Distributed Access Control in CROWN Groups

Jinpeng Huai, Yu Zhang, Xianxian Li
Dept. of Computer Science and Technology
Beihang University
Beijing, 100083, P.R.China
huaijp@buaa.edu.cn

Yunhao Liu
Dept. of Computer Science
Hong Kong Univ. of Science and Technology
Clearwater Bay, Kowloon, Hong Kong
liu@cs.ust.hk

Abstract

Security in collaborative groups is an active research topic and has been recognized by many organizations in the past few years. In this paper, we propose a fine-grained and attribute-based access control framework for our key project, CROWN grid. To avoid single point of failure and enhance scalability of the system, we employ a distributed delegation authorization mechanism. We successfully implement our proposed access control in CROWN grid, and evaluate this approach by comprehensive experiments.

Keywords: CROWN group, attribute-based access control, distributed delegation authorization, voting

1. Introduction

The emergence of decentralized and dynamic cooperative applications has recently gained significant attention due to its great potential for sharing a huge amount of resources with millions of users over a wide-area network [6, 7, 9, 15, 20, 21]. A large class of applications including file sharing, grid computing, multi-party conferencing, Internet e-commerce, benefit from a cooperative infrastructure. Collaborative settings may be synchronous or asynchronous, and communication models vary from one-to-many, few-to-many, to any-to-any.

In a complex grid environment, with the rapid growth of cooperation, dynamic peers join/leave and therefore the evolvement of the mesh network is free and uncontrolled. It is of great importance for multiple self-organizing peers with a common set of services aggregated in a controlled manner to accomplish their collective goals. The concept of collaborative peer groups [2, 8, 26] is introduced to refer to such cooperation applications.

The research described in this paper is part of a

larger project known as CROWN (China R&D Environment Over Wide-area Network) [12]. Started in late 2003, CROWN aims to empower in-depth integration of resources and cooperation of researchers nationwide and worldwide. As illustrated in Fig.1, a number of universities and institutes, such as THU (Tsinghua University), PKU (Peking University), CAS (China Academy of Sciences) and BHU (Beihang University) across several cities in China have joined CROWN. Through the Computer Network Information Centre of CAS, CROWN is connected to several popular grid systems including GLORIAD and PRAGMA. Lots of applications in different domains have been deployed into CROWN grid, such as gene comparison in bioinformatics, climate pattern prediction for environment monitoring, etc. The main research objective in this paper is to group home user resources with a robust, scalable, and secure grid middleware infrastructure in a distributed manner.

Consider the following scenario in the CROWN system. Multiple peers from THU, BHU, and CAS, are jointly working on an Air Pollution Monitoring project. These peers construct an APMGroup to share documents and resources. Usually, all peers first negotiate a Collaborative Policy Instance (CPI) to satisfy multiple peers' security requirements. Here we assume the existence of CPI, and the policy negotiation details [22, 28] are beyond the scope of this paper. Four roles are defined in APMGroup, *group authority*, *group member*, *director*, and *researcher*. If a BHU student would like to join the group, the CPI requires at least 40% votes from existing group members, and more than half of those votes must be *yes*. Additionally, large amounts of sensitive experiment data should be stored in a stand alone peer, and the data must be unable to be modified by one single researcher in the group. Thus, an access control framework is needed.

Figure 1. Overview of CROWN Grid

Security in collaborative groups is an active research topic and has been recognized by many organizations in the past few years. Most existing group access control mechanisms adapt centralized architecture, and authorization decisions are made based on requester identities [13]. In a distributed collaborative environment such as CROWN, peers are often dynamic and unknown to each other, which makes centralized and identity based access control less effective. Some trust management systems, granting certain permissions to the subject using credential chains, fail to support distributed environments [5].

In this paper we propose a fine-grained and attribute-based access control framework for CROWN groups. The policy model extends the role-based trust management language RT [16-18] to satisfy security requirements of the CROWN grid. The major contributions of this work are as follows:

1. To avoid single point of failure and enhance scalability of the system, instead of using a centralized model [13], we employ a distributed delegation authorization mechanism. Multiple authorities could exist in this design, reducing both the overhead and the response time of CROWN group authority.
2. Existing approaches fail to deal with the dynamics of the peers. Worse, peers are often unknown to each other, making identity-based approaches ineffective. Our framework addresses the two issues by employing an attribute-based approach. A voting mechanism is also introduced into accepting new members and granting permissions.
3. Sensitive experiment data generated during the collaboration should not be unilaterally modified by any single user. Our framework provides a secure cooperative process for multiple peers.
4. We successfully implement our distributed access control mechanism in CROWN grid. We introduce our implementation experiences and experimental results.

