

OC: A System for Open Collaborations

Zhengyi Le Yi Ouyang James Ford Fillia Makedon
DEVLAB, Department of Computer Science
Dartmouth College, Hanover, NH 03755, USA
{zhengyi.le, yi.ouyang, james.ford, fillia.makedon}@dartmouth.edu

Abstract

Existing collaboration systems have a fundamental limitation: they assume that collaborative groups consist of only of individuals known personally by someone with the authority to add to the group membership. The OC (Open Collaboration) research prototype presented in this paper addresses this problem by introducing trust negotiation into collaboration systems. Our approach separates the profiles used by groups and individuals in order to let entities control their privacy. We extend the family of RT languages to specify the requirements of assigning a role and describe how to create and maintain a collaboration. In addition, we design two modes for disseminating group profiles and classify three kinds of identities according to their sensitivity.

1 Introduction

The extensive use of the Internet and other networks provides a strong basis for global collaboration among large numbers of diverse entities. Collaborative applications, such as those providing remote education programs, supporting virtual patent-research team, and conducting collaborative computing and data sharing, have been calling for security services. Existing research focuses on scalable and fault-tolerant group key management protocols [2, 21, 32], data confidentiality and integrity [1, 3, 18, 22], large-scale and decentralized trust management for access control [15, 14, 16, 17, 13], and public-key support for the key-name binding problem [8, 7].

One important consideration that is often overlooked is that often entities that do not know each other may find they want to work together. Trust management gives access to requesters according to their attributes instead of their identities. In this way, a resource holder can answer a requester who it does not know, basing the access it grants on a deductive process [14]. Trust negotiation supports credential exchange between two strangers who wish to build mutual

trust [35, 36, 33, 34, 23, 24, 30, 31, 29, 27, 26, 28, 6, 5]. However, trust management alone deals with neither the certificate/credential exchange procedures, nor the creation and maintenance of a dynamic collaboration. Also, existing trust negotiation approaches are designed only for the two party case. Current commercial applications, such as Groove, Lotus Notes, Open-Xchange, and eGroupWare, only support certain fixed roles, require a central server, and do not support sophisticated enrollment and access policies.

This paper presents a research prototype called OC (Open Collaboration). In open environments, there are many diverse entities, most of them independent and autonomous. OC is designed to allow strangers to join a collaboration through trust negotiation and maintain dynamic collaborations in a purely peer-to-peer fashion, a combination that has not been addressed before. Our approach extends existing two-party trust negotiation to multi-party agreement in an open collaborative group on the basis of Li's RT family of trust management (TM) languages. Furthermore, since policies play a critical role in secure systems, OC supports three kinds of policies: role appointment policies, data (file and communication) access policies, and certificate release policies.

OC distinguishes itself from existing work by attempting to satisfy the following goals simultaneously.

- Allowing strangers to join a multi-party collaboration with mutually agreed on privacy and security policies.
- Giving users privacy they can control and security they can understand.
- Removing the need for a server and central administration.

Section 2 provides an overview of related work and examines it in the context of our goals. In Section 3, we present a security architecture, OC, for open collaborations. Section 4 gives our conclusions and lists some areas for future research.

2 Related Work

Collaboration research has now been quite active in the past decade; industrial software has been available for years and more tools are continually being developed.

Winsborough *et al.* first introduced the notion of automated trust negotiation and gave an architecture for managing the exchange of credentials between two strangers for the purpose of determining bilateral trustworthiness [29]. Researchers have explored automated trust negotiation systems in [35, 36, 33, 23, 6]. Subsequent research investigated related privacy and security issues and provided new approaches [26, 28, 34, 24, 5]. In addition, Li *et al* proposed an RSA-based protocol and an ID-cryptography-based protocol to address the cyclic interdependency problem in automated trust negotiation [12].

Li and Mitchell presented a role-based trust-management framework called *RT*, in [14]. *RT* provides policy language, semantics, a deduction engine, and pragmatic features to address large-scale and decentralized access control and authorization problems.

