they are never shown to anyone, thus the sensitive credentials are protected. The Hidden Credentials system also protects sensitive policies by not specifying which credentials can be used to decrypt the encrypted resource. Bradshaw *et al.* [27] extended the hidden credentials system to support complex access policies expressed as monotonic Boolean functions. They applied a secret splitting system to conceal the structure of such policies. The extended hidden credentials system protects the structure of Bob's polices. Frikken *et al.* [58] give a scheme that hides both credentials and policies. Most recently, Frikken *et al.* [59] proposed a protocol that allows the client and the servers to have policies for their credentials (to mitigate probing attacks) and hide these policies and the credentials. The above-mentioned work is in the conventional access control settings, where the server and authorized clients have established trust when hidden credentials are issued.

Anonymous credential systems have been developed [28, 32, 37, 39] to allow anonymous, yet authenticated and accountable, transactions between users and service providers. These systems give a technique for protecting the users' privacy when conducting Internet transactions. Our work is for anonymous role-based authorization, and can potentially be used as an anonymous credential system, where a user authenticates herself to be a valid member of a role. This can be achieved by generating a role signature, which is verified by a resource owner (verifier). The disadvantage of such an anonymous credential system compared to the state-of-the-art is that the credential is only one-time use instead of multi-use.

3.9 Conclusion

We have proposed an anonymous role-based cascaded delegation (RBCD) protocol that protects sensitive role-membership information of delegators. Although the anonymous RBCD model can use any group signature scheme to realize, we have

shown that there is a performance advantage using our anonymous-signer aggregate signature scheme. Anonymous-signer aggregate signature scheme is suitable for sensitive applications where a large number of signatures are produced and the role or group membership of a signer (instead of the identity of the signer) is needed for verification.

Chapter 4

Point-Based Trust

A partial and preliminary version of this paper won the Best Student Paper Award in the Eighth International Conference on Information and Communications Security (ICICS) 2006 [127].

4.1 Introduction

Access decisions in stand-alone systems are usually based on the identity of the entity requesting a resource, whereas in open systems such as grid computing and Internet, this approach becomes ineffective. This is because the resource owner and the client (or requester) typically belong to different security domains controlled by different authorities and are unknown to each other. The modern alternative is to use *digital credentials* for satisfying access policies. Digital credentials are digitally signed assertions about the credential owner by a credential issuer. Each digital credential contains a set of attributes about the owner. The decision to access a resource is based on the attributes in the client's credentials, such as age, citizenship, employment, group membership, or credit status.

A typical scenario for accessing a resource using digital credentials is for the client, Alice, to send her request to Bob, who responds with the policy that governs access to that resource. If Alice's credentials satisfy Bob's policy, she sends the appropriate credentials to Bob. After Bob receives the credentials and verifies them, he grants Alice access to the resource. Observe that, in this scenario, Alice learns Bob's policy

and Bob learns Alice's credentials. However, this mechanism is unacceptable if the credentials or the access control policies are considered to be sensitive information.

The motivation for hiding credentials is individual privacy, e.g., if the credentials are about one's physical impairment or disability, financial distress, political or religious affiliation, etc. The motivation for hiding the policy is not only security from an evil adversary, but simply the desire to prevent legitimate users from *gaming* the system — e.g., changing their behavior based on their knowledge of the policy (which usually renders an economically-motivated policy less effective). This is particularly important for policies that are not incentive-compatible in economic terms (they suffer from perverse incentives in that they reward the wrong kinds of behavior, such as free-loading). In yet other examples, the policy is simply a commercial secret — e.g., Bob has pioneered a novel way of doing business, and knowledge of the policy would compromise Bob's strategy and invite unwelcome imitators.

It is also important to point out that a process that treats Alice's credentials as confidential is ultimately not only to Alice's advantage but also to Bob's: Bob can worry less about rogue insiders in his organization illicitly leaking (or selling) Alice's private information, and may even lower his liability insurance rates as a result of this. Privacy-preservation is a win-win proposition, one that is appealing even if Alice and Bob are honest and trustworthy entities. We give a trust management model that quantitatively addresses degrees of sensitivity. Moreover, the degree of sensitivity of a given credential is private to each user, and can vary from one user to another.

4.1.1 Overview of Point-Based Trust Management

Quantitatively addressing trust establishment problem has existed in several papers on trust and reputation models [15, 47, 130, 136]. These models have applications in open systems such as mobile ad hoc networks, Peer-to-Peer networks [47], and e-trade systems.

We consider a new quantitative trust management policy that is private and should therefore not be revealed to Alice: Bob associates a number of points with every possible credential. Intuitively, the point value of a credential represents the trustworthiness of its holder. Bob also requires the sum of the points of those credentials that Alice uses to reach a minimum threshold before he grants her access to the

resource. The resource owner, Bob, defines an admissible threshold, and that threshold is itself private and should not be revealed to Alice. Alice needs to satisfy the threshold requirement to gain access by using a subset of her credentials that gives her the required number of points, but there can be many such subsets: Alice is interested in using the subset that has minimum privacy-value to her, according to her privacy-valuation function; that valuation function is itself private and should not be revealed to Bob. We give a protocol which determines which subset of Alice's credentials *optimally* satisfies Bob's threshold, i.e., it has minimum privacy value to Alice among all subsets that satisfy Bob's threshold. Bob's point-valuation of credentials, his thresholds, and Alice's privacy-valuation of her credentials are private and not revealed.

One of our main contributions is the formulation of the point-based trust management model. This model prevents premature information leakage (See more discussion in Section 4.2) during trust establishment, and gives the privacy protection to the client even when the trust establishment is unsuccessful, which may happen if the client thinks the total privacy scores of to-be disclosed credentials is too high.

4.1.2 Our Contributions

1. We propose a point-based trust management model and we formalize the credential selection problem of the model into a knapsack problem. Our point-based trust management model enables users to quantitatively distinguish the sensitivities of different credentials. It also allows a provider to quantitatively assign values to credentials held by clients. The point-based model has several features: **(i)** Policy specification is simple and easily allows dynamic adjustment of services provided based on released credentials; **(ii)** A user can proactively decide whether the potential privacy loss is worth the service without disclosing any sensitive information; **(iii)** To satisfy a policy, a user can select to disclose the *optimal* credential set that minimizes the privacy loss, based on his or her personal measure.

2. We give secure and private dynamic programming protocols for solving the knapsack problem. Our solution, consisting of a basic protocol and an improved protocol, allows the server and user to jointly compute the optimal sum

of privacy scores for the released credentials, without revealing their private parameters. The complexity of our basic protocol is $O(nT')$, where n is the total number of credentials and T' is the (private) *marginal threshold*, which corresponds to the sum of the points of the credentials that are *not* disclosed. The protocol uses homomorphic encryptions, and is semantically secure against semi-honest adversaries.

Our improved protocol, the *fingerprint protocol*, is secure in an adversarial model that is stronger than a semi-honest one (a.k.a honest-but-curious). The improved protocol prevents a participant from tampering with the values used in the dynamic programming computation. That is, while we cannot prevent a participant from lying about her input, we can force *consistency in lying* by preventing capricious use of different inputs during the crucial solution-traceback phase. The complexity of our fingerprint protocol is $O(n^2T')$.

3. One of our contributions that goes beyond the specific problem considered is a general *indexing expansion* method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants. Using this method, a participant is not required to trust the other party during the back-tracing phase. This is possible because the participant is able to efficiently identify whether the other party has tampered with the computation. For traceback in general dynamic programming problems, our algorithm not only allows a participant to independently and easily recover the optimal traceback solution, once the computed optimal value is given, but also enables the participants to verify the integrity of the optimal value.

Organization. In Section 4.2, we give an analysis on the properties of trust negotiation protocol that is the state-of-the-art solution for privacy-preserving authorization. Our point-based trust management model is presented in Section 4.3. Applications of our quantitative authorization model are described in Section 4.3.3. The basic protocol for privacy-preserving credential selection is given in Section 4.4. Fingerprint protocol is given in Section 4.5. We analyze the security in Section 4.6. We present an extension to the fingerprint protocol in Section 4.7. Related work is given in Section 4.8. Conclusions are given in Section 4.9.

4.2 An Analysis on Trust Negotiation

In a probing attack, Alice can engage in a protocol with Bob multiple times using different credential sets each time (all of which are subsets of her credentials) to gain information about Bob's policy. In the case where Alice is requesting access to a service, Bob will know whether she got access and can therefore also probe (by using different policies and observing their effect) to gain information about Alice's credentials.

One way of mitigating probing attacks is the one followed in the trust negotiation literature [19, 120, 121, 133, 134], in which the disclosure of a credential is governed by an access control policy that specifies the prerequisite conditions that must be satisfied in order for that credential to be disclosed. Typically, the prerequisite conditions are a subset of the set of all credentials, and the policies are modeled using propositional formulas. A trust negotiation protocol is normally initiated by a client requesting a service or a resource from a server, and the negotiation consists of a sequence of credential exchanges: Trust is established if the initially requested service or resource is granted and all policies for disclosed credentials are satisfied [121, 132].

The sequence of credentials disclosed during this negotiation depends on which strategy each negotiator uses for controlling which credentials are disclosed and when to disclose them, and when to terminate a negotiation [134]. Several negotiation strategies are proposed in [121, 132, 134]. For example, in the eager strategy [121], two negotiators take turns disclosing a credential to the other side as soon as the access control policy for that credential is satisfied.

Although mitigating probing attacks, the requirements of the trust negotiation literature have some practical limitations. **(1)** Probing is still possible when policies are not treated as sensitive resources, and the client (or server) can game the system in many ways. For example, if the client knows the access control policies for the server's credentials then she will know the path of least resistance to unlock certain credentials. **(2)** Premature information leaking is difficult to prevent in existing trust negotiation protocols including the recent framework using cryptographic credentials [87]. The premature information leaking refers to the situation when a negotiation is not successful, however sensitive credentials are already disclosed. **(3)**

The service model in trust negotiation is usually limited, that is, the requested service is fixed and independent of the amount of information released by the client at the end of the negotiation session. However, a client may end up disclosing more information than what is required for the initially requested service. The reward or service provided by the server should be dynamically adjustable with the amount of information released from the client.

In addition, an important notion of *cumulative privacy* cannot be efficiently supported by trust negotiation protocols. In existing solutions, revealing a credential has little to do with the credentials that have already been revealed. For example, Alice may consider her birthday or her birth-town to be insensitive, but consider the combination of them to be sensitive, as it would isolate her into a small group of people. Cumulative privacy is a desirable notion of privacy. It represents the cumulative sensitivity of a combination of credentials, which may or may not be sensitive themselves. Supporting cumulative privacy in trust negotiation is not efficient, as it requires the specifications of an exponential number of policies. As will become clear soon, our model and protocols presented in this work provide an alternative solution for trust establishment that improves on the security and flexibility of existing approaches.

The approach in [59] builds on this work, in that none of the credentials or policies is disclosed: A credential that became "disclosable" during the negotiation, becomes merely usable in [59] (not disclosable – although the very fact of using it also exposes it to possible probing, this is minimized as in the previous trust negotiation literature). As before, in [59] the client and server will each input a set of credentials along with an access control policy for each of their respective credentials. Protocols are given in [59] for determining the set of usable credentials (i.e., the credentials whose access control policies are satisfied by the other party) between the client and the server, and then for processing the resource (or service) request based on the client's usable credentials. This is done while satisfying the requirements that: (1) the policies for sensitive credentials may themselves be sensitive and therefore cannot be revealed, (2) the client should not learn information about which of her credentials or the server's credentials are usable, and (3) the server should not learn information about which of his credentials or the client's credentials are usable (if the client or server were to learn which of its credentials are usable, then this would reveal more information

about the other party's credential set and thus facilitate probing attacks). Note that, although mitigated, probing is still possible, e.g., by activating some of the server's credentials and not others.

As will become clear soon, our approach presented in this work mitigates the above-mentioned problems. The computation for determining whether a user satisfies a policy is privacy-preserving, where *neither* party needs to disclose sensitive information. Of the multiple ways of satisfying the policy, Alice will tend to use the one that utilizes the credentials whose privacy she values least.

4.3 Model

In this section, we describe a point-based trust management model, and define the credential selection problem in this model.

4.3.1 Point-Based Trust Management

In the point-based trust management model, the authorization policies of a resource owner defines an *access threshold* for each of its resources. The threshold is the minimum amount of points required for a requester to access that resource. For example, accessing a medical database requires fifty points. The resource owner also defines a *point value* for each type of credentials, which denotes the number of points or credits a requester obtains if a type of credential is disclosed. For example, a valid ACM membership is worth ten points. This means that a client can disclose his or her ACM membership credential in exchange for ten points. We call this a trust management model as opposed to an access control model, because the resource owner does not know the identities or role assignments of requesters *a priori*.

A requester has a set of credentials, and some of which may be considered sensitive and cannot be disclosed to everyone. However, in order to access a certain resource, the requester has to disclose a number of credentials such that the access threshold is met by the disclosed credentials. Different clients have different perspective on the sensitivity of their credentials, even though the credentials are of the same type. For example, a teenager may consider age information insensitive, whereas a middle-aged person may not be very willing to tell his or her age.

Therefore, in point-based trust management model, each client defines a *privacy score* for each of their credentials. The privacy score represents the inverse of the willingness to disclose a credential. For example, Alice may give privacy score 10 to her college ID, and 50 to her credit card. The client is granted access to a certain resource if the access threshold is met and all of the disclosed credentials are valid. Otherwise, the access is denied. From the requester's point of view, the central question is how to fulfill the access threshold while disclosing the *least* amount of sensitive information. In the next section, we define this as a credential selection problem. The credential selection problem is challenging, because the requester considers his or her privacy scores sensitive, and the server considers its point values and access threshold sensitive.

Typing of Points. One way to improve the expressiveness of point-based trust management is to support the *typing of points*. For example, financial point-type represents credit card and bank account, and demographic point-type represents birth date, address, and affiliation. In the on-line shopping scenario, a conventional policy defined by the server requires the client to disclose valid demographic information that is either an email address or a home address, and valid financial information that is either a credit card number or a bank checking account number, i.e., (email address $\vee$ home address) $\wedge$ (credit card $\vee$ bank account).

One way to translate this policy to points and thresholds in point-based trust management is as follows. The server specifies equal point values (e.g., 20) for the email address and the home address, and equal point values (e.g., 40) for the credit card number and the bank account. The threshold for *demographic point-type* is 20 and for the *financial point-type* is 40. In general, the number of options for the client to disclose private information may be large. For example, the client can disclose a certain combination of home phone/address, work phone/address, email address, fax number, etc. With this typing mechanism, the server can improve the expressiveness of point values, and the client can choose the optimal subset of information to release for each point-type. To support typing, the credential selection protocol (presented later) needs to be run multiple times, twice in this example. The translation between point-based policies and Boolean policies is an interesting research topic, and is subject to our future study.

Where do point values come from? One approach to obtain point values is from reputation systems [15, 116, 136]. Essentially the point value of a credential represents the trustworthiness of the organization that issues the credential. If a resource owner thinks organization A is more reputable than organization B, the resource owner specifies a higher point value for a credential issued by A than the one issued by B. This idea has been explored in a recent paper that quantitatively studies the connections between computational trust/reputation models with point values in point-based trust management. The paper also discusses the application of such models in privacy-preserving location systems. The work in trust models and reputation systems [15, 116, 136] serve as a starting point for demonstrating the applicability of point-based trust management.

4.3.2 Credential Selection Problem

Definition 4.3.1 *The credential selection problem is to determine an optimal combination of requester's credentials to disclose to the resource owner, such that the minimal amount of sensitive information is disclosed and the access threshold of the requested resource is satisfied by the disclosed credentials.*

We formalize the credential selection problem as an optimization problem. A credential selection algorithm answers the question what combination of credentials are to be released given the access control policies and privacy scores to each credentials. Our model assumes that the resource owner (or server) and the requester (or client) agree on a set of credential types as the universe of credentials $(C_1, \ldots, C_n)$. We define a binary vector $(x_1, \ldots, x_n)$ as the unknown variable to be computed, where x_i is 1 if credential C_i is selected, and 0 if otherwise. Integer $a_i \geq 0$ is the *privacy score* of credential C_i. It is assigned by the requester *a priori*. If the requester does not have a certain credential C_i, the privacy score a_i for that credential can be set to a large integer. Thus, the (knapsack) algorithm avoids choosing that credential type, as the cost is high. The server defines T that is the *access threshold* of the requested resource. Integer $p_i \geq 0$ is the *point value* for releasing credential type C_i. The requester considers all of a_i values sensitive, and the server considers the access threshold T and all of p_i values sensitive.

The credential selection problem is for the requester to compute a binary vector $(x_1, \ldots, x_n)$ such that the sum of points $\sum_{i=1}^{n} x_i p_i$ satisfies T, and the sum of privacy scores $\sum_{i=1}^{n} x_i a_i$ is minimized. This is captured in the following minimization problem. Compute a binary vector $(x_1, \ldots, x_n)$ such that the following holds:

$$\begin{aligned} \min \quad & \sum_{i=1}^{n} x_i a_i \\ \text{subject to} \quad & \sum_{i=1}^{n} x_i p_i \geq T \end{aligned}$$

The above minimization problem can be rewritten into a knapsack problem with a new variable $y_i = 1 - x_i \in \{0, 1\}$. For i-th credential, $y_i = 1$ represents not disclosing the credential, and $y_i = 0$ represents disclosing the credential.