The rest of this paper is organized as follows. Section 2 introduces related work. Section 3 discusses the access control policy model in CROWN groups. We present a formal joint authorization protocol in Section 4. Section 5 describes a secure cooperation process. Section 6 presents performance evaluations. We conclude this work in Section 7.

2. Related work

Many approaches have been taken on security issues in collaborative environments [10, 11, 13, 14, 19, 25, 29]. Gothic [13] provides a security service for IP-Multicast, which considers receiver access control. It employs an external access control server providing authentication and authorization based on PKI certificates. Antigone [10] includes a flexible policy framework for secure group communication and defines policies about re-keying, membership, and application messages. Antigone employs a centralized access control approach in which member access is mediated by a so called Session Leader. However, it is not designed for Grid and P2P systems.

Sconce [14, 25] presents an admission control framework on Gnutella like P2Ps [1]. It proposed three types of admission policy, including access control list APT_ACL, a centralized authority APT_GAUTH, and group members APT_GROUP. A group membership certificate can be issued to a peer under multi-voting schemes. However, Sconce treats peer groups as a flat structure without hierarchy where all peering nodes have identical responsibilities. Lacking the attribute of peers, it cannot simplify authorization in collaborative environments, and is not scalable.

JXTA [2, 8], an open-source project initiated by SUN, recently proposed a security mechanism based on PKI certificate [4]. Intergroup [3] provides access control using an authorization service called Akenti [27], which relies on X.509 identity certificate. All group members register with the authorization service off-line to obtain a membership certificate signed by the Akenti Server. Intergroup provides a coarse granularity for access control.

Spread [19] introduces roles into group. It is a hierarchical client-server architecture where an expensive distributed protocol runs among a set of servers, providing services to the clients. It does not discuss distributed authorization in detail.

Our work focuses on a decentralized model in grid collaborative systems. It is a distributed delegation authorization mechanism. By considering joint authorization and secure cooperation under voting schemes, security for communication and sharing of sensitive data among grouped peers are provided.

3. Access control design overview

In this section, we first introduce roles and permissions in CROWN groups, and then form the access control policy model. An instance is given to illustrate the model.

3.1. Roles and permissions

Many sensitive operations and services need access control [10, 19]. For example, CROWN group has following sensitive services.

- Group Creation Service: to allow a peer to create a CROWN group;
- Content Publishment Service: to allow a peer to publish content within the group;
- Rendezvous Service: to allow a peer to act as a rendezvous within the group;
- Policy Negotiation Service: to allow a peer to negotiate security policy with other peers.

Here we define two kinds of roles: *group role* and *application role*. Group roles are predefined by CROWN groups, and application roles are defined according to different collaborations. Multiple different authority peers may exist in each group, so that the overhead of group authority could be balanced. Group authorities may create or modify group policy template, create or destruct group, and accept or reject new members, etc. Group members may negotiate/modify the group policy, join/leave a group, send/receive messages, and negotiate/access the group key, etc.

3.2. Access control policy model

Before access control is implemented, peers need to be authenticated. General authentication mechanisms include username/password, Kerberos [24], and X.509 [27]. Since peers are often dynamic and unknown to each other, our framework adopts credential in trust management [5, 17] as authentication method. Additionally, what permissions a peer is allowed to carry out depends on the roles and environment factors [23]. To satisfy diverse environments, we introduce the notion of contexts. Group contexts consist of a set of name/value pairs, providing environmental information such as current time and group state.

We propose an access control policy model for grid collaborative systems, which defines the relations of roles and permissions, introduces six credentials from RT, and describes admission and removal policy of roles. Elements of access policy model are as follows.

- (1) *C*: Context, defines group contexts, which include variables and their values.
- (2) *OBJ*: Object Set, $OBJ = \{obj_1, obj_2, \dots, obj_n\}$.
- (3) *OP*: Operation Set, $OP = \{op_1, op_2, \dots, op_n\}$.
- (4) *P*: Permission Set, $P = OP \times OBJ \times C$, that is, $P = \{ \langle op_i, obj_j, c_i \rangle \mid op_i \in OP, obj_j \in OBJ, c_i \in C \}$.