The work most related to OC is by Ellison and Dohrmann [8] and Nita-Rotaru and Li [19]. Ellison and Dohrmann presented a security architecture called NGC (Next Generation Collaboration) [8]. They gave a process requiring human and machine interaction for binding a name to a public key and used SDSI names for both groups and individuals. However, NGC's flexibility was limited by the fact that new members could only be added by invitation, and that this could only be done by a subset of the current members pre-authorized to issue invitations. This means that in general only parties known to the core users could participate. Most recently, Nita-Rotaru and Li presented a framework for role-based access control in group communication systems [19]. They identified the set of all possible group operations that can be controlled and defined the group policy as a mapping between roles and operations using context as constraints. However, theirs is a centralized system, and it does not provide any mechanisms for allowing a stranger to join a collaboration.

Recently the commercial product Groove has provided a collaborative community for users. Users can create "workspaces" and invite people to join, after which members of the workspace can share files and manage projects together. Groove supports only three fixed roles—Manager, Participant, and Guest—and new members can join only by being invited. Groove is not a pure peer-to-peer system: it needs the help of a reply server. Open-Xchange (OX) and eGroupWare are open source communication software packages. Both are web-based applications, and a centralized server is thus required. They do not support roles, and users can be added only by the system administrator.

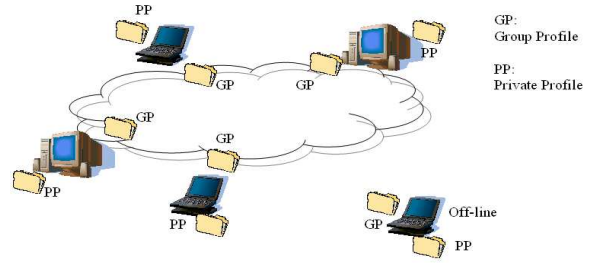


Figure 1. Group Profiles vs. Private Profiles

3 Our Security Architecture for Open Collaboration

Although some proposals exist today for systems for the negotiation and management of trust, none of them addresses in a comprehensive way the problem of how to build and manage an open collaboration among diverse, independent, and autonomous entities.

Requirements for secure collaboration are as follows: administration of group membership, data sharing with associated access policies, and secure communications among members of the group.

3.1 Group Profiles vs. Private Profiles

In order to allow effective collaborations, each entity must be kept updated with the information about every collaboration it is taking part in. We call the collection of group information the *group profile*. Each member may be allowed to see only a partial view of this group profile. The group profile is synchronized among group members. Since OC is a purely peer-to-peer application, each entity is responsible for helping its neighbor nodes update the group profile according to associated access policies when they come into contact (see Figure 1). The group profile provides comprehensive information on this collaboration, including the following items:

- a group name and mission description
- current time
- group state
- join requirements and benefits
- a list of all group members with associated roles
- a list of the names of all shared files with associated policies

OC is designed to work in open environments, where every entity may have its own set of goals and priorities. Each entity needs a *private profile* to manage the personal information that is related to its on-going collaborations. The private profile contains the local data for each entity, which is created, accessed, controlled and managed only by the entity itself. The private profile includes the following items:

- memberships of some collaborations
- personal certificates with release policies
- shared files with access policies created by this user
- local group policies
- local strategies

The notion of group profile and private profile is different from the notion of contexts in [19] since their system uses a client-server architecture and both the client and group contexts are stored on the server side while our system runs in a peer-to-peer fashion.

In OC, access control is enforced by file owners and their trusted entities. The list of the names of shared files with associated policies is synchronized; the files themselves need not be synchronized, but may be synchronized for efficiency in some cases. An entity can download a file, when needed, from its peers who already have a copy of it, if this entity satisfies the associated access policy. One reason for not contacting the source might be that some member machines may be mobile devices that are limited by memory, power, or bandwidth. The source could also be temporarily offline or overloaded. In contrast, NGC [8] synchronizes the list of the names of shared files, without associated policies since it does not support the latter.

3.2 The Creation of a Collaboration

Any entity can create a collaboration and publish portions of the group profile, which are then publicly accessible, on bulletin services or by other means. There are two phases in creating a collaboration: (1) making a group profile and (2) recruiting members.

Every collaboration is task-oriented. The OC system provides some group profile templates for general-purpose collaborations that group creators can use. A creator could follow the general-purpose templates or create his own. Generally, a creator must complete the following steps: (1) give the description of the mission of this collaboration; (2) define roles of this collaboration and the requirements for application; (3) list all the initial members and their roles, and also the names of the initial files and their associated policies, if any; (4) publish/broadcast the publicly accessible parts of the group profile.