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} y_i a_i \\ \text{subject to} \quad & \sum_{i=1}^{n} y_i p_i < \sum_{i=1}^{n} p_i - T \end{aligned}$$

We define the marginal threshold T', which coarsely correlates to the sum of the points of the credentials that are not disclosed.

Definition 4.3.2 *The marginal threshold T' of the credential selection problem is defined as $\sum_{i=1}^{n} p_i - T$, where p_i is the point value for credential type C_i, T is the access threshold for a requested resource, and n is the total number of credential types.*

Let us first review the dynamic programming solution for the 0/1 knapsack problem [45]. Then, we describe our protocol for carrying out private dynamic programming computation of the knapsack problem. The 0/1 knapsack problem is defined as follows. Given items of different integer values and weights, find the most valuable set of items that fit in a knapsack of fixed integer capacity. The dynamic programming solution is pseudo-polynomial: the running time is in $O(nT')$.

In the dynamic programming of knapsack problem, a table is made to track the optimal selection of items so far. A column indicates the range of values, which corresponds to the target weight of the knapsack. A row corresponds to each item.

The last table entry has the maximum capacity of the knapsack. The first column and the first row are initialized to zeros, i.e. $M_{0,j}$ and $M_{i,0}$ are zeros, for all $i \in [1, n]$ and $j \in [0, T']$. The table is filled from top to bottom and from left to right. Using the notations defined earlier, the recurrence relation is formally defined as follows. Denote $M_{i,j}$ as the value at i-th row and j-th column, and $i \in [0, n], j \in [0, T']$.

$$M_{i,j} = \begin{cases} M_{i-1,j} & \text{if } j < p_i \\ \max\{M_{i-1,j}, M_{i-1,j-p_i} + a_i\} & \text{if } j \geq p_i \end{cases}$$

Each entry of the table stores the total value of a knapsack, which is determined as either the value of a knapsack without the current item (expressed as the value directly to the top of the current entry), or the value of the knapsack with the current item added into it. At the end of the computation, the entry at the lower right corner of the table contains the optimal value of the knapsack. The selections of items can be obtained by bookkeeping the information of where the value of an entry comes from.

For our credential selection problem, the above recurrence relation can be interpreted as follows. If the point value of credential type C_i exceeds j, which is a value in the range of $[0, T']$, then the i-th credential is not selected and the privacy score $M_{i,j}$ is kept the same as $M_{i-1,j}$. Otherwise, the algorithm compares the score $M_{i-1,j}$ for not selecting the i-th credential with the score $M_{i-1,j-p_i} + a_i$ for selecting the i-th credential. The larger value is chosen to be the privacy score $M_{i,j}$.

The standard dynamic programming computation requires values a_i and p_i for all $i \in [1, n]$. However, in our model, the requester considers a_i sensitive, and the server considers p_i sensitive. We present a protocol that allows the completion of the dynamic programming computation without revealing any sensitive information. In addition to protecting sensitive a_i and p_i values, the entries in the dynamic programming table are also protected from both parties. From this perspective, our protocol provides better privacy protection than the secure multi-agent dynamic programming work by Yokoo and Suzuki [131], as their approach cannot prevent the disclosure of table entries.

What happens after the optimal subset is computed is flexible: If the privacy score of that subset is low enough then the client may choose to simply reveal those credentials to the server. The server verifies the validity of the credentials by checking

the credential issuers' signatures. However, if the score is high then the client may decide to abort the session with this server. Alternatively, the client may choose to then engage in an existing protocol (such as [76, 27, 58, 59]) in which she proves to the server that she has such a subset that satisfies the points-threshold requirement but without revealing the elements of that subset. Note that existing privacy-preserving protocols [76, 27, 58, 59] do not allow the finding of the *optimal* disclosure credential set.

Privacy score of a credential set. In the current model, the privacy score of multiple credentials is the sum of each individual privacy score. The summation is simple to model, and represents the additive characteristic of privacy, i.e., the more personal information revealed, the more privacy lost. Another advantage of the summation of privacy scores is the efficiency; the specification of privacy scores has a size linear in the number of credentials. However, the client may want to explicitly specify an arbitrary privacy score of a certain group of sensitive credentials. The group privacy score may be higher or lower than the sum of individual privacy scores. The latter case can happen when one credential might subsume or include some information that is included in the other credential(s). However, the dynamic programming solution is not clear for the dynamic programming problem with arbitrary constraints. It remains an interesting open question how to formulate the dynamic programming to support arbitrary privacy score specifications.

Our system is not a mere fine-grain privacy-preserving generalization of Multi-Level Security (MLS)[104] because of its additive nature, e.g., in an MLS-like scheme it is not possible to aggregate a number of lower-level clearances to access higher-clearance documents. However, our system subsumes MLS in that it can simulate it through a judicious assignment of quantitative values (e.g., by assigning large enough values to higher levels that they are unreachable through any additive aggregation of lower credentials).

4.3.3 Applications of Quantitative Authorization Policies

Using our point-based authorization policies, a resource owner or service provider can quantitatively adjust service quality or quantity based on the client's qualification. In our model, credentials are mapped with point values defined by the resource

owner, therefore the client's reward or service can be dynamically adjusted according to the amount of private information revealed. The flexibility makes the point-based model attractive to the trust management in web-services and e-commerce applications in general, as users have the incentives to carry on the computation for trust establishment, which facilitates business transactions. The more private information revealed, the better is the service provided to the client. Because the client is properly compensated for the sensitive information revealed (e.g., membership, affiliation, demographic data)

A concrete application for point-based model is privacy-aware presence systems [73, 124, 130], where presence data such as the location of a user is collected through devices such as GPS on a cellphone. The management of presence data is crucial, because it concerns not only user privacy, but also safety: presence data can be used to track and profile individuals. In the meantime, there may be emergency situations or extenuating circumstances when certain parties (like emergency workers) should have access to this kind of information, and friends and relatives of a user might be allowed to query his or her location information at any time. Therefore, a desirable feature of a location query system is that it provides different levels of precision based on the requester's trustworthiness or the context of the query. This requires a flexible authorization model for accessing the private location data, which can be offered by the point-based authorization model.

For example, Bob can use point-based authorization model to share and control his presence information, even with strangers who are unknown to Bob. Bob defines his own point-based policies that include the point values of credentials that are acceptable for authentication by him. Bob also defines the precision of his location associated with certain point values. For example, if the point value of the query issuer is twenty, then Bob might release his location information exactly. If the point value is five, then he might release a *fuzzy interpretation* of his location, e.g., just the building or street name of where he currently is. Phrased more concretely, suppose Alice who is unknown to Bob wants to know Bob's location; if she reveals her valid driver's licence to Bob, a precise answer can be returned as the driver's licence is highly trustworthy credential. However, if Alice reveals only a Yahoo! email account, then nothing about Bob's whereabout may be disclosed because Yahoo! email account

does not prove Alice's trustworthiness to Bob. Suppose Alice reveals her university ID, then Bob may reveal his location depending on how reputable or trustworthy the university is.

4.4 Basic Protocol

We present the basic protocol that is a secure two-party dynamic-programming protocol for computing the optimal solution of the credential selection problem. The basic protocol has two sub-protocols: recursion and traceback, which represent the two phases of dynamic programming. The protocol maintains the secrecy of sensitive parameters of both parties. Furthermore, neither the server nor the client learns any intermediate result. The main technical challenge is that the server does not want to reveal point values $\{p_i\}$ and the client does not want to reveal privacy scores $\{a_i\}$. As shown by the recurrence relation in Section 4.3, it seems difficult to compute entry $M_{i,j}$ without knowing p_i and a_i. We overcome the challenge by designing a protocol that hides the conditional testing from the client. The basic protocol is efficient and is secure in the semi-honest adversarial model.

4.4.1 Building Blocks

In our protocol, we store values in a modularly additively split manner with a large base L. The additively split manner means that the server and the client each has a share of a value, and the value equals to the sum of their shares modular L. If x^S and x^C represent the share of the server and the client, respectively, then the value equals to $x^S + x^C \bmod L$. We use $L - i$ to represent $-i$ (and use i to represent i). This notation implies that the range of the values is between $\frac{-L}{2}$ and $\frac{L}{2}$, and L must be chosen so that it is large enough to prevent accidental wrap-around. A secure two-party comparison protocol was given in [57] that allows the comparison of above-described split values, where two inputs to be compared are additively split between the client and the server. Their protocol outputs the comparison result in a XOR-split format, i.e., the comparison result is split between the two parties, and equals to the XOR of two shares. Therefore, neither party learns the comparison result. It is easy to modify a secure two-party comparison protocol to compute the

maximum of the values in an additively split format, which we refer to as a *private two-party maximization protocol* [1]. The maximization result can be additively split between the two parties so that neither party learns the value. We omit the details of private two-party comparison and maximization protocols, as we use them as a black box in this work.

Our protocols use homomorphic encryption extensively. Recall that a cryptographic scheme with encryption function E is said to be homomorphic, if the following holds: $E(x) * E(y) = E(x + y)$. Another property of such a scheme is that $E(x)^y = E(xy)$. The arithmetic performed under the encryption is modular, and the modulus is part of the public parameters for this system. Homomorphic schemes are described in [46, 97]. We utilize homomorphic encryption schemes that are semantically secure. Informally, a homomorphic scheme is *semantically secure* if the following condition holds. Given the public parameters of a homomorphic scheme E, and the encryption of one of the two messages m, m' where m is from a specific message and m' is chosen uniformly random from the message space, then $|(Pr(P(E(m))) = 1) - Pr(P(E(m')) = 1)|$ is negligible for any probabilistic polynomial time algorithm P.

4.4.2 Overview of Basic Protocol

The basic protocol consists of two sub-protocols: the basic recursion sub-protocol and the basic traceback sub-protocol.

- Basic recursion sub-protocol: the client and server compute a $(n+1) \times (T'+1)$ matrix M in an additive split form. Let $M_{i,j}$ denote the value stored at the i-th row and j-th column. Let E_C be the public encryption function of the client's semantically-secure homomorphic encryption scheme. The server learns $E_C(M_{i,j})$ values for all $i \in [1, n]$ and $j \in [1, T']$. From the security of E_C, a computationally-bounded server gains no information from the $E_C(M_{i,j})$ values. The server computes (with the client's help) the value $E_C(M_{i,j})$, when given

[1] Private two-party maximization protocol can be realized by expanding the scrambled circuit that is used for the comparison test. If the comparison result of two values is known, then the maximum can be computed. E.g., let $c \in \{0, 1\}$ be the result of a greater-than comparison between x and y, then $\max(x, y) = c * x + (1 - c) * y$.

$E_C(M_{i',j'})$ for all values (i', j') that are dominated by (i, j), for all $i \in [1, n]$ and $j \in [1, T']$. $M_{0,j}$ and $M_{i,0}$ are zeros, for all $i \in [0, n]$ and $j \in [0, T']$.

- Basic traceback sub-protocol: once the dynamic programming table has been filled out, the client discovers (with the server's help) the set of credentials that have been selected to disclose. The optimal selection is revealed to both parties.

Note that the basic recursion sub-protocol should unify the operations in the two cases ($j < p_i$ and $j \geq p_i$) of the recurrence relation. Otherwise, the client can learn p_i from the computation. We solve this by designing a generic and private maximum function and by additively splitting intermediate results between the two parties.

4.4.3 Basic Recursion Sub-Protocol

The basic recursion sub-protocol is described in Figure 4.1. When $j > T'$ (recall that $T' = \sum_{i=1}^{n} p_i - T$), the server terminates the protocol. The last entry $M_{n,T'}$ of the dynamic programming matrix has been computed. The client knows the marginal threshold T', as she keeps her share of the matrix. Yet, the client does not learn the individual point value p_i and access threshold T from the computation so far.

Lemma 1 *The complexity of the basic recursion sub-protocol is $O(nT')$, with $O(1)$ homomorphic encryptions or decryptions at each round, where n is the total number of credentials and T' is the marginal threshold.*

Proof: n corresponds to the row of the dynamic programming table, and T' corresponds to the column of the table. Filling up the entire dynamic programming table takes nT' rounds of computation. For each round of the basic recursion sub-protocol, there are constant number of homomorphic operations. Therefore, the lemma holds. □

The basic recursion sub-protocol runs in $O(nT')$, where marginal threshold T' or the number of credentials n can potentially be large. We point out that an important advantage of our protocol compared to conventional boolean-based policies lies in the privacy-preserving functionality offered. Our protocol not only computes the optimal selection of credentials, but also does it in a privacy-preserving fashion for both the

Setup: The client has published the public parameters of a semantically secure homomorphic scheme E_C. We will use the base of this scheme as the modulus for the additively split values.
Input: The server has $E_C(M_{i',j'})$ for all values (i', j') that are dominated by (i, j), where $i \in [1, n]$ and $j \in [0, T']$. The sever also has point values $p_1, \ldots, p_n$ and the client has privacy scores $a_1, \ldots, a_n$.
Output: The server learns $E_C(M_{i,j})$.
Steps:

1. The server creates a pair of values α_0 and α_1, where $\alpha_0 = E_C(M_{i-1,j})$, and $\alpha_1 = E_C(-\infty)$ if $p_i > j$, and $\alpha_1 = E_C(M_{i-1,j-p_i})$ otherwise. Without loss of generality, we assume that a_i values defined by the client are always bounded by an integer B that is known to the server, i.e. $a_i \leq B$ for all $i \in [1, n]$. The server then uses $-B - 1$ as $-\infty$. The server also chooses random values r_0 and r_1, and sends to the client $\alpha_0 E_C(r_0)$ and $\alpha_1 E_C(r_1)$.

2. The client decrypts the values to obtain β_0 and β_1. The server sets its shares to $-r_0$ and $-r_1$ and the client sets its shares to β_0 and $\beta_1 + a_i$. Note that the two candidate values for $M_{i,j}$ are additively split between the client and the server.

3. The client and the server engage in a private maximization protocol to compute the maximum of these two values in an additively split format. Denote the shares by x^S and x^C.

4. The client sends $E_C(x^C)$ to the server, and the server computes $E_C(x^C + x^S)$ and sets this value as his output.

Figure 4.1: Basic recursion sub-protocol.

server and client. For conventional policies, the latter aspect cannot be easily achieved without having the server to publish or disclose unfairly its policies.

The protocol presented here is secure in the semi-honest adversary model, which is improved later by our indexing expansion method in Section 4.5. The detailed security analysis is given in Section 4.6.

4.4.4 Basic Traceback Sub-Protocol

To support the back-tracking of the optimal solution (i.e., the optimal credential set to be disclosed), the basic recursion sub-protocol needs to be modified accordingly. At step 3 in the basic recursion sub-protocol, not only the maximum but also the *comparison result* of the two candidate values for $M_{i,j}$ are computed for all $i \in [1, n]$ and $j \in [1, T']$. During the computation, neither the server nor the client knows the result of the comparison tests, as the result is split between them. From the recurrence relation in Section 4.3, it is easy to see that the comparison result directly indicates whether a_i is contained in $M_{i,j}$ and thus whether credential C_i is selected. Denote F as a matrix that contains the result of the comparisons, we modify the previous basic recursion sub-protocol so that the server learns $E_C(F_{i,j})$ for the entire matrix. In the basic traceback sub-protocol, the server and the client work together to retrieve the plaintext comparison results starting from the last entry of the table, following the computation path of the optimal dynamic programming solution.

In this sub-protocol, the server sends its share $E_C(F_{n,T'})$ of the last entry of the table to the client, who decrypts it and obtains $F_{n,T'}$. $F_{n,T'}$ is either 0 or 1, and indicates whether the last credential C_n is selected or not. Then the client informs the server the plaintext value $F_{n,T'}$, which is used by the server to locate the value where $M_{n,T'}$ is computed from. If $F_{n,T'}$ is zero, then C_n is not selected and $M_{n,T'}$ is computed from $M_{n-1,T'}$. In this case, the next value to be revealed to the client is $E_C(F_{n-1,T'})$. Otherwise, credential C_n is selected to be disclosed. $M_{n,T'}$ is computed from $M_{n-1,T'-p_n}$. The value to be revealed to the client is $E_C(F_{n-1,T'-p_n})$. The is determined from the recurrence relation definition. Both parties repeat the sub-protocol until the upper left corner of the table F is reached. Figure 4.2 describes the basic traceback sub-protocol.

Lemma 2 *The complexity of the basic traceback sub-protocol is $O(n)$, with $O(1)$ homomorphic decryptions at each round, where n is the total number of credentials.*

The following theorem states the overall complexity of the basic protocol.

Theorem 4.4.1 *The complexity of the basic protocol is $O(nT')$, where n is the total number of credentials and T' is the marginal threshold.*

Input: The server has matrix entries $\{E_C(M_{i,j})\}$ and $\{E_C(F_{i,j})\}$ encrypted with the client's public key, for all $i \in [1, n]$ and $j \in [1, T']$. The client has her private key.
Output: The client learns the optimal value of the dynamic programming computation of knapsack. The server and the client learn the optimal selection of credentials, or nothing.
Steps:

1. The server sends the client $E_C(M_{n,T'})$. The client decrypts the ciphertext to obtain the result $M_{n,T'}$. $M_{n,T'}$ represents the privacy score associated with the unselected credentials. If this value is acceptable to the client according to some pre-defined privacy standard set by the client, then this sub-protocol continues. Otherwise, this sub-protocol terminates.