- (5) *RoleTerm*: It is defined as $A.r(h_1, \dots, h_n)$, where *A* is entity name(optional), *r* is role name. A *RoleTerm* may include zero or more restriction parameters h_i . For example, a student of BHU registered after the year of 2000 could be described as $BHU.student(since=2000)$.
- (6) *R*: *RoleTerm* Set, $R = SR \cup AR$. *SR* and *AR* are all *RoleTerm* Set, and *SR* is group roles set, while *AR* is application roles set.
- (7) *PA*: Relations of *R* and *P*, $PA \subseteq R \times P$.
- (8) *Credential*: Our system introduces six kinds of *Credential* from RT [17], each *Credential* has a head part and body part as follows (R_i is *RoleTerm*, *D* is entity).
 - $R \leftarrow D$: The body part consists of a simple entity *D*, which means *D* is the member of *R*.
 - $R \leftarrow R_l$: The body part consists of a *RoleItem* R_l , which means the principal set of *R* contains the principal set of R_l .
 - $R \leftarrow R_1 \cap \dots \cap R_k$: The body part consists of an Intersection element, which means the principal set of *R* contains the principal set of $R_1 \cap \dots \cap R_k$.
 - $R \leftarrow R_l.R_2$: The body part consists of a LinkRole element, which means the principal set of *R* contains the principal set of $K_B.R_2$, in which K_B is the member of R_l . If R_l is a manifold role, that is, $\{K_{B_1}, \dots, K_{B_k}\}$ is the member of R_l , then the principal set of *R* contains the principal set of $K_{B_1}.R_2 \cap \dots \cap K_{B_k}.R_2$.
 - $R \leftarrow R_1 \odot \dots \odot R_k$: The body part consists of Product element, which means the principal *p* is the member of *R* and $p = p_1 \cup \dots \cup p_k$. p_j is the member of R_j .
 - $R \leftarrow R_1 \otimes \dots \otimes R_k$: The body part consists of ExclusiveProduct element, which means the principal *p* is the member of *R* and $p = p_1 \cup \dots \cup p_k$. p_j is the member of R_j , especially for each $i \neq j$, $p_i \cap p_j = \emptyset$.
- (9) *AP*: Access Policy, each statement has the form of $\langle ar, c, vote \rangle$, where *ar* is access rule and similar to *credential*, *c* is group context variable. When a peer requests the role of *ar*'s head part, all policy statements are checked one by one until one of them approve the access. *vote* has one of the following forms:
 - *true*: *vote* is always *true*.
 - *fixed* (*r*, *m*, *f*): A voting is called among members of the *r* role. If *k* votes are received and $f \times k$ are yes, then *vote* is *true*(*m*, $k \in \text{integer}$; $k \geq m$; $f \in [0,1]$).
 - *dynamic* (*r*, f_1 , f_2): This is equivalent to *fixed*(*r*, $m = n \times f_1$, f_2), where the role *r* has *n* members(*m*, $k \in \text{integer}$; $f_1, f_2 \in [0,1]$).
- (10) *RP*: Remove Policy, each statement has the form

$\langle r, c, vote \rangle$, in which r is role, c is context variables. If c and $vote$ are *true*, then a peer can be removed from the role.

3.3. Collaboration policy instance

According to the above access control policy model, the kernel parts of APMGroup policy is depicted in Table 1.

Table 1. Collaborative policy instance

GroupName: <i>APMGroup</i>
C: $day \in \{MON, \dots, SUN\}$
R: {group authority, group member, director, researcher}
PA: group authority: $\langle create, APMGroup, true \rangle$ $\langle modify, CPI, true \rangle$
group member: $\langle join, APMGroup, true \rangle$ $\langle receive, content, true \rangle$ $\langle access, group key, true \rangle$
researcher: $\langle issue, content, true \rangle$
director: $\langle update, sensitive data, day=FRI \rangle$
AP : group authority $\leftarrow K_{BHU.projectleader}, true, true$
group member $\leftarrow K_{BHU.student(since=2000)}, true, vote (group member, 0.4, 0.5)$
researcher $\leftarrow K_{THU.teacher}, true, true$
researcher $\leftarrow K_{CAS.master}, true, true$
director $\leftarrow researcher \otimes researcher, true, true$
RP: researcher, true, vote (group authority, 2, 1)

The APMGroup defines two application roles, namely *director* and *researcher*. For example, the teacher of THU could apply for the researcher role in APMGroup, which is denoted as $researcher \leftarrow K_{THU.teacher}, true, true$. The director role constructed by two different researchers of CROWN group is a manifold role and may update the sensitive data on a given day, such as Friday.