OC supports two modes by which a new entity can join a group: *by invitation* or *on its own*. Inviting is a safe way to introduce a new member; however, this limits the breadth of collaborations for those who are qualified but have no acquaintances in this collaboration. OC also allows an entity to join an open collaboration on its own. This might subject the collaboration to certain attacks, so OC requires that an entity who wishes to join a group perform trust negotiation with a senior member who has the privilege of referring others to the collaboration group for approval. The following is a summary of OC's two modes for adding members to a group:

1. **By invitation.** An inviting party is a member of this group and has been appointed to a role which has the privilege of inviting. There are four phases during invitation. In phase 1, the inviter must decide what kind of invited party should be invited, i.e., what kind of attributes the invitee should have. In phase 2, the inviting party goes over his local history records or public information services to find such a invitee. In phase 3, the inviter sends an invitation to the invitee. This invitation includes benefits and requirement of joining this collaboration. In phase 4, the invitee replies to the invitor with a "Yes" or "No". If "Yes", his reply will include his credentials satisfying these requirements, or a counter request for some credentials from the inviter if this invitee's credential release policies so require.
2. **On its own.** There are 4 phases in this procedure. In phase 1, the entity finds interesting information on public bulletins. In phase 2, the entity sends a request to a senior group member he chooses. In phase 3, the two parties perform trust negotiation. If this trust negotiation succeeds, he will be referred into this collaboration with his true name or a pseudonym covered by his referrer with the approval of the group policy and other members' local group policies.

The addition of new members can also involve many parties, rather than only two. The enrollment requirements might ask for the proof of some attributes and the approval of a majority of senior members. In this case, enrollment is not complete after the steps described above for either mode. After introduction to this collaboration either by invitation or on its own, there could be three possible results:

- All senior members approve, and the new entity joins.
- A majority of senior members approve, and the new entity joins. However, those in the minority of members may take some corresponding actions, such as changing local group policies or quitting this collaboration.

- The majority of senior members object or abstain, and the new entity fails to join.

After being invited or referred, and before being confirmed, information on the new entity appears in the group profile. Members see its presence, its referer or inviter, and a pending tag. If a member completely trusts the referrer according to its local group policy, he will automatically approve the enrollment of the new entity. If he partially trusts the referrer, he may initiate a trust negotiation with the new entity and then decide. This trust negotiation may skip some steps according to how much the referrer is trusted. If he does not trust the referrer, he may initiate a complete trust negotiation with the new entity and then decide. The counting of approvals and objections can be implemented in either of the following two ways: (1) a specific role, such as the secretary or the referrer/inviter, collects results and then announces whether to let him in, or (2) members broadcast their results as a part of the group profile. After a period of time, each member will know the final result through the group profile. The first approach is easier to implement and more efficient but is not transparent to all members. The referrer or inviter will take full responsibility for collecting individual results. The second is transparent but costs more in both communications and time. After counting, the status of the new entity will be set to “approved” or “rejected”.

3.3 The Maintenance of a Collaboration

Once the group profile for a given mission has been specified, the creator releases it to other on-line entities according to the policies it specifies. OC supports two different methods for updating group profiles, *passive* and *active*, which are described in the following:

- In the passive mode, every on-line entity passively receives group profile updates from other entities periodically. In other words, every group member sends its group profile to other on-line entities at regular intervals of time. Different entities may see different portions of the same group profile on the basis of their roles and the associated access control policies in the group profile.
- Active mode is the complementary mechanism for the passive mode. In the active mode, an entity sends update requests to other on-line entities so that they can update its group profile. OC is designed for open environments, where an entity might show up at any time and then disconnect after several seconds. Under these conditions, the active mode is necessary for group profile updates.

In either mode, OC is a purely peer-to-peer application. An entity uploads or downloads its group file to or from

other peers according to the policies specified in the group profile. The receiving entity decides whether to update its old version according to the timestamp on the new group file and the role of the entity it is communicating with. Roles provide an indication of trust relationships. In order to obtain a role, an entity must pass the trust negotiation with multi-party approval. This mechanism is quite like what happens in the real world. For example, suppose a college student traveling in Africa wants to update information on the academic calendar at his school. He could call a classmate or department secretary; however, the secretary may not be available at that time, and his phone connection may be limited by time constraints. So, suppose he calls his classmate. His classmate only can give him the information she knows. If the department secretary were available, she may have more or newly released information about their academic calendar.

OC supports dynamic collaborations. When, during the performance of a task, some policies or roles need to be changed, a senior member can propose such a change. The proposal will show up in the group file, where only authorized or senior members can see it.