2. The server reveals the entry $E_C(F_{n,T'})$ to the client.

3. The client decrypts $E_C(F_{n,T'})$ to obtain $F_{n,T'} \in \{0, 1\}$. The client sends the plaintext value $F_{n,T'}$ to the server (The server then knows whether C_n is selected or not.)

   If $F_{n,T'} = 1$, then credential C_n will not be disclosed. $F_{n,T'} = 1$ also means that entry $M_{n,T'}$ is computed from entry $M_{n-1,T'}$. Therefore, the server next reveals $E_C(F_{n-1,T'})$ to the client. If $F_{n,T'} = 0$, then the server next reveals $E_C(F_{n-1,T'-p_n})$, as the entry $M_{n,T'}$ is computed from entry $M_{n-1,T'-p_n}$.

4. The revealed entries represent the computation path of the optimal knapsack dynamic programming solution. The above process is repeated until n reaches zero.

Figure 4.2: Basic traceback sub-protocol

Proof: n is the row of the dynamic programming table, and T' is the column of the table. Each invocation of the basic recursion sub-protocol fills up one entry of the table. Therefore, filling up the entire table takes nT' rounds. In the basic traceback sub-protocol, each round of the communication between the server and the client discovers whether a credential C_i is selected. Therefore, $O(n)$ number of rounds

are required for all the credentials. Hence, the basic protocol has the complexity of $O(nT')$. □

The basic traceback sub-protocol assumes that the server does not maliciously alter the computation results. In the case of a malicious server, the server may send $E_C(0)$ instead of the real values to mislead the client to disclose all credentials. Although the attack might be caught by the client (as the client may find a subset of credentials that still satisfies the threshold constraint), we give a stronger traceback algorithm that proactively prevents this type of attacks in the next section.

4.5 Fingerprint Protocol

In this section, we give an improved protocol for privacy-preserving knapsack computation. The new approach is inspired by the *subset sum problem*, yet we stress that this solution does not require the client to solve the general subset sum problem. The main idea is to allow the client (*not the server*) to efficiently identify the selected credentials from the optimal privacy score. The new protocol, which we refer to as the *fingerprint protocol*,[2] is an important step towards a protocol that is secure against malicious servers, because it can be extended to prevent the server from tampering the computation during traceback.

In addition to solving our credential selection problem (and thus the knapsack problem), the fingerprint protocol can be generalized to solve the traceback problem in a large variety of integer linear programming problems. It can be used for one party to securely and privately trace the optimal solution from the final computed value, with very little or no participation from the other party. The technique guarantees the correctness of the traceback results, even though the other party cannot be trusted during traceback.

4.5.1 Fingerprint Protocol Description

The key idea of the fingerprint protocol is to convert the client's privacy scores $\{a_i\}$ into another set of scores $\{A_i\}$, such that the following two conditions hold. (1) The

[2]The name is because of the similarities between fingerprinting in forensics and the indexing technique that we use to uniquely identify a subset.

optimal credential selection computed with $\{A_i\}$ should be the same as the optimal credential selection computed with $\{a_i\}$. (2) The privacy score computed with $\{A_i\}$ should reveal which set of credentials are used to obtain that score. Thus, this transformation process requires the following two properties:

Property 1 Ordering consistency: *For two sets S and R in $2^{\{1,\ldots,n\}}$, if $\sum_{i\in S} A_i < \sum_{i\in R} A_i$, then $\sum_{i\in S} a_i \leq \sum_{i\in R} a_i$.*

Property 2 Uniqueness: *For any two distinct sets S and R in $2^{\{1,\ldots,n\}}$, $\sum_{i\in S} A_i \neq \sum_{i\in R} A_i$.*

The ordering consistency property ensures that the set of revealed credentials computed with the transformed scores is optimal even when the original scores are used. The uniqueness property guarantees that traceback is possible, as only one set of credentials can generate a specific score. Although the above properties do not imply that an efficient traceback is possible, our transformation leads to an efficient traceback method. Our *indexing expansion* method transforms a privacy score a_i to A_i as follows.

$$A_i = a_i * 2^n + 2^{i-1}.$$

In binary representation, the indexing expansion shifts the binary form of a_i to the left by n positions, and gives zeros to n least significant bits except the i-th least significant bit, which is given a one. For example, suppose there are four privacy scores 2, 3, 5, 8 or in binary form 010, 011, 101, 1000. Here $n = 4$. After the transformations, the expanded scores have the binary form 010 0001, 011 0010, 101 0100, 1000 1000, respectively. Readers can verify that the example satisfy the two required properties. We now prove that the indexing expansion has the desired properties.

Lemma 3 *The indexing expansion achieves the ordering consistency property.*

Proof: For ease of notation, we use $A[S]$ to denote $\sum_{i\in S} A_i$, and $a[s]$ to denote $\sum_{i\in S} a_i$. Note that $A[S] = 2^{n+1}a[S] + \sum_{i\in S} 2^i$. Now suppose we have two sets S and R where $A[S] < A[R]$. Thus, $2^{n+1}a[S] + \sum_{i\in S} 2^i < 2^{n+1}a[R] + \sum_{i\in R} 2^i$. Now, it is easy to show that $\sum_{i\in S} 2^i < 2^{n+1}$ and $\sum_{i\in R} 2^i < 2^{n+1}$. Thus $a[S] \leq a[R]$. □

Lemma 4 *The indexing expansion achieves the uniqueness property.*

Proof: To show that the sums are unique, suppose we are given two sets S and R, where $S \neq R$. There must be some element j that is in one set but not the other, without loss of generality suppose $j \in S$. Now the jth bit of $A[S]$ will be 1, but it will be 0 for $A[R]$, and thus these two values are distinct. □

Hence, the indexing expansion method allows the client to compute the credentials that are used to achieve a specific privacy score. Although the optimal value obtained from the secure dynamic programming with the A_i scores is different from the one with the original a_i scores, the set of credentials corresponding to the optimal privacy values are the same. We now describe the fingerprint protocol, which makes use of the indexing expansion.

The indexing expansion of privacy scores requires n additional bits for each credential, where n is the total number of credentials. In Lemma 5 below, we prove that in order to satisfy the uniqueness property, the number of bits required for the transformed privacy scores is bounded by $\Omega(n)$.

Lemma 5 *For any transformation of index to satisfy the uniqueness property, the number of additional bits introduced for a privacy score is lower-bounded by $\Omega(n)$, where n is the number of credentials.*

Proof: Equation 4.1 holds because of the uniqueness property:

$$\sum_{i=1}^{n} A_i \geq 2^n - 1 \tag{4.1}$$

To be more precise, Equation 4.1 is because: i) each subset of credentials S must have a unique privacy score, ii) there are 2^n subsets, and iii) all A_i values must be positive. This implies that the maximum A_i is at least $2^{n-\log n} - \frac{1}{n}$, because $n(2^{n-\log n} - \frac{1}{n}) = 2^n - 1$. Because the length of the maximum value is at least $n - \log n - 1$, there must exist one A_i whose length is $n - \log n - 1$. Therefore, the number of bits introduced by the transformation is lower bounded by $n - \log n - 1$, and thus is $\Omega(n)$. □

Lemma 6 *The communication complexity of the traceback phase in the fingerprint protocol is $O(n)$, where n is the total number of credentials; the computation cost is $O(1)$ for the server, and is $O(n)$ for the client.*

Input: The server has the marginal threshold T' and point values $p_1, \ldots, p_n$. The client has privacy scores $a_1, \ldots, a_n$.
Output: The client (*not the server*) learns the optimal selection of credentials.
Steps:

1. The client applies the indexing expansion to each of her privacy scores $\{a_i\}$ and obtains the transformed scores $\{A_i\}$.
2. The server and the client carry out the basic recursion sub-protocol (in Figure 4.1) with the transformed privacy scores $\{A_i\}$. Recall that at the end of the basic recursion sub-protocol, the server has computed $E_C(M_{n,T'})$ in entry (n, T') of the dynamic programming matrix.
3. The server sends the ciphertext $E_C(M_{n,T'})$ to the client.
4. The client decrypts $E_C(M_{n,T'})$ to obtain $M_{n,T'}$.
5. The client expresses the optimal value $M_{n,T'}$ in binary form and identifies the non-zero bits in the last n bits. The positions of such bits give the indices of credentials that give the optimal solution[3]. Note that the i-th least significant bit of $M_{n,T'}$ is true if and only if credential i was used to obtain the optimal value.

Figure 4.3: Fingerprint protocol

Proof: Once the dynamic programming table is computed, the server only needs to send value $E_C(M^S_{n,T'})$ to the client. Hence, the number of communication rounds is constant. Because each privacy score a_i is expanded with n additional binary bits, the size of information transmitted is in the order of n – assuming that the privacy scores $\{a_i\}$ before the indexing expansion are bounded by a constant. Therefore, the communication cost of the algorithm is $O(n)$. It is trivial to show that the server's computation cost is constant. For the client, because she needs to identify $O(n)$ indexing bits, her computation cost is $O(n)$. □

Theorem 4.5.1 *The complexity of the fingerprint protocol is $O(n^2T')$, where n is*

the total number of credentials and T' is the marginal threshold.

Proof: The proof is similar to Theorem 4.4.1. For each round, both the server and the client perform constant number of homomorphic operations on transformed privacy scores $\{A_i\}$. Because A_i is $O(n)$ bits long – assuming that untransformed privacy scores $\{a_i\}$ are bounded by constant, the cost at each round is $O(n)$ for both parties. Hence, the overall complexity is $O(n^2T')$. □

4.5.2 Detection of Value Substitution by the Server

In the method described above, although difficult, it is not impossible for a malicious server to forge its share of the optimal value and thus mislead a client to disclose more credentials. The probability of the server correctly guessing a credential's privacy score and its bit position in the indexing expansion may not be negligible. For example, the server may have $1/n$ probability of correctly guessing the bit position of a credential, where n is the total number of credentials. Also, it may have $1/\max\{a_i\}$ probability of correctly guessing the privacy score, where $\{a_i\}$ represents the set of untransformed privacy scores. In Section 4.7, we describe a simple checksum technique for preventing the server from tampering with the traceback computation. This is done by appending randomized information to privacy scores.

4.6 Security

We define our security model as a semi-honest (a.k.a. honest-but-curious) model. Intuitively, semi-honest model means that adversaries follow the protocol but try to compute additional information other than what can be deduced from their input and output alone. A protocol is defined as secure if it implements a function f, such that the information learned by engaging in the protocol can be learned in an ideal implementation where the functionality is provided by a trusted oracle. This definition follows the standard definitions given by Goldreich [65] for private multi-party computation.

Let A be any one of the two parties in our protocol, we use $view_A$ to represent all of the information that A sees during the protocol. A protocol is secure against a

semi-honest A, if and only if there exists an algorithm that can simulate $view_A$ when given A's inputs and A's output. To be more precise, two probability ensembles $X \stackrel{\text{def}}{=} \{X_n\}_{n\in\mathcal{N}}$ and $Y \stackrel{\text{def}}{=} \{Y_n\}_{n\in\mathcal{N}}$ are computationally indistinguishable (i.e., a polynomial bounded algorithm cannot distinguish the two distributions) if for any PPT algorithm D, any positive polynomial p, and sufficiently large n it holds that: $|(Pr(D(X_n, 1^n) = 1)) - (Pr(D(Y_n, 1^n) = 1))| < \frac{1}{p(n)}$. Let A's input and output be represented by A_I and A_O respectively. A protocol is secure in the semi-honest model against adversary A, if there is an algorithm SIM_A such that $view_A$ and $SIM_A(A_I, A_O)$ are computationally indistinguishable (i.e., SIM_A simulates A's view of the protocol).

To prove the security of the basic protocol (in Figure 1), we state a lemma about the security of the private two-party maximization protocol used in step 3 of the basic protocol.

Lemma 7 *The private two-party maximization protocol is secure in the semi-honest model.*

The above lemma states that there exists a private two-party maximization protocol such that when given the client's inputs a^C and b^C, there is an algorithm that simulates the client's view of the maximization protocol. Denote this algorithm by $SIMMAX_C$ ($SIMMAX_S$) for the client (server). More specifically, if the client has input a^C and b^C, then $SIMMAX_C(a^C, b^C)$ is computationally indistinguishable from the client's view of interaction between the client and the server and the output of the protocol (i.e., the client's share of the maximum value). It is worth noting that the client's (server's) view of the max protocol is simulateable from that client's (server's) input alone, because the client's (server's) output from the maximization protocol is computationally indistinguishable from a random value.

Given such a private two-party maximization protocol, we show that the basic recursion sub-protocol in Section 4.4 is secure.

Theorem 4.6.1 *The basic recursion sub-protocol is secure in the semi-honest adversarial model.*

Proof: We must show that the server's and the client's view is simulateable from their input and output alone. The server's view consists of three things: i) the

interaction from the secure two-party maximization protocol, ii) the value x^S (i.e., the server's output) from the secure max protocol, and iii) $E_C(x^C)$. The simulator for the server outputs $(SIMMAX_S(-r_0, -r_1), E_C(r_2))$ for randomly chosen values r_0, r_1 and r_2. This simulation is computationally indistinguishable from the real view because of Lemma 7 and by the semantic security properties of E_C.

The client's view consists of four things: i) $M_{i-1,j} + r_0$, ii) $M_{i-1,j-p_i} + r_1$ or $-\infty + r_1$, iii) the interaction from the secure two-party maximization protocol, and iv) x_C. Since r_0 and r_1 are uniformly chosen values from the server, parts i) and ii) are computationally indistinguishable from a random value. Thus the simulator outputs $(r_2, r_3, SIMMAX_C(r_2, r_3))$ for random value r_2 and r_3. □

Theorem 4.6.1 shows that each individual round is secure in the basic recursion sub-protocol. The multi-round protocol is also secure, as the composition follows from the composition theorem [34]. Next, we show the basic traceback sub-protocol (in Figure 4.2) is secure. Note that the basic traceback sub-protocol makes uses of a matrix F that is computed in the recurrence phase. Each entry of matrix F contains the selection decision of a credential. The computation of F is secure, which can be easily deduced from Theorem 4.6.1. Theorem 4.6.2 summarizes the security of the traceback sub-protocol.

Theorem 4.6.2 *The basic traceback sub-protocol is secure in the semi-honest adversarial model.*

Proof: The output given to the server from the basic traceback sub-protocol is either a set of credentials that the client has disclosed or is an abort command from the client (when the client decides not to proceed with the transaction because the private information required by the server exceeds the client's tolerance). The server's view is simply the abort command or whether or not each credential is revealed by the client. This view is clearly simulateable from the server's output.

The output for the client is the privacy requirement for gaining access and the set of credentials that are to be revealed to the server (if the client chooses not to abort). The client's view of the protocol is $E_C(M^S_{n,T'})$ and $E_C(F_{i,j})$ (for row i and column j). They are intermediate results of knapsack computation encrypted with the public key of the client. The client's view consists of ciphertexts that are indistinguishable from random, and thus is simulateable. □

Given Theorem 4.6.1, the fingerprint protocol (in Figure 4.3) is secure, because once the server gives $E_C(M_{n,T'})$ to the client, the client carries out the traceback computation without any communication from the server.

Corollary 3 *The fingerprint protocol is secure in the semi-honest adversarial model.*

4.7 Technique for Cheating Detection

Our checksum technique has applications beyond the specific problem considered, and is a general method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants. We elaborate on this feature in this section. Let us consider an adversarial model described as follows. An adversary may tamper with intermediate results during the protocol, which is not allowed in a semi-honest model. An adversary is curious as in a semi-honest model, in that she may store all exchanged data and try to deduce information from it.

While we cannot prevent a participant from lying about her inputs, our checksum method described in this section enforces the *consistency in lying* by preventing capricious use of different inputs during the crucial solution-traceback phase. For complex functions such as the one being studied, lying about one's input wrecks the worthiness of the answer for both participants, and the participant who does so would have been better off not engaging in the protocol in the first place (this is not true for simple functions where the lier can still get the answer by *correcting for her lie*).

Our goal is to thwart the tampering attack by the server. In our protocols described so far, the server stores ciphertexts $E_C(M_{i,j})$ for all matrix values, therefore, the server can replace any value of the matrix with another value $E_C(v)$ for any arbitrary value v. In the fingerprint protocol, the server has to guess the weights used for each credential. The client can easily check if the proposed sum is created by a certain set of credentials. However, as described earlier, the server may have a non-negligible probability of successfully replacing these values. Next, we describe a technique that reduces the probability of a successful replacement by the server to a negligible value in terms of a security parameter.