4. Joint authorization

In a distributed environment, peers wish to manage group security by themselves without appealing to a central server or CAs. Joint authorization by multiple peers under voting schemes could satisfy this requirement. Table 2 summarizes the notions used in the rest of this paper.

In the previous example, $group member \leftarrow K_{BHU.student(since=2000)}, true, vote (group member, 0.4, 0.5)$ represents an event that a BHU student registered after year 2000 is requesting to join the group, and a vote is called among peers. If at least 40% votes are received and half of those votes are *yes*, the requester is allowed to join the group. Specifically, the joint authorization protocol has five phases, which are *group initialization*, *searching group*

advertisement, *authorization request*, *voting*, and *CGC issuance*.

Table 2. Notion summary

GA	group authority
M_i	the i^{th} peer within CROWN group
OC_i	organization credential of M_i
CGC_i	CROWN group credential of M_i
SK_i, PK_i	M_i 's secret and public keys
$S_i(x)$	signature of message x with SK_i
ID_i	The Peer ID is the fingerprint hash of the CGC_i 's public key

1) Group Initialization: The group authority peer initializes the local secure environment by creating a secure CROWN group, and then inserts the secure group advertisement including CPI into the network. The CPI contains the access control policy of CROWN groups and various parameters such as group name, voting type, etc.

2) Searching Group Advertisement: When a new peer wants to join the group, it must obtain the advertisement of its attributive CROWN group first. In this design peers have two ways to get this information. (1) Via some rendezvous points. A rendezvous point could be a special peer which keeps information about the groups, or it could be a public website. (2) To flood a query into the grid system, and get response from other peers.

3) Authorization Request (Step 1 in Fig.2): Having the advertisement message, a new peer may connect with the corresponding authority peer. The new node should provide the related credential OC_{new} , which is obtained offline from its organization. For example, the credential of *YuChu* who is a student of BHU is $K_{BHU.student} \leftarrow YuChu$. Then the M_{new} will generate a CGC_{new} issuance request containing OC_{new} information about the new node and its desired privileges.

$$M_{new} \rightarrow GA: Role_REQ, S_{new}(Role_REQ), OC_{new}$$

4) Voting (Step 2, 3 in Fig.2): Upon receipt of the authorization request, the authority peer first verifies the signature. In a fully distributed CROWN group, the request is either accepted or rejected by the collective set of current members. The authority peer then propagates the request to call a vote. According to the CPI, multiple peers authenticate the attribute of a requester, vote, and reply with a signed message to approve or reject the authorization request.

$$GA \rightarrow M: Role_REQ, S_{new}(Role_REQ), OC_{new}$$

$$GA \leftarrow M: vote_i, CGC_i (vote_i = (RES)^{SK_i} \bmod n_i)$$

Figure 2. Joint authorization under voting (1 authorization request, 2 propagate request, 3 multiple peers vote, 4 credential issuance, 5 new peer join)

5) CGC Issuance (Step 4, 5 in Fig.2): Once sufficient votes are collected, GA verifies all the votes, and determine whether to accept the new node as a member or not. If the requester is qualified, the authority peer will issue the CGC_{new} to it and update the related CROWN group information. Having the CGC_{new} , the new node can join the secure CROWN group.

$$M_{new} \leftarrow GA: CGC_{new}$$

5. Secure cooperation

Large sensitive data shared by multiple peers will be generated during the whole life cycle of collaboration and should not be modified by any single user. Such resources are usually stored in a stand alone peer. In the APMGroup scenario, the director constructed by two different researcher of the APMGroup is a manifold role and may update the sensitive data on Friday. The policy is described as:

director: <update, sensitive data, day=FRI>
director ← *researcher* ⊗ *researcher*, true, true

The secure cooperation progress of multiple peers is as follow.