Discussion is an optional follow-up to any proposal. Authorized members can send their approvals or objections to a proposer, or attach their votes to the group profile, just as in recruiting a new member.

Private profiles are controlled entirely by their owners, and are never released to others. They can be updated or changed by their owners at any time.

3.4 The Identities of Members

The identities of members in OC are just like those in real-world groups. For example, when a person is introduced into a group of people, those people can identify him with his true name, or “our guest”, or “the friend of some introducer”. OC provides the mapping from these three kinds of identities in the real world to the digital world.

- True name. Identity certificates are the counterparts of true names. These certificates could be, for example, in X.509 [11] or PGP [4] format. In the X.509 format, there is a field called SUBJECT that holds the name of the entity whose public key is being certified, *i.e.*, the name of the certificate holder. In addition, there is another optional field called SUBJECTUNIQUEIDENTIFIER, which uniquely identifies the subject of this certificate. The PGP format also binds public keys to unique global names.
- Anonym. Entities can join anonymously, either to participate in collaborations that only consider the attributes they have, instead of who they are, or as guests

or spectators. They can be identified by either temporary SPKI certificates [9], since an SPKI public key or its fingerprint can function as an identity, or temporary X.509 certificates with pseudonyms. An entity takes care of his anonymity by himself. In other words, if he chooses to join anonymously, he must get his temporary identity certificate and anonymous attribute certificates [10] (or we can use the method given in [5]) beforehand. Otherwise, he will not be able to prove his attributes without losing his anonymity.

- Proxy name. Entities could join with proxy certificates [25] or SPKI/SDSI [9, 20] certificates. In the case of proxy certificates (which are an X.509 variant), the introducer will sign a new short-lived public key for the new entity. In the case of SPKI/SDSI, the introducer is the only authority who binds the names of his introducees with their public keys. Other entities are in a position to question these assignments or bindings. The introducer, say “Carol”, can screen the true name of her introducee by defining “Carol’s friend”.

As discussed above, note that one of the requirements for performing in open environments is the ability to process different kinds of certificates, such as PGP, X.509, and SPKI/SDSI. Entities must have the required certificates ready with the help of CAs or their introducers before they formally enroll in a collaboration.

3.5 The Roles of Members

OC supports the RT family of TM languages in [14]. OC adds a new language, RT^A , to the existing RT_0 , RT_1 , RT^T , and RT^D languages in the RT family. This RT^A describes the requirements of applying for and assigning a role. In general, there are three kinds of requirements: attribute requirements, ID requirements, and majority requirements. The formal specifications for role assignment policies with RT^A are as follow, where R is a specific role, A is the set of all attributes, I is the form of identity, and \odot and \otimes are two logic operators, denoting *or* and *and*.

- $R \leftarrow (A_1 \odot \dots \odot A_i) \otimes (A_{i-1} \odot \dots \odot A_j) \otimes \dots$
where $A_i \in A$. The head is a role; the body is the attribute requirement in conjunctive normal form of propositional logic.
- $R \leftarrow I.i$
where $I.i$ is a specific kind of identity: identity certificate, temporary anonymous certificate, or a proxy certificate. We may ask that some roles be associated with true names while others are not. For example, in the case of a remote education program, guests might be anonymous while lecturers might have to show their real names.

- $R \leftarrow (\alpha_1 R_1 \odot \dots \odot \alpha_i R_i) \otimes (\alpha_{i-1} R_{i-1} \odot \dots \odot \alpha_j R_j) \otimes \dots$

where $\alpha_i \in [0, 1]$ is the percentage of approvals and R_i is a role. For example, in order to obtain the role of a department chair, an entity might need 80% approval from the faculty and 50% approval from the graduate students.

In addition, there are two other kinds of policies: credential release policies and file sharing policies. They directly inherit from the RT_1^{DT} in [14], where RT_1^{DT} is the combination of RT_1 , RT^D , and RT^T . Roles can be viewed as attributes of an entity. Adopting role based access control on files and credentials already implies that accesses will be trust-based since obtaining a role already has required certificates verified by entities in charge. Every entity in an open environment is self-governed and self-willed. So, it is quite reasonable that an entity completely controls when and to whom it will release its identity or credentials in order to join collaborations it wants, and arbitrarily decides what kinds of entities can do what kinds of operations on its files. It can do this by writing corresponding policies on its credentials and shared files. Thus, the privacy of each entity is something it can control itself, and the restrictions on the use of shared files are defined by file owners.