The idea is that the client performs transformations on each of her privacy scores

by appending a random tag. The client creates a new set of value $\hat{A}_1, \ldots, \hat{A}_n$ that satisfy the traceback properties outlined in Section 4.5. For each value, A_i, the client randomly chooses a ρ-bit value (where ρ is the security parameter), which we call r_i. The client sets $\hat{A}_i = A_i 2^{\lg n + \rho} + r_i$ (where A_i is the already transformed value for traceback). This operation simply means shifting the binary representation of A_i $\lg n$ positions to the left and then appending r_i. It is straightforward to show that these values satisfy the properties outlined in Section 4.5. Furthermore, for the server to tamper with an intermediate result, it needs to guess a ρ bit random value. The server can guess successfully with only negligible probability in the security parameter ρ.

An attack that the server can launch on the above method is that it can send any intermediate value of the matrix to the client, and claim that the value is the final result. Because an intermediate value is well-formed, it cannot be detected by our above technique. However, we argue that the server does not gain from this type of attacks. If the server chooses a value from a higher row (with a smaller row index), then this attack can be achieved by setting the point values of some credentials to zero (i.e., they are useless to the client and are never used). If a different column is chosen, then this attack can be achieved by increasing the access threshold T. If the intermediate value is from a different row and a different column, then the effect of this attack can be achieved by increasing the threshold and setting the point values of some credentials to zero at the same time. The server may attempt to form linear combinations of row entries, but there is a non-negligible chance of being caught by the client because a repeated entry may be found.

4.8 Related Work

In the access control area, the closest work to ours is the framework for regulating service access and release of private information in web-services by Bonatti and Samarati [19]. They study the information disclosure in open systems such as Internet using a language and policy approach. In comparison, we design cryptographic solutions to control and manage information exchange. In addition, we focus on solving the optimality in selecting the set of credentials to disclose. Bonatti and Samarati considered two data types in the portfolio of a user: data declaration (e.g., identity, address,

credit card number) and credential. Although we only consider credentials in the description of our model, the protocols can be generalized to include data declarations as long as the server and the client agree on their specifications. In general, credentials (e.g., driver's license and credit card) contain a set of data declaration information, which is usually requested as a group. For example, the credit card number and the expiration date are usually asked for at the same time. Using credentials to represent private information may be sufficient in some cases.

Our point-based trust management model quantitatively treats memberships or credentials, which is conceptually different from most existing access control models. Our approach aims to address the fact that different individuals or groups of people have different privacy concerns in terms of protecting sensitive information. This goal differs from conventional access control models. The flexibility provided by the point-based model enables users to proactively protect their private information. Furthermore, thresholds specified by resource owners prevent unqualified users from accessing the resource.

Anonymous credential and idemix systems have been developed [28, 32] to allow anonymous yet authenticated and accountable transactions between users and service providers. Together with zero-knowledge proof protocols, they can be used to prove that an attribute satisfies a policy without disclosing any other information about the attribute. Our work focuses on finding the optimal credentials to disclose, and can be integrated with anonymous credential systems. A zero-knowledge proof protocol can be used when the necessary information to satisfy a policy is discovered. We can apply anonymous credential techniques to implement membership credentials in the point-based trust management model. These credentials are then used to prove user's memberships without revealing individual identity.

In hidden credentials system [27, 76], when a signature derived from an identity based encryption scheme [21, 44, 107] is used to sign a credential, the credential content can be used as a public encryption key such that the signature is the corresponding decryption key. Hidden credentials can be used in such a way that they are never shown to anyone, thus the sensitive credentials are protected. Bradshaw et al. [27] extended the hidden credentials system to support complex access policies expressed as monotonic Boolean functions. They applied a secret splitting system

to conceal the structure of such policies. The extended hidden credentials system protects the structure of Bob's polices. Frikken et al. [58] give a scheme that hides both credentials and policies. Most recently, a protocol [59] was proposed that allows both the client and the server to define *private* access policies of their credentials.

The setup of hidden credential protocols does not allow the computation of the *optimal* selection of credentials. In addition, as explained in the recent work by Frikken, Li, and Atallah [59], the server learns whether the client obtained access or not in some environments even when hidden credential schemes are used. In this case, the server can make inferences about the client's sensitive credentials. For example, if the server's policy is *one must have top secret clearance and be a FBI agent*, then the server can deduce a significant amount of information about the client when the access control decision is made. Our proposed solution allows the client to estimate potential privacy loss without leaking any sensitive information.

We have compared the trust negotiation protocols [120, 121, 133, 134] with our point-based trust management model in the introduction. Li, Li, and Winsborough introduce a framework for trust negotiation, in which the diverse credential schemes and protocols including anonymous credential systems can be combined, integrated, and used as needed [87]. The paper presents a policy language that enables negotiators to specify authorization requirements. The research on trust negotiation that is closest to ours is by Chen, Clarke, Kurose, and Towsley [41]. They developed heuristics to find an approximation of the optimal strategy that minimizes the disclosure of sensitive credentials and policies [41]. Using their methods, when negotiation fails, premature information disclosure is still a problem. Our protocols prevent premature information leakage, because the computation does not disclose sensitive parameters. Because the selection computation is private, the minimization problem is simpler to define in our point-based model than in trust negotiation frameworks. In addition, the solution computed by our basic and fingerprint protocols, if exists, is the exact optimal solution, not an approximation.

Secure Multi-party Computation (SMC) was introduced in a seminal paper by Yao [125], which contained a scheme for secure comparison. Suppose Alice (with input a) and Bob (with input b) desire to determine whether or not $a < b$ without revealing any information other than this result (this is known as *Yao's Millionaire Problem*).

More generally, SMC allows Alice and Bob with respective private inputs a and b to compute a function $f(a, b)$ by engaging in a secure protocol for public function f. Furthermore, the protocol is private in that it reveals no additional information. This means that Alice (or Bob) learns nothing other than what can be deduced from a (or b) and $f(a, b)$. Elegant general schemes are given in [14, 38, 64, 66] for computing any function f privately. However, these general solutions are considered impractical for many problems, and it was suggested in [67] that more efficient domain-specific solutions can be developed.

An efficient private computing protocol needs to have an advantage over the protocols based on the general results. General results in SMC simulate a circuit and require either (depending on the implementation): i) a 1-out-of-2 oblivious transfer (OT) per input wire, constant number of rounds, $O(1)$ invocations of a pseudorandom function per gate, and communication proportional to the number of gates, or ii) an OT per gate and rounds equal to the depth of the circuit.

Our protocols are more efficient than general circuit-based solutions. Circuits require multiplication operations, and the easiest circuits for k-bit multiplication require $O(k^2)$ gates. There are asymptotic improvements to these circuits, but they come at the cost of large constant factors; the asymptotically best of them (and the worst in terms of having impractically large constant factors) is a circuit of size $O(k \log k \log \log k)$ derived from the textbook Schoenhage-Strassen integer multiplication algorithm (which is itself of mainly theoretical interest, and not used in practice). Our protocols use homomorphic encryption multiplication, which requires $O(k)$ communication.

Besides the generic work in the area of SMC, there has been extensive work on the privacy-preserving computation of various functions. For example, computational geometry [3, 52], privacy-preserving computational biology [5]. The private dynamic programming protocol given by Atallah and Li [5] is the most relevant work to ours. Their protocol compares biological sequences in an additively split format. Each party maintains a matrix, and the summation of two matrices is the real matrix implicitly used to compute the edit distance. Our protocols also carry out computation in an additively split form. What distinguishes us from existing solutions is that we are able to achieve efficiently a stronger security guarantee without using

Zero-Knowledge Proofs [68]. Recently, there are also solutions for privacy-preserving automated trouble-shooting [77], privacy-preserving distributed data mining [78], private set operations [56, 83], and equality tests [91]. We do not enumerate other private multi-party computation work as their approaches significantly different from ours.

A novel category of interdisciplinary work on the economics of information security and privacy has recently been proposed [33, 48, 85]. Economics and psychology knowledge is applied to the problem of pricing the worth of security externalities and privacy information. In a recent study, Danezis, Lewis, and Anderson obtain an estimate of the value that users attach to their location data [48]. These important studies and analysis lay the foundation for applications that deploy our point-based model.

4.9 Conclusions and future work

This work is the first to formalize and solve the privacy-preserving credential selection problem. We gave a semantic-secure private two-party computation protocol for finding the optimal selection in an adversarial model that can handle cheating. The indexing expansion method that we described for the fingerprint protocol goes beyond the specific problem considered. It yields a general method for recovering an optimal solution from any value-computing dynamic programming computation, while detecting cheating by the participants.

The point-based trust management is an interesting framework that hosts much promising future research opportunities. One direction is to consider the constraint knapsack problem where a client specifies an arbitrary privacy score for a credential combination. This problem in general may be hard, but it would be interesting to see whether heuristics can be developed and private computation can be achieved. In addition, the expressiveness of the model can also be improved by solving multi-knapsack problem.

A related important topic is to study whether a satisfactory point scheme exists and how to systematically find one. The concept of quantitatively addressing the trust establishment problem has existed in several papers on trust and reputation models [15, 47, 130, 136]. These models have applications in open systems such as

presence systems [10] and peer-to-peer networks [47]. Sometimes, a suitable point scheme may not exist. For example, suppose Bob requires from Alice either (C_1 and C_2) or (C_3 and C_4) before he discloses some credential to Alice. Suppose Bob requires a threshold of 4 points. Then, whatever points we give to the four credentials, Alice can use one of the four invalid combinations (C_1 and C_3), (C_1 and C_4), (C_2 and C_3) and (C_2 and C_4) to get access, as one of them is guaranteed to be no less than 4 because their sum is at least 16. One solution to this problem is for the server to specify a point for the set (C_1 and C_3) higher than the sum of individual points. More efficient solutions are to be studied.

It would also be interesting to study how a service provider determines the point associated with each credential. Existing reputation systems [15] can be applied to this topic, and trust management mechanisms such as delegation-based access control model [7, 8] and role-based cascaded delegation [114] can be utilized to extend the scope of trust establishment.

Chapter 5

Privacy-Preserving Computation of Trust

A partial and preliminary version of this work was published in IFIPTM 2007 – Joint iTrust and PST Conferences on Privacy, Trust Management and Security 2007 [130].

5.1 Introduction

Conventional access decisions in stand-alone systems are usually made based on the identity of the entity requesting a resource. By comparison, in open systems such as the Internet, this approach becomes less effective. The main reason is that there is no central authority that can make access decisions. Thus, the resource owner and the requester typically belong to security domains administrated by different authorities that are unknown to each other. For example, Alice is holding a student credential from an organization A, but Bob, the resource owner, may know nothing about A in terms of its trustworthiness, etc. Therefore, there is a strong need for designing a flexible trust establishment model.

Another motivation for flexible authorization comes from financial applications such as e-commerce. An issue that may dissuade consumers from fully utilizing e-commerce applications is the potential misuse of their disclosed private information by vendors. In most situations, consumers do not have a quantitative measure of how much their sensitive credentials are worth, and may be under-compensated when

disclosing private information in exchange for rewards. Without a quantitative model, it is hard for consumers to make intelligent decisions on whether or not to disclose a credential in exchange for rewards.

Privacy-aware presence systems are another important area that needs a flexible trust and authorization model. Location information obtained via GPS devices embedded in cellphones or cars represents private user data that should not be queried freely by the public. Similarly, in a workplace such as an office building or hospital, the privacy of presence information should be protected. The management of presence data is crucial, because it concerns not only user privacy, but also safety: presence data can be used to track and profile individuals. In the meantime, there may be emergency situations or extenuating circumstances when certain parties (like emergency workers) should have access to this kind of information, and friends and relatives of a user might be allowed to query his or her location information at any time. Therefore, a desirable feature of a location query system is that it provides different levels of precision based on the requester's trustworthiness or the context of the query. This requires a flexible authorization model for accessing the private location data.

To meet the requirements of trust establishment in open systems, we develop a trust model for access control based on the credentials provided by a requester. The model computes a trust value on the requester, which is used to make access control decisions by a provider.

Reputation or trust models [35, 84, 136] provide an open, flexible, and dynamic mechanism for trust establishment, where the requester does not belong to the resource owner. Trust models have applications in distributed systems such as peer-to-peer networks, e-commerce applications such as online auctions, or in resource-sharing systems such as Grid computing. Trust models are typically built on information such as recommendations and previous experiences of individuals. Various algorithms have been proposed to evaluate trust values [15, 116], in particular how transferred trust are computed.

We address two aspects of computational trust models: **(1)** how to protect the privacy of personal opinions during computation, and **(2)** how to design a scalable computational trust model.

In computational trust models, the recommendations on the trustworthiness of users are usually assumed to be public. However, recommendations represent one's personal opinions of other entities, and are usually considered sensitive. For example, Bob has bad experiences doing business with Paul on an auction site, but, he does not want to publish his negative recommendation on Paul for fearing of Paul's revenge. Alice, who has not dealt with Paul previously, would like to use Bob and others' recommendations to evaluate Paul's trustworthiness. In the meantime, Alice has her own private evaluations on Bob and others, which give weights to individual recommendation (e.g., Alice knows and trusts Bob, so Bob's recommendation has a higher weight.) The problem is how to enable Alice to compute the weighted recommendation on Paul without disclosing everyone's sensitive parameters. We formalize this problem as a secure multi-party computation of scalar product, and present an efficient protocol for solving it.

Figure 5.1 gives a simple example of trust relationships and values. Suppose Alice wants to buy somethings from Bob but does not know anything about Bob. However, Alice's peers Carl, Doug, and Ed have had previous interactions with Bob. Each of them gives a trust score on Bob's trustworthiness that is 0.8, 0.6, 1.0, respectively. Suppose 1 is complete trustworthy and 0 is not trustworthy at all. In the meantime, Alice may not completely trust her peers opinions and she has her own judgement on the trustworthiness of her peers. For example, she thinks that Carl's opinions are more reliable than Ed's. Thus, Ed's opinion on Bob is *discounted* by Alice. Alice and her peers uses our private scalar product protocol to compute a value of trust score on Bob based on all these known factors. A more complex scenario is shown at the bottom of Figure 5.1 where Alice's peer has indirect knowledge about Bob rather than direct previous interactions. Frank, an Alice's peer, knows Greg who knows Helen who knows Bob. Here, the chained trust relationship reflects how trust is transferred and the computation should incorporate all these factors. We formalize this problem in our trust model.

We also describe an approach to improve the scalability of trust and reputation models. Ideally, a trust model should be able to accurately and efficiently classify a group of users. In trust management applications with a large number of users, such as Shibboleth [110], the trustworthiness of individual users becomes less important if

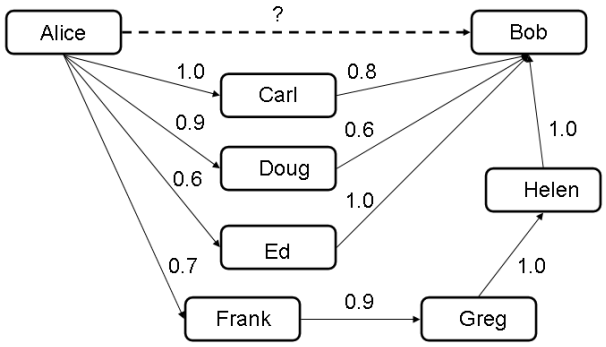

Figure 5.1: An example of trust relationships and trust values.

the resource owner knows the home organization of the individual. For example, if the user is a professor from a reputable college, then he or she is likely to be trustworthy. We aim to improve the scalability of the typical grass-root approach of building trust. Our approach takes advantage of the pre-existing organizational infrastructure, in particular the credential-based administration model. The trustworthiness of an individual is deduced from her digital credentials and the credential issuers' trustworthiness.

5.1.1 Our Contributions

Our contributions are summarized as follows.

1. We present a private multi-party computation protocol for computing weighted trust values. The problem is for A to infer the trust value of an unknown entity X based on what other entities think about X together with A's confidence in these entities. In a world where there is no privacy concern or there is a trusted third-party, the problem can be solved by computing the scalar product of two vectors – one vector representing A's confidence values for a set of entities, and the other vector representing recommendations of these entities on X. In real life, this information is usually considered sensitive, e.g., B may not want to disclose that he does not trust X at all, and A hopes to conceal the fact that her confidence in B is low. Private two-party scalar product protocols are

available [3, 63, 117]. However, they are not suitable for our problem, where one of the vectors in the computation is distributed among multiple entities. We design an efficient private multi-party computation protocol for scalar products where individual values of a vector can have different owners. The sensitive information of all parties is not revealed (except the final scalar product).

2. We propose a credential-based trust model for inferring trustworthiness in decentralized environments. Our credential-based trust model not only simplifies and scales the decision-making process, but also improves the reliability of computed trust scores by using role certificates. We describe how to compute trust values from multiple credentials, delegation credentials, and from peers' recommendations. Our model can also be used for computing point values in the existing point-based authorization model.

3. We also describe a location-query system for giving approximate location information based on the trustworthiness of the query issuer. This system is a practical application of the point-based authorization model, and demonstrates the ability to give flexible yet confident trust verdicts in open systems. Location-aware applications are made popular by the increasing deployment of sensor networks, RFID, and GPS-enabled cellphone networks.

5.1.2 Outline

A private multi-party computation protocol for distributed scalar products is presented in Section 5.2. This protocol supports efficient and privacy-preserving computation of trust values. Our credential-based trust model is introduced in Section 5.3. In Section 5.4, we describe how our trust model can be integrated with the existing point-based trust management model. In Section 5.5, we present an application of point-based trust management to the location query problem for sensor networks. Related work is described in Section 5.6. Finally, future work is given in Section 5.7.

5.2 Private Distributed Scalar Product Protocol

In this section, we define, construct, and analyze the private distributed scalar product protocol. The private distributed scalar product protocol has applications in privacy-preserving data mining problems. In Section 5.3.2, we show how it is used to privately compute trust values from peers' recommendations.

5.2.1 Definitions

In what follows, we define that all arithmetic is done in $\mathbb{Z}_m$ for some m. A private distributed scalar product protocol is to compute $X \cdot Y$, where $X = (x_1, x_2, \ldots, x_n) \in \mathbb{Z}_m^n$ and $Y = (y_1, y_2, \ldots, y_n) \in \mathbb{Z}_m^n$ are vectors of length n.

The protocol is run by l numbers of players where $1 \leq l \leq 2n$, and x_i and y_i are disjointly partitioned among the players. That is, each player knows one or more of the elements in the vectors, and a vector is known by one and only one player. In a centralized case where $l = 1$, the problem is reduced to trivial scalar product computation. If $l = 2$, i.e. a two-party private computation problem, one can use existing private scalar product protocols [3, 63, 117]. If there are $2n$ players, each party knows only one element in X or Y. The goal of the protocol is for the players to jointly compute $X \cdot Y$ without disclosing each own's private information, i.e., x_i or y_i values. The security of the protocol can be intuitively thought of as players do not gain non-negligible knowledge of others' private information (besides the final scalar product). In particular, the property should hold even if players collude. The security of the protocol is further analyzed in Section 5.2.4.

For our trust model in Section 4.3, we are interested in a specific scenario with $n+1$ players: Alice wants to compute the point value for an unknown entity E. She knows n entities $B_1, B_2, \ldots, B_n$, and Alice's point value for entity B_i is x_i. Each entity B_i knows entity E, and has assigned point y_i to E, respectively. Alice and $B_1, B_2, \ldots, B_n$ jointly compute $X \cdot Y$, which is given to Alice at the end of the protocol, but not to any of the B_is. We present our private distributed scalar product protocol for this special case. The protocol can be easily generalized to cases where l is anywhere between 3 and $2n$, where n is the length of the vector.

5.2.2 Building Blocks

Our private distributed scalar product protocol uses the homomorphic encryption scheme and a private multi-party summation protocol.

Homomorphic Encryption

A homomorphic encryption scheme has three functions $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, where Gen generates a private key sk and a public key pk, Enc and Dec are encryption and decryption functions, respectively. The encryption function Enc is said to be homomorphic, if the following holds: $\mathsf{Enc}_{\mathsf{pk}}(x; r) \cdot \mathsf{Enc}_{\mathsf{pk}}(y; r') = \mathsf{Enc}_{\mathsf{pk}}(x + y; r \cdot r')$, where x and y denote plaintext messages and r and r' denote random strings. Another property of such a scheme is that $\mathsf{Enc}_{\mathsf{pk}}(x; r)^y = \mathsf{Enc}_{\mathsf{pk}}(x \cdot y; r^y)$. This means that a party can add encrypted plaintexts by doing simple computations with ciphertexts, without having the private key. The arithmetic performed under the encryption is modular, and the modulus is part of the public parameters for this system. Homomorphic schemes are described in [46, 97]. We utilize homomorphic encryption schemes that are semantically secure. A homomorphic scheme is called *semantically secure* when a probabilistic polynomial-time adversary cannot distinguish between random encryptions of two elements chosen by herself.

Private Multi-Party Summation Protocol

Our protocol also uses an efficient private multi-party summation protocol, which was presented by Atallah *et al.* [4]. Their protocol is to make n parties, each with a number V_i, cooperate to *simultaneously* find out $\sum_{i=1}^{n} V_i$ without revealing to each other anything other than the answer. To achieve this, each party chooses a random value, which is used to hide the input. The intermediate sum is additively split among the participants.

The summation protocol by Atallah *et al.* [4] is described in Figure 5.2. Note that to compute the sum, the protocol should not let each party send his share in the clear to all other parties, which is obviously insecure. The protocol in [4] gives a non-trivial way to do this by requiring the participants to compute a randomized private sum. We use the summation protocol as a black box, and refer readers to the literature for more details [4].

PRIVATE INPUTS: Every party i has a private value V_i.
PRIVATE OUTPUT: Every party learns $V = \sum_{i=1}^{n} V_i$.

1. Party i chooses a random number R_i.
2. Every party $2i$ gives to $2i+1$ his $V_{2i} + R_{2i}$, then every $2i+1$ gives to $2i$ his R_{2i+1} in a secure channel.
3. The odd-numbered parties together compute the sum $V + R$, where $V = \sum_{i=1}^{n} V_i$ and $R = \sum_{i=1}^{n} R_i$. The even-numbered parties together compute the sum R.

   The summation of $V+R$ and R can be done in a tree-based approach where the parties are organized at leaf nodes of a tree and the summation is computed in a bottom-up fashion. The root of the tree gives the final sum.
4. An odd party with $V + R$ and an even party with R simultaneously exchange their quantities to obtain V.

   The simultaneous exchange of secrets can be realized using methods (see e.g., [105]). However, as pointed out in [4], a simpler and more efficient bit-exchange approach is suitable for the summation protocol and does not compromise the security. An odd party sends one bit of his value to the even party, and the even party sends one bit to the odd party. Then they alternate until done.

Figure 5.2: Privacy-preserving summation protocol by Atallah *et al* [4]. Note that lying during the exchange in Step 4 cannot be prevented, yet a player can achieve the same effect by lying about his input. In addition, lying does not let the player learn anything about the sum V.

5.2.3 Protocol Description

Our private distributed scalar product protocol is shown in Figure 5.3. Alice's input of the protocol is a private vector X. Each party B_i (for $1 \leq i \leq n$) has a private value y_i. At the end of the protocol, the scalar product $X \cdot Y$ is learned by Alice or by every participant, where $Y = (y_1, \ldots, y_n)$.

Alice encrypts each element x_i of her vector X with her public key in homomorphic encryption. The ciphertext c_i is sent to B_i, respectively. Because B_i does not know Alice's private key, Alice's value is safe. Because of the properties of homomorphic encryption, entity B_i is able to compute the ciphertext corresponding to $x_i y_i$, even though he does not know x_i. The resulting ciphertext is w_i in Figure 5.3. To hide y_i, B_i computes the ciphertext w_i' corresponding to $x_i y_i - s_i$, where s_i is a random number. Alice receives ciphertext w_i' from each B_i, and computes the product of all w_i's, which is decrypted to $X \cdot Y - \sum_{i=1}^{n} s_i$. Next, all of B_is carry out a private multi-party summation protocol that computes $\sum_{i=1}^{n} s_i$. At the end of the summation protocol, every B_i learns the sum. Alice obtains the sum from B_is, and computes $X \cdot Y$ without learning the individual y_i values.

Our private distributed scalar product protocol is based on the private two-party scalar product protocol by Goethalsh *et al.* [63], where each party has a vector and the protocol outputs the scalar product result of the two vectors in a split form. That is, the scalar product result is split between the two parties, and equals to the sum of two shares. The concept of shared private computation can also be found in [3, 57]. A variant of our protocol allows all participating parties to learn the scalar product result $X \cdot Y$. Alice with S_A and all B_is, each with s_i, carry out a private multi-party summation protocol with their inputs. Our analysis is based on the protocol in Figure 5.3.

5.2.4 Analysis of the Protocol

The correctness of the protocol is obvious. Alice obtains from B_i (for all $i \in [1, n]$) an encryption of $x_i y_i - s_i$. Alice multiplies the n ciphertexts, and decrypts to obtain the sum $\sum_{i=1}^{n} x_i y_i - s_i$. Once Alice obtains $\sum_{i=1}^{n} s_i$, she computes $X \cdot Y = \sum_{i=1}^{n} x_i y_i$. The security and efficiency of our private multi-party protocol for distributed scalar product are analyzed.

PRIVATE INPUTS: Private vector $X = (x_1, \ldots, x_n) \in \mathbb{Z}_m^n$ by Alice; private values y_1 by entity B_1, ..., y_n by entity B_n, where $y_i \in \mathbb{Z}_m$ for all $i \in [1, n]$.
PRIVATE OUTPUTS: Alice learns $X \cdot Y \mod m$, where m is a public parameter.

1. Setup phase. Alice does: Generate a private and public key pair $(\mathsf{sk}, \mathsf{pk})$. Send pk to all B_i.
2. Alice does for $i \in \{1, \ldots, n\}$: Generate a random new string r_i. Send $c_i = \mathsf{Enc}_{\mathsf{pk}}(x_i; r_i)$ to B_i.
3. B_i does: Set $w_i = c_i^{y_i} \mod m$. Generate a random plaintext s_i and a random nonce r_i'. Send to Alice $w_i' = w_i \cdot \mathsf{Enc}_{\mathsf{pk}}(-s_i; r_i')$.
4. Alice does: Compute the product of ciphertext w_i's as $\Pi_{i=1}^n w_i' \mod m$. Use her private key sk to decrypt the product, and obtain the partial result $S_A = X \cdot Y - \sum_{i=1}^n s_i$.
5. All B_is, each with s_i, carry out a private multi-party summation protocol with their inputs (described in Section 5.2.2 and Figure 5.2). At the end of that protocol, each B_i obtains $S_B = \sum_{i=1}^n s_i$.
6. Alice does: Obtain S_B from (any of the) B_is. Compute $X \cdot Y = S_A + S_B$.

Figure 5.3: Private Distributed Scalar Product Protocol. m is a public parameter of the homomorphic encryption scheme.

The security of our private multi-party scalar product protocol is based on the security of the private two-party scalar product protocol [63] and the private multi-party summation protocol [4]. In general, the multi-party protocol among players is secure when the privacy and correctness are guaranteed for all players. It is said that a protocol protects privacy when the information that is leaked by the distributed computation is limited to the information that can be learned from the designated output of the computation [99]. In our problem, Alice's private vector X and each entity B_i's private value y_i are not leaked to each other, besides the scalar product.

Operation	Scalar Product Phase	Summation Phase	Total
Comp. (Alice)	$O(n)$ homomorphic op.	$O(1)$	$O(n)$ homomorphic op.
Comm. (Alice)	$O(n)$	$O(1)$	$O(n)$
Comp. (B_i)	$O(\log y_i)$ homomorphic op.	$O(1)$	$O(\log y_i)$ homomorphic op.
Comm. (B_i)	$O(1)$	$O(1)$	$O(1)$

Table 5.1: Computation (Comp.) and communication (comm.) complexities of the private distributed scalar product protocol. We denote by n the length of Alice's vector X. The logarithmic factor is due to using multiplications to compute exponentiation in step 3.

Note that in almost all existing private scalar product solutions, one player can construct a system of linear equations based on the specification of the protocol, and solve it for the secret values.

Our security is in the semi-honest model, where it is assumed that all players follow the protocol, but they are also curious: that is, they may store all exchanged data and try to deduce information from it. One challenge in designing the multi-party scalar product protocol is to prevent collusions among players. In particular, during the step of summation, Alice may attempt to collude with a subset of players B_is to discover the private values of other players.

As in almost all private multi-party protocols, we assume that each party inputs his or her true private values. Providing skewed values during computation can result in inaccurate results, and wasting the computation power and bandwidth of all participants including the dishonest party. In addition, the effect of providing skewed intermediate value by a participant can be achieved by raising or lowering his or her own input. This issue is standard in multi-party protocols (both semi-honest and malicious models). Suppose A wants to compute the trustworthiness of C with help of $B_1, \ldots, B_n$, and suppose B_i is a friend of C, B_i may modify the output of the protocol by raising s_i in Figure 5.3. As a result, A gets a higher value for C. However, B_i can achieve the same effect by choosing a different input to begin with. Therefore, this type of attacks is not considered in multi-party protocols including ours. It is worth mentioning that once detected, this type of behaviors could be folded back into the reputation of participants, which can provide incentives for being honest during the computation.

Because of the intrinsic nature of the problems considered, even if the protocol is

secure in the malicious model (discussed later), multi-party computation such as ours is still vulnerable to probing attacks. For example, if A wants to learn B_i's private value y_i, A can engage the protocol with input $X = (0, \ldots, 0, 1, 0, \ldots, 0)$ by setting only the i-th entry to be one. After the protocol A learns $X * Y = y_i$, which is the private value of B_i.

The security of our protocol is summarized in the following theorem.

Theorem 5.2.1 *Assume that* $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is a semantically secure homomorphic public-key cryptosystem. The private distributed scalar product protocol presented in this section is secure in the semi-honest model. Alice's privacy is guaranteed when for all* $i \in [1, n]$*, entity* B_i *is a probabilistic polynomial-time machine. Also, for all* $i \in [1, n]$*,* B_i*'s privacy is information-theoretical.*

Proof: Each entity B_i only sees a random ciphertext from Alice, for which B_i cannot guess the ciphertext. This is because of the semantic security of the homomorphic encryption scheme. Hence, B_i cannot guess Alice's value x_i.

During the summation protocol, each B_i only sees random values exchanged. Hence, B_i cannot guess the random secret s_j of B_j for all $j \neq i$.

On the other hand, Alice only sees (1) random value $x_i y_i - s_i$, (2) the sum of all s_i, and (3) the final computation scalar product $X \cdot Y$. She does not gain additional information about Y besides the final scalar product. In addition, the protocol prevents collusions among Alice and a subset D of B_is to discover private y_j value of B_j for $B_j \notin D$, because the summation protocol guarantees that all B_is learn the sum simultaneously. Thus, Alice obtains no information about any B_i except the scalar product $X \cdot Y$, and each B_i obtains no information about Alice and entity B_j for all $j \neq i$. □

The overall computation and communication complexities of our protocol are the same as the private two-party scalar product protocol by Goethals *et al.* [63]. The private multi-party summation protocol is efficient, as it does not require any type of encryption schemes. The summation step does not introduce significant overhead. Details of complexities are summarized in Table 5.1.

Security in a malicious model Malicious adversaries, unlike semi-honest ones, can behave arbitrarily without following the protocol. They may refuse to participate the protocol, abort the protocol without finishing it, and tamper with intermediate

values. Any protocol secure against honest-but-curious adversaries can be modified to a protocol that is secure against malicious adversaries using standard zero-knowledge proofs showing that all parties follow the protocol. At each step of the protocol, each party uses their transcripts and zero-knowledge proofs to convince the other parties that they have followed the protocol without cheating. We do not describe the details of how this transformation is done here.

5.3 Credential-Based Trust Model

In this section, we present a simple credential-based trust model that is useful for the trust management in distributed environments. The main idea is to convert role-based credentials and related information into quantitative trustworthiness values of a requester, which is used for making authorization decisions. Quantitative authorization policies can allow fine-tuned access decisions instead of binary (allow or deny) verdicts, and provide more diversified access options for requesters. In addition, quantitative authorization enables providers to correlate the quality of service with the qualifications of requests (e.g., more rewards or higher resolution with higher trustworthiness). This approach utilizes and leverages existing credential and role-based management infrastructure for autonomous domains (e.g., [114, 129]) and improves the accuracy of trustworthiness prediction.

Our private multi-party scalar product protocol in the previous section can be used to compute trust values from recommendations in Section 5.3.2.

We divide our description of the credential-based trust model into the following topics.

1. Derive the trust value of an affiliated role credential, which is defined next.
2. Compute the trust value of a delegation role credential, which is defined next.
3. Integration with point-based trust management system, which is described in Section 5.4.

Terminology: In our model, we define the *administrator* of a role as the organization that creates and manages the role. If a role credential of an entity D is signed and issued by the administrator of the role, that role is said to be an *affiliated role*

of D (this type of role is usually obtained through the affiliation with an organization, and thus the name). If a role credential of D is instead issued through delegation and signed by entities other than the administrator of the role, that role is called a *delegated role* of D. We define an *entity* to be an organization or an individual. An entity may issue credentials. Also, an entity may have one or more affiliated roles or delegated roles, which are authenticated by role credentials. An *affiliated role credential* is the credential for an affiliated role, and is signed by the administrator of the role. Similarly, a *delegated role credential* is the credential for proving a delegated role. A *privilege* can be a role assignment or an action on a resource. A role r administered by entity A is denoted as $A.r$. A role defines a group of entities who are members of this role.

5.3.1 Definitions in Credential-Based Trust Model

A trust value in the credential-based trust model represents what an entity thinks about the trustworthiness of another entity or a role in another entity. More specifically, trust value $t(A, B)$ in the credential-based trust model represents what entity A thinks about the trustworthiness of entity B; trust value $t(A, B.r)$ in the credential-based trust model represents what entity A thinks about the trustworthiness of role $B.r$ administered by entity B. For example, a Grid Computing facility GCLab assigns trust values to types of users, such as role *professor* and role *student* in a university U, and role *researcher* from a research center C. When a user holding a certain role credential requests for access to the grid computing facility, his or her privileges are specified based on the trust value of that role. Note that the credential-based trust model is different from existing trust models that generate rating certificates, which are signed certificates of one's trustworthiness generated by one's peers [101].