4 Conclusions and Future Work

This paper introduces trust negotiation into collaboration systems so that our system can perform in a pure peer-to-peer network without the assumption that collaborative groups are made up only of individuals known personally by someone with the authority to add to the group membership. This assumption is a major concern for existing collaboration systems. We separate the group profile and private profile to let entities control their privacy, extend the RT languages to specify the requirements for assigning a role, and describe how to create and maintain a collaboration. By using the OC approach, collaboration systems will be able to support a wider variety of collaborations among entities who previously would not have been supported.

There are several things to be addressed in future work, membership revocation, group dissolution, and more sophisticated group operations, such as group merging and nesting. These are related to key and certificate management in collaborations. Furthermore, designing and implementing a reconciliation checker is also important, especially in systems without centralized control. The emphasis will be on making every entity comply with the policies made by others, a need also related to key management. So, key and certificate management for open collaboration will be our next focus. In addition, policy writing is an open problem. Providing an easily understood user interface to

help users write policies and recommending general policies for familiar credentials are important problems that we will also investigate.

Acknowledgments

The authors would like to thank the previous anonymous reviewers for their valuable comments and suggestions. This project was supported under Award number 2000-DT-CX-K001 from the U.S. Department of Homeland Security, Science and Technology Directorate. Points of view in this document are those of the author(s) and do not necessarily represent the official position of the U.S. Department of Homeland Security or the Science and Technology Directorate.

References

- [1] D. A. Agarwal, O. Chevassut, M. R. Thompson, and G. Tsudik. An integrated solution for secure group communication in wide-area networks. In *Proceedings of the 6th IEEE Symposium on Computers and Communications*, July 2001.
- [2] Y. Amir, Y. Kim, C. Nita-Rotaru, J. Schultz, J. Stanton, and G. Tsudik. Secure group communication using robust contributory key agreement. *The IEEE Transactions on Parallel and Distributed Systems*, 15(5):468–480, May 2004.
- [3] Y. Amir, C. Nita-Rotaru, J. Stanton, and G. Tsudik. Scaling secure group communication systems: Beyond peer-to-peer. In *Proceedings of the 3rd DARPA Information Survivability Conference and Exposition (DISCEXIII)*, April 2003.
- [4] D. Atkins, W. Stallings, and P. Zimmermann. PGP Message Exchange Formats. *RFC 1991*, 1996.
- [5] E. Bertino, E. Ferrari, and A. C. Squicciarini. Privacy-Preserving Trust Negotiation. In *Proceedings of the 4th Workshop on Privacy Enhancing Technologies.*, May 2004.
- [6] E. Bertino, E. Ferrari, and A. C. Squicciarini. Trust-X: A Peer-to-Peer Framework for Trust Establishment. *IEEE Trans. Knowl. Data Eng.*, 16(7):827–842, 2004.
- [7] S. Dohrmann and C. Ellison. Public-key support for collaborative groups. In *Proceedings of the 1st Annual PKI Research Workshop*, pages 139–148, 2002.
- [8] C. Ellison and S. Dohrmann. Public-key support for group collaboration. *ACM Trans. Inf. Syst. Secur.*, 6(4):547–565, 2003.
- [9] C. M. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. *RFC 2693*, 1999.
- [10] S. Farrell and R. Housley. An Internet Attribute Certificate Profile for Authorization. *RFC 3281*, 2002.
- [11] R. Housley, W. Ford, T. Polk, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. *RFC 2459*, 1999.
- [12] N. Li, W. Du, and D. Boneh. Oblivious signature-based envelope. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 182–189, New York, NY, USA, 2003. ACM Press.
- [13] N. Li and J. C. Mitchell. DATALOG with Constraints: A Foundation for Trust Management Languages. In *Proceedings of Practical Aspects of Declarative Languages, 5th International Symposium (PADL 2003)*, volume 2562 of *Lecture Notes in Computer Science*, pages 58–73. Springer, Jan 2003.
- [14] N. Li and J. C. Mitchell. RT: A Role-based Trust-management Framework. In *The Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, *IEEE Computer Society Press*, pages 201–212, 2003.
- [15] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a Role-Based Trust-Management Framework. In *IEEE Symposium on Security and Privacy*, pages 114–130, 2002.
- [16] N. Li, W. H. Winsborough, and J. C. Mitchell. Beyond Proof-of-Compliance: Safety and Availability Analysis in Trust Management. In *IEEE Symposium on Security and Privacy*, pages 123–139, 2003.
- [17] N. Li, W. H. Winsborough, and J. C. Mitchell. Distributed Credential Chain Discovery in Trust Management. *Journal of Computer Security*, 11(1):35–86, 2003.
- [18] P. McDaniel, A. Prakash, and P. Honeyman. Antigone: A flexible framework for secure group communication. In *Proceedings of the 8th USENIX Security Symposium*, pages 99–114, August 1999.
- [19] C. Nita-Rotaru and N. Li. A Framework for Role-Based Access Control in Group Communication Systems. In *Proceedings of 2004 International Workshop on Security in Parallel and Distributed Systems*, September 2004.
- [20] R. Rivest and B. Lampson. SDSI - A Simple Distributed Security Infrastructure. <http://theory.lcs.mit.edu/~rivest/sdsi10.html>, 1996.
- [21] O. Rodeh, K. P. Birman, and D. Dolev. Optimized Rekey for Group Communication Systems. In *Proceedings of the Network and Distributed System Security Symposium*, 2000.
- [22] O. Rodeh, K. P. Birman, and D. Dolev. The architecture and performance of security protocols in the ensemble group communication system. *ACM Trans. Inf. Syst. Secur.*, 4(3):289–319, 2001.
- [23] K. E. Seamons, M. Winslett, T. Yu, B. Smith, E. Child, J. Jacobson, H. Mills, and L. Yu. Requirements for Policy Languages for Trust Negotiation. In *3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, pages 68–79, 2002.
- [24] K. E. Seamons, M. Winslett, T. Yu, L. Yu, and R. Jarvis. Protecting Privacy during On-Line Trust Negotiation. In *Proceedings of the 4th Workshop on Privacy Enhancing Technologies.*, pages 129–143, 2002.
- [25] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson. Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. *RFC 3820*, 2004.
- [26] W. H. Winsborough and N. Li. Protecting sensitive attributes in automated trust negotiation. In *Proceedings of the 2002 ACM Workshop on Privacy in the Electronic Society (WPES 2002)*, pages 41–51, 2002.
- [27] W. H. Winsborough and N. Li. Towards Practical Automated Trust Negotiation. In *3rd International Workshop on Policies for Distributed Systems and Networks (POLICY 2002)*, pages 92–103, 2002.

- [28] W. H. Winsborough and N. Li. Safety in Automated Trust Negotiation. In *IEEE Symposium on Security and Privacy*, pages 147–160, 2004.
- [29] W. H. Winsborough, K. Seamons, and V. Jones. Automated Trust Negotiation. *DARPA Information Survivability Conference and Exposition (DISCEX'00)*, 1:64–73, 2000.
- [30] M. Winslett. An Introduction to Automated Trust Establishment. In *GI Jahrestagung*, pages 106–113, 2002.
- [31] M. Winslett, T. Yu, K. E. Seamons, A. Hess, J. Jacobson, R. Jarvis, B. Smith, and L. Yu. Negotiating Trust on the Web. *IEEE Internet Computing*, 6(6):30–37, 2002.
- [32] C. K. Wong, M. G. Gouda, and S. S. Lam. Secure Group Communications Using Key Graphs. In *Proceedings of the ACM SIGCOMM'98*, pages 68–79, 1998.
- [33] T. Yu and M. Winslett. A Unified Scheme for Resource Protection in Automated Trust Negotiation. In *IEEE Symposium on Security and Privacy*, pages 110–122, 2003.
- [34] T. Yu and M. Winslett. Policy migration for sensitive credentials in trust negotiation. In *WPES '03: Proceedings of the 2003 ACM workshop on Privacy in the electronic society*, pages 9–20, New York, NY, USA, 2003. ACM Press.
- [35] T. Yu, M. Winslett, and K. E. Seamons. Interoperable strategies in automated trust negotiation. In *CCS '01: Proceedings of the 8th ACM conference on Computer and Communications Security*, pages 146–155, New York, NY, USA, 2001. ACM Press.
- [36] T. Yu, M. Winslett, and K. E. Seamons. Supporting structured credentials and sensitive policies through interoperable strategies for automated trust negotiation. *ACM Trans. Inf. Syst. Secur.*, 6(1):1–42, 2003.