Ideally, an entity A maintains a trust value for each role in organization B. For example, GCLab gives different trust value to role *student* and role *professor* in a university. Hence, a requester with a *professor* role credential may be granted a different level of access privileges from a requester with a *student* role credential.

Definition 5.3.1 *If an entity A gives a role $B.r$ in B a trust value $t(A, B.r)$, then any individual who has a valid affiliatedrole credential of role $B.r$ issued by B has the trust value $t(A, B.r)$.*

In case a resource owner does not know the trust value of a role in an organization, the trust value of that organization is used as a guideline for the trustworthiness of the role. In general, we define that any requester who has a valid role credential issued by organization B has the same trust value as B.

There are two main approaches for an entity A to obtain the trust value of B. One is based on A's previous direct interactions with B. The other approach is to derive from other entities's trust values on B, which can be thought of as recommendations. The two approaches can be combined to bring a more precise judgement. In this work, we do not address the first approach, namely, how to directly derive trust values from previous transactions with an entity or its roles, because the specific methods to be used depend highly on the applications. For example, Tran *et al.* proposed how to derive trust scores in a P2P file-sharing systems [122]. We focus on techniques for computing trust values from other entities' recommendations and on how to carry out the computation in a privacy-preserving fashion. In what follows, we use *trust value of a credential* to mean the trust value of the credential issuer.

5.3.2 Derive Trust Value From Recommendations

We describe a *weighted average* method for an entity A to compute a trust value on entity B or role $B.r$. This computation is useful when A does not have any previous interaction experience with B or $B.r$, and A wants to combine others' opinions of B or $B.r$ in forming her trust value.

In the credential-based trust model, the *recommendation* by an entity E on B is the trust value $t(E, B)$ that E gives to B. A *confidence value* represents how much A trusts the judgement of a recommender, and is defined as the trust value of A on the recommender.

Above definitions mean that recommendations are weighted by A's confidence on the recommenders. Formally, we define the weighted average computation of trust value as follows. We denote n as the number of recommenders, and E_i represents the i-th recommender. Let MAX_TRUST be the public upper bound of all trust values. Without loss of generality, we assume a trust value is non-negative. We assume that A has already obtained her trust values $t(A, E_1)$, $t(A, E_2)$, ..., $t(A, E_n)$ on the recommenders. We also assume that each of the recommenders E_i has formed

her trust value $t(E_i, B)$ on the target entity B. (In case no one in the system knows about entity B, a default trust value can be assigned to B to indicate this situation.) The formula for computing $t(A, B)$ is shown as follows, where weight $w(A, E_i) = t(A, E_i)/\text{MAX\_TRUST}$.

$$t(A, B) = \frac{1}{n} \sum_{i=1}^{n} w(A, E_i) t(E_i, B) \tag{5.1}$$

Value $w(A, E_i)$ represents the weight of E_i's recommendation (trust value) on B for A. Variants of weighted average computation have been used in other reputation systems, such as ordered weighted average [123]. The above description also applies when the target to be evaluated is a role, for example $B.r$, instead of an entity.

Application of private distributed scalar product protocol. Equation (5.1) is useful for A only when all the trust values $t(E_i, B)$ are available. However, trust value $t(E_i, B)$ is private information of E_i, who has the incentive to hide it, especially when E_i thinks negatively about B. Similarly, A may consider her trust values $t(A, E_i)$ sensitive too. The problem is how to compute the weighted average in (5.1) without leaking the private information of each entity. Our protocol for private multi-party scalar product in Section 5.2 solves this problem and satisfies the privacy requirement. Note that our model is *not* based on the assumption of two degrees of separation between any two entities, that is, we do not need to assume that a new entity is known by an existing peer. If an entity is not known to the community, it is initialized with trust value zero, which may increase provided that the entity behaves well with other peers. Recall that a trust value can be based on previous experience of interactions with an entity.

Combining trust values for access. If a requester presents multiple role credentials, then the trust values of the credentials are to be combined. For example, one simple method is to sum the trust values. This means that the requester with multiple credentials of low trust values can gain the same access privileges as a requester with one credential of a high trust value. This combination method is intuitive and is used in point-based trust management model [127], which will be discussed in Section 5.4.

5.3.3 Generalization of our computational model

The trust relationships of our trust model described so far assumes that A's peers directly know B. However, a more general scenario is where A indirectly knows B through multiple peers, e.g., the scenario depicted at the bottom of Figure 5.1. We generalize our trust model to incorporate this aspect as follows. We model the trust relationships of entities in the system as a directed graph G, where there is a weighted directed edge between entity X and Y, if X knows Y and the weight of the edge is the trust value $t(X, Y)$. The distance between X and Y depends on the directed path chosen, e.g., in Figure 5.1, the distance between Alice and Bob is four if the bottom path is chosen. We refer the directed path connecting two entities A and B as a trust path.

Our model for computing trust values can be generalized to include long trust paths by multiplying weighted trust values corresponding to a trust path. To compute A's trust value on B, A first needs to choose the trust paths connecting to B. We assume that A has already known n non-overlapping directed paths from A to B. Note that A's paths do not have to be the complete set of such paths in G. However, incorporating more paths into the computation generates a more accurate estimation on B's trustworthiness. For the i-th path ($i \in [1, n]$) between A and B, let m_i be the number of entities on the path besides A and B. Denote such a node as $E_{i,j}$ where $j \in [1, m_i]$. Trust value $t(A, B)$ is computed as Equation 5.2.

$$t(A, B) = \frac{1}{n} \sum_{i=1}^{n} w(A, E_{i,1}) w(E_{i,1}, E_{i,2}) \dots t(E_{i,m_i}, B) \tag{5.2}$$

In Equation 5.2, $t(A, B)$ is computed by incorporating the n weighted paths between A and B. The computation does not favor longer paths in that the longer the path, the more weights (≤ 1) are multiplied that may lower the resulting trust value. This trend is consistent with the intuition that direct recommendation is more trustworthy than indirect one. Because the weights are normalized by MAX_TRUST, more paths (i.e., higher n) do not necessarily give a higher trust value. However, considering more paths in the computation does produce a more accurate reflection on B's trustworthiness by the community. How to choose the non-overlapping paths may depend on A's preferences. This topic is out of the scope of this work and is not

discussed here.

5.3.4 Delegation Chain and Trust Computation

In this section, we describe how our trust model is further generalized to support delegation credentials. Delegation [8, 114, 129] is important for transferring trust in decentralized environments. Associating trust values with delegation credentials is different from role credentials because the values should not only depend on the initial credential issuer, but also the intermediate delegators's trustworthiness.

A delegation credential represents how a certain prvilege is transferred among multiple delegators. Intuitively, if a delegator on a delegation chain has low trustworthiness, then the trust value of the delegation credential should be affected. If all the delegators are highly trustworthy, the delegation credential should earn its holder a trust value similar to a directly-issued role credential. We capture these intuitions in computing the trust value of a delegation credential into a term *discount factor*, which represents how much the trust value of a delegated privilege is decreased due to intermediate delegators. Before we give details of the definition, we first introduce several important concepts of delegation.

The *original issuer* or *original delegator* of privilege P is the first entity on a delegation chain, and is the owner of the resources associated with privilege P. A *delegation chain* of privilege P is the path that shows the delegation sequence of P between entities. The chain connects a delegated entity to the original issuer of P.

In general, there are two types of role-based delegations, based on who is allowed to issue delegation credentials. One type is that an organization delegates its permissions to roles in other organizations [89]. The delegation is issued by the administrator of an organization. The other type is administrator-free delegation, where an individual role member of an organization issues the delegation to other roles without the participation of administrators during delegation. The latter is designed for decentralized transfer of trust, and is embodied in a model called role-based cascaded delegation [114, 129] as shown in Figure 5.4. We give a general method for computing discounted trust value of a delegation credential for both delegations with or without administrators.

Next, we briefly introduce role-based cascaded delegation model.

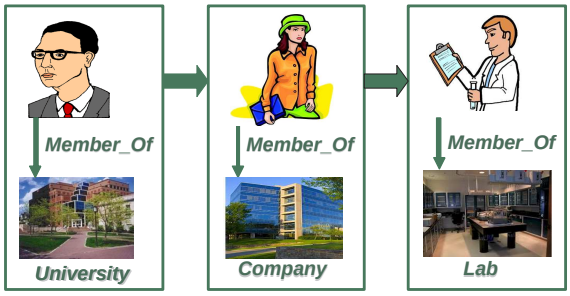

Figure 5.4: The schematic drawing of a role-based delegation chain. It shows that a member of a university delegates permissions to a member of a company, who then delegates the permission to a member of a lab. The horizontal arrows indicate delegation of permissions. The vertical arrows indicate membership relationship.

Role-Based Cascaded Delegation

Role-based Cascaded Delegation [114] model supports administrator-free delegation. It enables flexible and dynamic authorization in a decentralized environment. It comprises four operations: **Initiate**, **Extend**, **Prove**, and **Verify**. **Initiate** and **Extend** are used by a resource owner and an intermediate delegator, respectively, to delegate a privilege to a role. **Prove** is used by a requester to produce a proof of a delegation chain that connects the resource owner with the requester. **Verify** decides whether the requester is granted the access based on the proof.

In the RBCD protocol [114], a delegation credential includes role membership certificates of each intermediate delegator, and delegation extension credentials that are proofs of delegation transactions signed by delegators. Credentials associated with a delegation chain are transmitted to delegated role members at each delegation transaction. Therefore, for a delegation chain of length n, the number of certificates required to verify the delegation path is $2n$.

Discounted Trust Value for Delegation Credential

Suppose that an individual B has a delegation credential. We denote D_0 as the original delegator or the resource owner. We denote $D_0.r$ as the role being delegated in the delegation credential, which is a role administrated by D_0. We denote $D_1, \ldots, D_n$ as the intermediate delegators on the delegation credential. We denote $D_1.r, \ldots, D_n.r$ as roles of intermediate delegators. Note that in our credential-based trust model, the

specific role member who issues the delegation is not needed to participate in computing discounted trust value. The trust value that an entity A gives to the credential holder B is computed as follows, where weight $w(A, D_i) = t(A, D_i)/\text{MAX\_TRUST}$.

$$t'(A, B) = \Pi_{i=1}^{n} w(A, D_i.r) t(A, D_0.r) \tag{5.3}$$

In Equation 5.3, the trust value of a delegated credential is based on the length of the delegation chain, the trust value $t(A, D_0.r)$ of the delegated role, and the trust values $t(A, D_i.r)$ of intermediate delegator D_i for all $i \in [1, n]$. The weight $w(A, D_i.r)$ represents the discount factor on the trust value $t(A, D_0.r)$. Intuitively speaking, if entity A thinks that intermediate delegators are highly trustworthy, the final result $t(A, B)$ is close to value $t(A, D_0.r)$.

Finally, to make the role-based delegation valid, the credential holder B needs to not only possess the delegation credential, but also have a valid affiliated role credential of the last authorized role of the chain. We denote $D_{n+1}.r$ as the last role that the delegation credential is issued to. For example, in Figure 5.4, this corresponds to a lab member. B should be an affiliated role member of $D_{n+1}.r$. Therefore, the complete trust value of credential holder B with role $D_{n+1}.r$ should combine the trust value of the delegation credential with the trust value $t(A, D_{n+1}.r)$ of his or her role, as shown in Equation 5.4. As mentioned in Section 5.3.2, we use the summation to combine trust values from multiple credentials.

$$t(A, B) = t'(A, B) + t(A, D_{n+1}.r) \tag{5.4}$$

$$= \Pi_{i=1}^{n} w(A, D_i.r) t(A, D_0.r) + t(A, D_{n+1}.r) \tag{5.5}$$

The above description of computing discounted trust values of a delegation credential applies to both types of role-based delegations: delegation with or without administrators.

5.4 Integration With Point-Based Trust Management

Our proposed private multi-party protocol and trust model are useful for general access control in a decentralized environment. We describe how it can be used for deriving point values in the existing point-based trust management model [127], which was proposed for the privacy protection of sensitive information in open environments. We briefly introduce the point-based model next.

5.4.1 Point-Based Trust Management

In the point-based trust management model [127], the authorization policies of a resource owner define an *access threshold* for each of its resources. The threshold is the minimum number of points required for a requester to access that resource. For example, accessing a medical database might require fifty points. The resource owner also defines a *point value* for each type of credential, which denotes the number of points or credits a requester obtains if a type of credential is disclosed. For example, a valid ACM membership might have ten points. This means that a user can disclose his or her ACM membership credential in exchange for ten points. (This is called a trust management model as opposed to an access control model, because the resource owner does not know the identities or role assignments of requesters *a priori* as in conventional access control settings.)

Each user defines a *sensitivity score* for each of their credentials. The sensitivity score represents the unwillingness to disclose a credential. For example, Alice may give a sensitivity score of ten to her college ID, and give fifty to her credit card. The user is granted access to a certain resource if the access threshold is met and all of the disclosed credentials are valid. Otherwise, the access is denied. From the requester's point of view, one central question is how to fulfill the access threshold while disclosing the least amount of sensitive information.

The credential selection problem in the point-based trust management model is to determine an optimal combination of requester's credentials to disclose to the resource owner, such that the minimal amount of sensitive information is disclosed and the access threshold of the requested resource is satisfied by the disclosed credentials.

A private two-party dynamic programming protocol has been proposed to solve the credential selection problem [127].

The point-based authorization model assumes that the resource owner (or server) and the requester (or user) agree on a set of credential types as the universe of credentials $(C_1, \ldots, C_n)$. A binary vector $(x_1, \ldots, x_n)$ is defined as the unknown variable to be computed, where x_i is one if credential C_i is selected and zero if otherwise. Integer variable $a_i \geq 0$ is the *sensitivity score* of credential C_i. It is assigned by the requester *a priori*. If the requester does not have a certain credential C_i, the sensitivity score a_i for that credential can be set to any integer larger than T, where T is the trust threshold for the requested resource. Integer variable $p_i \geq 0$ is the point value for releasing credential type C_i. The requester considers all a_i values sensitive, and the server considers all p_i values sensitive.

The credential selection problem is for the requester to compute a binary vector $(x_1, \ldots, x_n)$ such that the sum of points $\sum_{i=1}^{n} x_i p_i$ satisfies T, and the sum of sensitivity scores $\sum_{i=1}^{n} x_i a_i$ is minimized. This is captured in the following minimization problem. Compute a binary vector $(x_1, \ldots, x_n)$ such that the following holds:

$$\min \quad \sum_{i=1}^{n} x_i a_i \tag{5.6}$$

$$\text{subject to} \quad \sum_{i=1}^{n} x_i p_i \geq T \tag{5.7}$$

The above minimization problem can be rewritten into a knapsack problem, which can be solved by dynamic programming. A private two-party computation protocol was given in [127] for the dynamic programming problem with sensitive p_i and a_i values. The protocol in [127] is different from our private distributed scalar product protocol, as we aim to solve how point values can be privately computed in a reputation model.

5.4.2 Derivation of Point Values

Previous work on the point-based trust management model [127] focused on the privacy protection of sensitive information and assumes that the point value associated with each credential type of the requester has already been determined by the

server [127]. It does not describe how point values are obtained or how to systematically derive points corresponding to credentials. The mechanism for determining the point value of a credential is crucial to the applicability of the trust management model, and needs to be formalized. In cases where the credential issuer of a requester is not previously recognized by the resource owner, we need a protocol to compute an appropriate point value for the credential held by the requester. The credential-based trust model presented in Section 4.3 answers this question. Using the described methods, a resource owner computes the trust values of credential issuers and their roles. The resulting trust values are to be used as point values of a resource owner in point-based trust management.

For delegation credentials presented by a requester, a resource owner can use the trust model to compute the discounted trust value of the credential. The trust value can only be computed exactly when the delegation credential is revealed. However, this information is private to the requester in the credential selection computation in point-based trust management. To mitigate this problem, a resource owner can use an approximate trust value during the credential selection computation, and then make adjustments when credentials are exchanged later.

The credential-based trust model completes the description of an important aspect in point-based authorization. Next, we give a concrete application for point-based authorization in location-query systems.

5.5 Applications to Location Query Systems

Privacy is an important concern in systems that use presence and other real-time user data. Presence provides great utility, but also has the potential for abuse. Managing security and privacy preferences in these systems can be complex. One approach to protect the privacy is to apply distributed anonymity algorithms to sensor networks [73, 74]. Another type of solutions is to augment existing routing protocols to enhance source-location privacy in sensor and conventional networks [82, 113].

However, these existing solutions are not suitable for several types of applications. In many scenarios such as 911 or medical emergency, road-side emergency of a GPS-enabled vehicle, and police enforcement agents, the location information of a subject

is critical, and should not be hidden or anonymous. Also for example, in distributed collaboration applications such as Meeting Central [124], being able to share presence information to trusted collaborators is desirable.

Generally, sharing presence information implies sharing sensitive personal data such as computer activity, physical location, IM status, phone use, and other real-time attributes associated with a given user. Managing the privacy of this data requires capturing the user's preferences and concerns, which are typically quite individualistic. Some users feel comfortable sharing any personal details, but most want at least some control over what is shared and with whom.

We are interested in how to manage access to private presence information in a way that makes users feel that their preferences are met. In this section, we describe how point-based authorization can be used as a key component for flexible privacy management in presence systems. The point-based trust management is intuitive enough to let the user understand the implications of their sharing decisions.

5.5.1 A Location-Query Service

As an application of point-based trust management, we have started to prototype a presence system that applies points to access control. A presence system can provide a service that runs on behalf of each user, acting as that user's always-online proxy. Through this proxy, the user has ultimate control over all their associated data. The proxy is resolvable based on the user's identity, and can expose services that can be queried by other entities in the system. One such service provides presence querying.

Entities in the system can pose questions to Alice's proxy like *where is Alice now?* This is handled by Alice's presence service, which must first find valid answers to the question, and then determine which answers, and to what degree of specificity, will be returned. The answers are generated by interpreting real-time presence data (GPS coordinates, keyboard and mouse activity, current calendar appointments, etc.) associated with Alice, which may be captured from arbitrary locations but which flows exclusively into her proxy, thereby giving Alice ultimate authority over her own personal presence data. The allowable answers are determined by querying Alice's access system, which uses points in several ways.

5.5.2 Advisors and Point-Based Decisions

Alice's proxy chooses access decisions through a set of domain-specific entities called advisors. Each advisor provides input on possible decision responses based on its domain of expertise (e.g., reputation, purpose of the query, context of the exchange, value of the requested data). These inputs are then aggregated to determine the overall advice about a possible response. The idea is to provide a flexible mechanism that more accurately represents a user's decision process. Our credential-based trust model and point-based authorization can be used to implement a flexible advisor system. For this example, we focus just on reputation, but the point-based model can generally be applied to a number of these domain-specific problems.

Alice's proxy contains her policies and preferences, including the trust values of credentials that may be used for authentication. Alice also defines the precision associated with certain trust values. For example, if the trust value of the query issuer is twenty, then she might release her location information exactly. If the trust value is five, then she might release a *approximate interpretation* of her location, for example, the building or city where she is currently. Phrased more concretely, if Alice's closest friend, Bob, queries about her location, a precise answer is returned. If a stranger queries her location, nothing about Alice should be disclosed.

The reputation advisor computes the trust value of each query issuer, based on their credential information. The trust value is then compared to Alice's policies, and the corresponding location result is returned. The advisors reside in Alice's proxy that is a tamper-resistant system in order to prevent the leaking of private trust values. Note that this model makes it easy to use the trust value not just in deciding what to share, but in determining the system's confidence that the right decision is made. A high trust value represents high confidence and can be executed without bothering Alice. A low trust value represents low confidence in a decision, and if low enough, may warrant interrupting Alice to check that the right decision is being made for her. This confidence metric is then fed back into the system for use the next time a similar query from the same entity arrives, and used to provide an aggregate sense of past confidence.

For location-query systems, the main advantages of using point-based trust management as opposed to conventional access control mechanisms are the flexibility of

making access control decisions with an arbitrary degree of precision and the ability to derive some simple notion of confidence. In order to achieve the same expressiveness, a boolean-based access control policy would be very inefficient, as one needs to enumerate all of the possible combinations of authorizations.

5.6 Related Work

Secure Multi-party Computation (SMC) was introduced in a seminal paper by Yao [125], which contained a scheme for secure comparison. Suppose Alice (with input a) and Bob (with input b) desire to determine whether or not $a < b$ without revealing any information other than this result (this is known as *Yao's Millionaire Problem*). More generally, SMC allows Alice and Bob with respective private inputs a and b to compute a function $f(a, b)$ by engaging in a secure protocol for public function f. Furthermore, the protocol is private in that it reveals no additional information. This means that Alice (resp. Bob) learns nothing other than what can be deduced from a (resp. b) and $f(a, b)$. Elegant general schemes are given in [14, 38, 64, 66] for computing any function f privately.

Besides the generic work in the area of SMC, there has been extensive work on the privacy-preserving computation of various functions. For example, computational geometry [3, 52], privacy-preserving computational biology [5], and private two-party dynamic programming for the knapsack problem [127]. Compared to existing private scalar product protocols [3, 63, 117], our protocol is designed for general privacy-preserving distributed scalar product computation, where vector values are distributed among multiple players. The protocol has promising applications in the information discovery of reputation systems. Our security is efficient, and is comparable to the private two-party scalar product of Goethalsh *et al.* [63].

Recently, there are also solutions for privacy-preserving automated trouble-shooting [77], privacy-preserving distributed data mining [78], private set operations [56, 83], and equality tests [91]. We do not enumerate other private multi-party computation work as their approaches are significantly different from ours.

There has been much work on the privacy-awareness for ubiquitous computing environments [73, 82, 86, 111]. An existing approach to protect the location-privacy

in sensor networks is through distributed anonymity algorithms that are applied in a sensor network, before service providers gain access to the data [73]. Another category of solutions is to augment existing routing protocols to enhance source-location privacy in sensor and conventional networks [82, 113]. A more fine-grained approach for managing the access to location data is based on privacy-policies [86, 111], which is closer to our solution. Using point-based authorization, we are able to support more flexible trust establishment mechanism without rigid boolean-based policy specifications.

Our trust model work is related to the existing work on recommendation or reputation systems in decentralized models. [15]. [84]. Trust evidences that are generated by recommendations and past experiences have been used for trust establishment in both ad-hoc and ubiquitous computing environments [54, 108, 115]. This type of trust evidence is flexible and straightforward to collect. The notion of uncheatable reputation was proposed in recent work by Carbunar and Sion [35], who developed a reputation mechanism that prevents untruthful reputation information using witnesses. In comparison, the main property of our trust model is the use of role-based organizational infrastructure to derive trust values, which aims to improve the scalability of trust computation.

5.7 Conclusions and Future Work

We have developed a general protocol for privacy-preserving multi-party scalar product computation. This protocol can be used for peers to jointly compute a weighted trust score from *private* recommendations and *private* weights. We have also presented a simple credential-based trust model for evaluating trustworthiness based on role and delegation credentials, and recommendations. Finally, we have described the architecture of a location-query system for giving approximate location information based on the trust score of a requester.

There are several interesting areas to explore for future work. One is to evaluate other types of trust computation besides weighted average. For example, the ordered-weighted-average operator allows the user to weight the input values in relation to their relative ordering [123]. Another promising direction is to design private

multi-party protocols for other desirable functionalities in a trust model. For example, an entity wants to find out who else in the system has a similar profile of trust values as his or her own — other entities who have similar likes and dislikes. The problem becomes how to privately compute the distance between two set of trust values according to certain metrics. As part of future works, we also plan to evaluate the effectiveness of credential-based trust model in answering approximate location queries. This experimentation involves an implementation of the point-based authorization model, the weighted scalar protocol computation, and the comparison tests with conventional trust models.

Chapter 6

Conclusions and Open Problems

6.1 Conclusions

This dissertation studies privacy-aware authentication and authorization problems in trust management systems and presents solutions for flexible, efficient, and secure trust establishment for parties who are from different administrative domains and initially unknown. In particular, we have been focusing on the authentication and authorization in distributed environments, the security of shared resources, and the privacy of participants. Many of our protocols involve multiple autonomous and self-interested participants interacting with each other to establish trust and service transactions. Much of our effort has been directed toward the design of provable-secure cryptographic protocols that are resilient to adversarial attacks, while enabling privacy protections and accountability for access, and supporting efficient and scalable wide-area deployment.

In this thesis, we give an administrator-free role-based delegation protocol that facilitates dynamic cross-domain collaborations. The main feature of the protocol is that any role member can issue delegation certificates to transfer access privileges that can be verified without the participation of role administrators. We also point out that our delegation protocol can be efficiently realized by aggregate signature scheme so that the delegation chains bear compact and short signatures. We give a simple cryptographic signature scheme for hiding the identity of delegators in a role-based delegation chain. Our signature scheme preserves the anonymity of delegator as a

verifier can only infer that a signature is signed by a valid role member, but does not learn the exact identity. In addition, the multiple signatures can be aggregated that give compact and efficient presentation.

To minimize the disclosure of the sensitive credentials and policies in trust management, we develop a privacy-preserving quantitative authorization protocols. Our solution allows the client to compute the optimal set of credentials to disclose before actually revealing them. The client has the power of selecting what private information to reveal to the resource owner. Therefore, our authorization model gives incentives to clients to participate in the transactions which is important for the advance of e-commerce in general. We study the integration of our quantitative authorization model with recommendation systems that are used to derive trust parameters in our authorization model. We develop an efficient secure multi-party scalar product protocol for protecting sensitive recommendations where a recommender does not need to publish his or her recommendations and thus avoid possible revenges from peers. Our quantitative authorization model can be used to fine-tune services based on qualifications of a client. We describe how this application is realized in a presence system.

Authentication and authorization are important components of computer systems. The outcome of the relevant research efforts can potentially have wide real-world applications in a number of fields including web-services and e-commerce in general, digital identity management, Email and anti-spam, and auditing services. The aim of protocols developed in this area is to have broad and far-reaching impacts on people's perspectives on the trust and privacy of today's digital world. The desirable impacts include drastically improved consumers' confidence in the security and privacy of online shopping and banking, significantly savings for financial institutions and ISPs due to reduced ID thefts and more robust message delivery, and the increased participation of the society in web-services and e-commerce in general. Next, we list a few specific open problems to be solved in the near future.

6.2 Open Problem: Authentication and Privacy of Identities

Dumpster diving, shoulder surfing, phishing, key logging are common causes of current identity theft problem. However, most identity management protocols, both physical and digital ones, are fundamentally susceptible to identity theft, because key pieces of personal information are usually entered in clear on paper or on computer (e.g., one needs to type in her credit card number during an on-line purchase). Recently, a few counter-measurements against identity theft have been proposed, but they either involve a centralized system or require significant changes to existing financial and administrative infrastructures.

A challenging problem is to design an usable identity management system that not only is robust against identity theft, but also requires minimal changes to current infrastructures. One approach is to systematically investigate the tradeoffs between usability and security of an identity management system, and the tradeoffs between a centralized management model and a fully decentralized one. These studies will provide insights for designers to better understand important factors in designing a successful identity management system. Beside protocol design and implementation, we also need a formal security treatment to the identity theft problem, which includes a formal definition of identity theft, security, and adversarial models. Our previous work on this topic demonstrated the feasibility of a notary middleware for web-services and federated identity management [72].

Protecting user privacy is an important aspect of digital identity management. Cryptographic protocols need to be carefully designed such that users enjoy web-services to a maximum degree without concerning about losing privacy. For example, service providers and identity providers should be unable to collude to gather an user's transaction history. A challenging aspect is to protect user privacy under a broader definition of digital identity. The IP address is one (usually fixed) identity of an user. Conventional routing protocols allow ISP and search engines to deduce shocking details of people's private lives based on search queries and websites visited. Anonymous techniques such as anonymous routing and anonymous credentials are one promising approach to address these problems, although much work remains to

be done to improve the efficiency. Our work on compact anonymous-signer signature [128] is a good first-step toward this goal.

6.3 Open Problem: Authentication Services and Middleware

In domain spoofing attacks, such as DNS cache poisoning, adversaries exploit security vulnerabilities in name servers and launch attacks by impersonating financial services. SSL certificates make domain spoofing harder for attackers, however, they are costly to purchase, and time-consuming to update. Furthermore, the society still needs to strengthen the effort of educating average consumers in order to take full advantage of the power of SSL certificates. DNSSEC is a cryptographic extension for DNS proposed in 1999, which aims to use cryptographic protocols to secure the communications between name servers. However, the scalability of DNSSEC seriously limits the adoption of the protocol for today's Internet. Improving the security protection of ISP's name servers (by patching) can certainly improve the overall reliability of the naming systems. However, end users will need to passively rely on ISP's system administrators being wisely maintaining their name servers.

A proactive approach is to deploy a middleware architecture to allow end users' computers to authenticate the IP/name mappings of visited web servers (e.g., verifying that 192.193.226.190 is the correct IP of citicard.com's web server). The IP/name information is publicly available and can be easily crawled. The challenging aspect of the approach will be to dynamically, efficiently, and securely disseminate the authentication information to end users. For example, many financial institutions outsource their customer services to third-party providers, which result in cross-domain redirects. Cryptographic protocols will need to be carefully designed to take into consideration the dynamic nature of addresses. This work will not only offer a more general system for domain authentication compared to my previous solution in accredited mails [71], but also provides insights to the design of scalable authentication middleware for many other important security problems.

Bibliography

[1] A. W. Appel and E. W. Felten. Proof-carrying authentication. In *ACM Conference on Computer and Communications Security*, pages 52–62, 1999.

[2] R. Aringhieri, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. Assessing efficiency of trust management in peer-to-peer systems. In *1st International Workshop on Collaborative Peer-to-Peer Information Systems (COPS '05)*, 2005.

[3] M. J. Atallah and W. Du. Secure multi-party computational geometry. In *Proceedings of 7th International Workshop on Algorithms and Data Structures (WADS 2001)*, volume 2125 of *Lecture Notes in Computer Science*, pages 165–179. Springer Verlag, August 2001.

[4] M. J. Atallah, H. G. Elmongui, V. Deshpande, and L. B. Schwarz. Secure supply-chain protocols. In *2003 IEEE International Conference on Electronic Commerce (CEC 2003)*, pages 293–302. IEEE Computer Society, 2003.

[5] M. J. Atallah and J. Li. Secure outsourcing of sequence comparisons. In *4th Workshop on Privacy Enhancing Technologies (PET)*, volume 3424 of *Lecture Notes in Computer Science*, pages 63–78, 2004.

[6] G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In *Advances in Cryptology — CRYPTO '00*, volume 1880 of *LNCS*, pages 255–270. Springer Verlag, 2000.

[7] T. Aura. On the structure of delegation networks. In *Proceedings of 11th IEEE Computer Security Foundations Workshop*, pages 14–26. IEEE Computer Society Press, 1998.

[8] T. Aura. Distributed access-rights management with delegation certificates. In *Secure Internet Programming – Security Issues for Distributed and Mobile Objects*, volume 1603 of *LNCS*, pages 211–235. Springer, 1999.

[9] D. Balfanz, G. Durfee, N. Shankar, D. Smetters, J. Staddon, and H. Wong. Secret handshakes from pairing-based key agreements. In *2003 IEEE Symposium on Security and Privacy*, pages 180–196. IEEE Press, 2003.

[10] N. Banerjee, A. Acharya, and S. Das. Enabling SIP-based applications in Ad Hoc networks. *Journal of Wireless Networks*. Invited submission under review.

[11] P. S. Barreto, H. Y. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology — Crypto '02*, volume 2442 of *LNCS*, pages 354–368. Springer-Verlag, 2002.

[12] M. Bellare, D. Micciancio, and B. Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In *Advances in Cryptology - EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 614–629, 2003.

[13] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.

[14] M. Ben-Or and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *The Twentieth Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–10. ACM Press, 1988.

[15] T. Beth, M. Borcherding, and B. Klein. Valuation of trust in open networks. In *Proceedings of the Third European Symposium on Research in Computer Security (ESORICS '94)*, pages 3–18, November 1994.

[16] R. Bhatti, J. Joshi, E. Bertino, and A. Ghafoor. X-GTRBAC admin: a decentralized administration model for enterprise wide access control. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 78–86, 2004.

[17] M. Blaze, J. Feigenbaum, and A. D. Keromytis. KeyNote: Trust management for public-key infrastructures. In *Proceedings of Security Protocols International Workshop*, 1998.

[18] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[19] P. A. Bonatti and P. Samarati. A uniform framework for regulating service access and information release on the web. *Journal of Computer Security*, 10(3):241–272, 2002.

[20] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Advances in Cryptology — Crypto '04*, LNCS, 2004.

[21] D. Boneh and M. Franklin. Identity-Based Encryption from the Weil Pairing. In *Proceedings of Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229. Springer, 2001.

[22] D. Boneh and M. K. Franklin. Identity-based encryption from the Weil pairing. In *Advances in Cryptology — Crypto '01*, volume 2139 of *LNCS*, pages 213–229. Springer-Verlag, 2001.

[23] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In *Advances in Cryptology — Eurocrypt '03*, pages 416–432, 2003.

[24] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. A survey of two signature aggregation techniques. *CryptoBytes*, 6(2), 2003.

[25] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology – Crypto '05*, 2005.

[26] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. In *Advances in Cryptology — Asiacrypt '01*, volume 2248 of *LNCS*, pages 514–532. Springer-Verlag, 2001.

[27] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *Proceedings of 11th ACM Conference on Computer and Communications Security (CCS)*, Oct. 2004.

[28] J. Camenisch and A. Lysyanskaya. Efficient non-transferable anonymous multishow credential system with optional anonymity revocation. In B. Pfitzmann, editor, *Advances in Cryptology — EUROCRYPT 2001*, volume 2045 of *Lecture Notes in Computer Science*, pages 93–118. Springer Verlag, 2001.

[29] J. Camenisch and A. Lysyanskaya. Dynamic accumulators and application to efficient revocation of anonymous credentials. In *Advances in Cryptology — Crypto '02*, volume 2442 of *LNCS*, pages 61–76, 2002.

[30] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology — CRYPTO '04*, 2004.

[31] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. In *Advances in Cryptology – CRYPTO '97*, volume 1296 of *LNCS*, pages 410–424. Springer-Verlag, 1997.

[32] J. Camenisch and E. Van Herreweghen. Design and implementation of the *idemix* anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS)*, pages 21–30, 2002.

[33] L. J. Camp and C. Wolfram. Pricing security. In *Advances in Information Security – Economics of Information Security*, volume 12, pages 17–34. Kluwer Academic Publishers, 2004.

[34] R. Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[35] B. Carbunar and R. Sion. Uncheatable reputation for distributed computation markets. In *Financial Cryptography and Data Security Conference (FC '06)*, 2006.

[36] M. Chase and A. Lysyanskaya. On signatures of knowledge. In *Advances in Cryptology - CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 78–96. Springer, 2006.

[37] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM*, 28(10):1030–1044, October 1985.

[38] D. Chaum, C. Crépeau, and I. Damgard. Multiparty unconditionally secure protocols. In *The twentieth annual ACM Symposium on Theory of Computing (STOC)*, pages 11–19. ACM Press, 1988.

[39] D. Chaum and J.-H. Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *Proceedings of Advances in cryptology—CRYPTO '86*, pages 118–167, January 1987.

[40] D. Chaum and E. van Heijst. Group signatures. In *Advances in Cryptology — Eurocrypt '91*, pages 257–265. Springer-Verlag, 1991.

[41] W. Chen, L. Clarke, J. Kurose, and D. Towsley. Optimizing cost-sensitive trust-negotiation protocols. In *Proceedings of the 24th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, volume 2, pages 1431–1442, 2005.

[42] X. Chen, F. Zhang, and K. Kim. A new ID-based group signature scheme from bilinear pairings. In K. Chae and M. Yung, editors, *Proceedings of International Workshop on Information Security Applications (WISA) 2003*, volume 2908 of *LNCS*, pages 585–592. Springer, August 2003.

[43] D. Clarke, J.-E. Elien, C. Ellison, M. Fredette, A. Morcos, and R. L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9(4):285–322, 2001.

[44] C. Cocks. An identity based encryption scheme based on quadratic residues. In *8th IMA International Conference on Cryptography and Coding*, volume 2260, pages 360–363. Springer, Dec. 2001.

[45] T. H. Cormen, C. E. Leiserson, R. L.Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, 2001.

[46] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *4th International Workshop on Practice and Theory in Public Key Cryptosystems (PKC '01)*, LNCS 1992, pages 119–136, 2001.

[47] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, and F. Violante. A reputation-based approach for choosing reliable resources in peer-to-peer networks. In *ACM Conference on Computer and Communications Security (CCS '02)*, pages 207–216, 2002.

[48] G. Danezis, S. Lewis, and R. Anderson. How much is location privacy worth? In *Fourth Workshop on the Economics of Information Security (WEIS 2005)*, 2005.

[49] P. Devanbu, M. Gertz, C. Martel, and S. Stubblebine. Authentic third-party data publication. *Journal of Computer Security*, 11(3), 2003.

[50] Y. Ding, P. Horster, and H. Petersen. A new approach for delegation using hierarchical delegation tokens. In *2nd Int. Conference on Computer and Communications Security*, pages 128 – 143. Chapman and Hall, 1996.

[51] FIPS 186-2 Digital signature standard, 2000.

[52] W. Du. A study of several specific secure two-party computation problems, 2001. PhD thesis, Purdue University, West Lafayette, Indiana.

[53] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Yloenen. Simple public key certificate. `http://www.ietf.org/rfc/rfc2693.txt`.

[54] L. Eschenauer, V. D. Gligor, and J. Baras. On trust establishment in mobile ad-hoc networks. In *Proceedings of the Security Protocols Workshop*, April 2002.

[55] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th National Computer Security Conference*, 1992.

[56] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *Advances in Cryptology – Eurocrypt '04*, volume 3027 of *LNCS*, pages 1–19. Springer-Verlag, May 2004.

[57] K. B. Frikken and M. J. Atallah. Privacy preserving route planning. In *Proceedings of the 2004 ACM workshop on Privacy in the Electronic Society (WPES)*, pages 8–15. ACM Press, 2004.

[58] K. B. Frikken, M. J. Atallah, and J. Li. Hidden access control policies with hidden credentials. In *Proceedings of the 3nd ACM Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2004.

[59] K. B. Frikken, J. Li, and M. J. Atallah. Trust negotiation with hidden credentials, hidden policies, and policy cycles. In *Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS)*, 2006.

[60] C. Günther. An identity-based key exchange protocol. In *Advances in Cryptology — Eurocrypt '89*, volume 434 of *LNCS*, pages 29–37. Springer-Verlag, 1989.

[61] C. Gentry. Certificate-based encryption and the certificate revocation problem. In *Advances in Cryptology — Eurocrypt '03*, volume 2656 of *LNCS*, pages 272–293, 2003.

[62] M. Girault. Self-certified public keys. In *Advances in Cryptology — Eurocrypt '91*, volume 547 of *LNCS*, pages 490–497. Springer, 1991.

[63] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen. On private scalar product computation for privacy-preserving data mining. In C. Park and S. Chee, editors, *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2004.

[64] O. Goldreich. Secure multi-party computation, Oct. 2002. Unpublished Manuscript.

[65] O. Goldreich. *The Foundations of Cryptography*, volume 2. Cambridge University Press, 2004.

[66] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *The nineteenth annual ACM conference on theory of computing*, pages 218–229. ACM Press, 1987.

[67] S. Goldwasser. Multi-party computations: past and present. In *The sixteenth annual ACM symposium on principles of distributed computing*, pages 1–6. ACM Press, 1997.

[68] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof-systems. In *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC)*, pages 291–304, 1985.

[69] M. T. Goodrich, M. Shin, R. Tamassia, and W. H. Winsborough. Authenticated dictionaries for fresh attribute credentials. In *Proc. Trust Management Conference*, volume 2692 of *LNCS*, pages 332–347. Springer, 2003.

[70] M. T. Goodrich, R. Tamassia, N. Triandopoulos, and R. Cohen. Authenticated data structures for graph and geometric searching. In *Proc. RSA Conference—Cryptographers' Track*, volume 2612 of *LNCS*, pages 295–313. Springer, 2003.

[71] M. T. Goodrich, R. Tamassia, and D. Yao. Accredited DomainKeys: a service architecture for improved email validation. In *Proceedings of the Conference on Email and Anti-Spam (CEAS '05)*, July 2005.

[72] M. T. Goodrich, R. Tamassia, and D. Yao. Notarized federated identity management for increased trust in Web services. In *Proc Conf. on Data and Applications Security (DBSec)*, LNCS, pages 133–147. Springer, 2006.

[73] M. Gruteser and D. Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)*, 2003.

[74] M. Gruteser, G. Schelle, A. Jain, R. Han, and D. Grunwald. Privacy-aware location sensor networks. In *9th USENIX Workshop on Hot Topics in Operating Systems (HotOS IX)*, 2003.

[75] C. A. Gunter and T. Jim. Policy-directed certificate retrieval. *Software: Practice and Experience*, 30:1609–1640, September 2000.

[76] J. E. Holt, R. W. Bradshaw, K. E. Seamons, and H. Orman. Hidden credentials. In *Proceedings of the 2nd ACM Workshop on Privacy in the Electronic Society (WPES)*, Oct. 2003.

[77] Q. Huang, D. Jao, and H. J. Wang. Applications of secure electronic voting to automated privacy-preserving troubleshooting. In *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS)*, November 2005.

[78] G. Jagannathan and R. N. Wright. Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. In *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining*, pages 593–599, 2005.

[79] T. Jim. SD3: A trust management system with certified evaluation. In *Proceedings of the 2001 IEEE Symposium on Security and Privacy*, pages 106–115. IEEE Computer Society Press, May 2001.

[80] J. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Trans. Knowl. Data Eng.*, 17(1):4–23, 2005.

[81] A. Juels. Trustee tokens: Simple and practical tracing of anonymous digital cash. In *Financial Cryptography '99*, volume 1648 of *LNCS*, pages 33–43. Springer-Verlag, 1999.

[82] P. Kamat, Y. Zhang, W. Trappe, and C. Ozturk. Enhancing source-location privacy in sensor network routing. In *Proceedings of 25th International Conference on Distributed Computing Systems (ICDCS)*, 2005.

[83] L. Kissner and D. Song. Private and threshold set-intersection. In *Advances in Cryptology – CRYPTO '05*, August 2005.

[84] R. Kohlas and U. M. Maurer. Confidence valuation in a public-key infrastructure based on uncertain evidence. In *Proceedings of the Third International Workshop on Practice and Theory in Public Key Cryptography (PKC '00)*, volume 1751 of *Lecture Notes in Computer Science*, pages 93–112. Springer, 2000.

[85] C. E. Landwehr. Improving information flow in the information security market. In *Advances in Information Security – Economics of Information Security*, volume 12, pages 155–163. Kluwer Academic Publishers, 2004.

[86] M. Langheinrich. A privacy awareness system for ubiquitous computing environments. In *In 4th International Conference on Ubiquitous Computing*, 2002.

[87] J. Li, N. Li, and W. H. Winsborough. Automated trust negotiation using cryptographic credentials. In *Proceedings of 12th ACM Conference on Computer and Communications Security (CCS)*, pages 46–57, 2005.

[88] N. Li, B. N. Grosof, and J. Feigenbaum. Delegation Logic: A logic-based approach to distributed authorization. *ACM Transaction on Information and System Security (TISSEC)*, Feb. 2003. To appear.

[89] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 114–130, 2002.

[90] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed credential chain discovery in trust management. *Journal of Computer Security*, 11(1):35–86, February 2003.

[91] H. Lipmaa. Verifiable homomorphic oblivious transfer and private equality test. In *Advances in Cryptology — Asiacrypt '03*, LNCS, pages 416–433, 2003.

[92] A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology — Eurocrypt '04*, volume 3027 of *LNCS*, pages 74–90. Springer-Verlag, 2004.

[93] N. Nagaratnam and D. Lea. Secure delegation for distributed object environments. In *Proceedings of the 4th USENIX Conference on Object-Oriented Technologies and Systems (COOTS)*, April 1998.

[94] M. Naor and K. Nissim. Certificate revocation and certificate update. In *Proceedings of the 7th USENIX Security Symposium*, pages 217–228, 1998.

[95] B. C. Neuman. Proxy-based authentication and accounting for distributed systems. In *International Conference on Distributed Computing Systems*, pages 283–291, 1993.

[96] T. Okamoto and D. Pointcheval. The gap-problems: a new class of problems for the security of cryptographic schemes. In *PKC '01*, volume 1992 of *LNCS*, pages 104–118. Springer-Verlag, 2001.

[97] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology – EUROCRYPT 1999*, LNCS 1592:223–238, 1999.

[98] S. L. Pallickara, B. Plale, L. Fang, and D. Gannon. End-to-end trustworthy data access in data-oriented scientific computing. In *Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2006)*, pages 395–400, 2006.

[99] B. Pinkas. Cryptographic techniques for privacy-preserving data mining. *KDD Explorations*, 4(2):12–19, 2002.

[100] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. *Commun. ACM*, 21:120–126, 1978.

[101] P. Ruth, D. Xu, B. K. Bhargava, and F. Regnier. E-notebook middleware for accountability and reputation based trust in distributed data sharing communities. In C. D. Jensen, S. Poslad, and T. Dimitrakos, editors, *iTrust*, volume 2995 of *Lecture Notes in Computer Science*, pages 161–175. Springer, 2004.

[102] R. S. Sandhu. Lattice-based access control models. *IEEE Computer*, 26(11):9–19, November 1993.

[103] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29, Number 2:38–47, 1996.

[104] O. S. Saydjari. Multilevel security: Reprise. *IEEE Security and Privacy*, 02(5):64–67, 2004.

[105] B. Schneier. *Applied Cryptography: protocols, algorithms, and source code in C*. John Wiley and Sons, Inc., New York, 1994.

[106] C. P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 1991.

[107] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology – Crypto'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer, 1984.

[108] B. Shand, N. Dimmock, and J. Bacon. Trust for ubiquitous, transparent collaboration. *Wirel. Netw.*, 10(6):711–721, 2004.

[109] M. Shehab, E. Bertino, and A. Ghafoor. Secure collaboration in mediator-free environments. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS '05)*, November 2005.

[110] Shibboleth. `http://middleware.internet2.edu/shibboleth/`.

[111] E. Snekkenes. Concepts for personal location privacy policies. In *In Proceedings of the 3rd ACM Conference on Electronic Commerce (CEC)*, pages 48–57. ACM Press, 2001.

[112] K. R. Sollins. Cascaded authentication. In *Proceedings of 1988 IEEE Symposium on Security and Privacy*, pages 156–163, April 1988.

[113] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 44–54, May 1997.

[114] R. Tamassia, D. Yao, and W. H. Winsborough. Role-based cascaded delegation. In *Proceedings of the ACM Symposium on Access Control Models and Technologies (SACMAT '04)*, pages 146 – 155. ACM Press, June 2004.

[115] G. Theodorakopoulos and J. S. Baras. Trust evaluation in ad-hoc networks. In *WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security*, pages 1–10. ACM Press, 2004.

[116] H. Tran, M. Hitchens, V. Varadharajan, and P. Watters. A trust based access control framework for P2P file-sharing systems. In *Proceedings of the Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05) - Track 9*, page 302c. IEEE Computer Society, 2005.

[117] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *Proceedings of The 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 639–644. ACM Press, July 2002.

[118] V. Varadharajan, P. Allen, and S. Black. An analysis of the proxy problem in distributed systems. In *Proceedings of 1991 IEEE Symposium on Security and Privacy*, pages 255–275, 1991.

[119] W. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, pages 147–160. IEEE Press, May 2004.

[120] W. H. Winsborough and N. Li. Safety in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*. IEEE Computer Society Press, May 2004.

[121] W. H. Winsborough, K. E. Seamons, and V. E. Jones. Automated trust negotiation. In *DARPA Information Survivability Conference and Exposition*, volume I, pages 88–102. IEEE Press, Jan. 2000.

[122] L. Xiong and L. Liu. A reputation-based trust model for Peer-to-Peer ecommerce communities. In *2003 IEEE International Conference on Electronic Commerce (CEC 2003)*, pages 275–284. IEEE Computer Society, 2003.

[123] R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man and Cybernetics*, 18(1):183–190, 1988.

[124] N. Yankelovich, W. Walker, P. Roberts, M. Wessler, J. Kaplan, and J. Provino. Meeting central: making distributed meetings more effective. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work (CSCW '04)*, pages 419–428, New York, NY, USA, 2004. ACM Press.

[125] A. C. Yao. How to generate and exchange secrets. In *Proceedings of the 27th IEEE Symposium on Foundations of Computer Science*, pages 162–167. IEEE Computer Society Press, 1986.

[126] D. Yao, N. Fazio, Y. Dodis, and A. Lysyanskaya. ID-based encryption for complex hierarchies with applications to forward security and broadcast encryption. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 354 – 363. ACM Press, 2004.

[127] D. Yao, K. B. Frikken, M. J. Atallah, and R. Tamassia. Point-based trust: Define how much privacy is worth. In *Proc. Int. Conf. on Information and Communications Security (ICICS)*, volume 4307 of *LNCS*, pages 190–209. Springer, 2006. Best Student Paper Award.

[128] D. Yao and R. Tamassia. Cascaded authorization with anonymous-signer aggregate signatures. In *Proc. IEEE Systems, Man and Cybernetics Information Assurance Workshop (IAW)*, pages 84–91, June 2006.

[129] D. Yao, R. Tamassia, and S. Proctor. On improving the performance of role-based cascaded delegation in ubiquitous computing. In *Proceedings of IEEE/CreateNet Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm '05)*, pages 157–168. IEEE Press, September 2005.

[130] D. Yao, R. Tamassia, and S. Proctor. Private distributed scalar product protocol with application to privacy-preserving computation of trust. In *Proc. IFIPTM Joint iTrust and PST Conferences on Privacy, Trust Management and Security*, July 2007.

[131] M. Yokoo and K. Suzuki. Secure multi-agent dynamic programming based on homomorphic encryption and its application to combinatorial auction. In *First joint International Conference on Autonomous Agents and Multiagent Systems (AAMAS-2002)*, pages 112–119. ACM Press, 2002.

[132] T. Yu, X. Ma, and M. Winslett. PRUNES: An efficient and complete strategy for automated trust negotiation over the internet. In *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*, pages 210–219, November 2000.

[133] T. Yu and M. Winslett. A unified scheme for resource protection in automated trust negotiation. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 110–122. IEEE Computer Society Press, May 2003.

[134] T. Yu, M. Winslett, and K. E. Seamons. Interoperable strategies in automated trust negotiation. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS '01)*, pages 146–155. ACM Press, Nov. 2001.

[135] X. Zhang, S. Oh, and R. Sandhu. PBDM: A flexible delegation model in RBAC. In *Proceedings of the ACM Symposium on Access Control Models and Technologies*, pages 149 – 157. ACM Press, 2003.

[136] C. Zouridaki, B. L. Mark, M. Hejmo, and R. K. Thomas. A quantitative trust establishment framework for reliable data packet delivery in MANETs. In V. Atluri, P. Ning, and W. Du, editors, *Proceedings of the Third ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN)*, pages 1–10. ACM, 2005.