1) Cooperation Request: When a researcher in the APMGroup wants to update the sensitive resource R, it propagates the cooperation request to all researchers.

$$M_{initiator} \rightarrow M_i: updateR_REQ, \\ S_{initiator}(updateR_REQ), CGC_{initiator}$$

2) Cooperation Response: Once the request is received, the peers verify the signature, and then the request is either accepted or rejected by the set of current researchers.

$$M_{initiator} \leftarrow M_i: res_i, CGC_i$$

$$(res_i = (updateR_RES)^{SK_i} \bmod n)$$

3) Cooperation Implementation: Once enough signed responses are collected, the initiator sends all the signed messages to a stand alone peer M_{server} where sensitive data is stored in. The stand alone peer will approve or reject the request according to the CPI.

$$M_{initiator} \rightarrow M_{server}: updateR_REQ, \\ S_{initiator}(updateR_REQ), CGC_{initiator}, \\ \{(res_1, \dots, res_i), (CGC_1, \dots, CGC_i)\}$$

6. Performance evaluation

6.1. CROWN group architecture

We implement the distributed access control in CROWN systems using Java programming language. The cooperation facility among peers is provided by CROWN grid, a fully decentralized grid middleware infrastructure, whose functionalities include file sharing, auctions, distributed computing, and event subscription and publishing.

The system has a three-layer architecture. The bottom layer is the core of CROWN groups. It has building blocks to enable key mechanisms for cooperation applications, including transport, the creation of CROWN groups, and associated security primitives.

Our scheme is implemented in the middle layer, which is built on top of the communication middleware, including security and network services. Examples of network services include searching and indexing, peer discovery, protocol translation, etc. There are three dominant security requirements in CROWN grid systems: confidentiality, integrity, and availability. They actually include functionalities such as authentication, access control, encryption, secure communication, non-repudiation, membership, and group key management. The sequence diagram for CROWN group service design is illustrated in Fig.3.

Generally, to use a service, a peer must present its credential to service providers. We use XML to represent both access control policy and credentials. The credential has the form (issuer ID, owner ID, attribute, issue time, expiration date, peer signature). The credential is signed by the issuer. Delegation credentials should have a short enough lifetime that they are revoked automatically when they expire. Li [16] presents a type system about credentials storage together with algorithms, which ensures chains can be found among distributed credentials storage. The techniques developed in [16] can be used in the CROWN group.

The other component that deserves some words in the middle layer is the policy engine, which acts as the central agent, checking for conformance to the security

Figure 3. Sequence diagram for CROWN group service

policy. All interpretation of policy occurs within the policy engine, so that multiple policy approaches can be integrated. The enforcement of authentication and access control is performed by the policy engine. Each service is protected by policy. For example, a membership service consults the policy engine when a new node attempts to join the CROWN group. The policy stating the requirements to gain access to the group (i.e., the group contexts and credentials) are stated in the authentication and access control rules, such as $group\ member \leftarrow K_{BHU.student}(since=2000), true, vote(group\ member, 0.4, 0.5)$.

Access control policy infrastructures are evolving with the complex environments that they support. Context is used by policy to allow environmental factors to influence how and when policy is enforced. In a grid environment, a context handler mainly collects such information including message context, resource context, and group context, etc.

6.2. Implementation environment

Our proposed access control model is employed by

Figure 4. Success ratio in centralized authority

Figure 5. Success ratio in two group authority

Figure 6. Success ratio in three group authority

the CROWN grid. To evaluate its performance, we conduct comprehensive experiments on a 32-node cluster with a high-speed LAN in the CROWN grid. All nodes are Intel Nocona Xeon 2.8GHz, 2G RAM Linux machines. The cluster is connected to the Internet through a 100M bps connection. No other tasks are running on each node except the necessary CROWN middleware. As the setup phase, the group authority creates and publishes the group authorization service advertisement. Nodes discover the authorization advertisement messages from the rendezvous peer or by flooding. All group access control protocol messages are encapsulated within standard CROWN group message types. The group authority will refresh the authorization service advertisement after delegating the authority attribute to another group member.

6.3. Experiment results and analysis

6.3.1. Efficiency for concurrent requests

In our first set of experiments, we implement secure CROWN group consisting of 40 members. The access control policy of each node is configured by using XML. We let several new nodes send joint requests concurrently. To better evaluate the CROWN group, we also implement a Gothic [13] like approach, and compare its performance with our design in the prototype system.

In Figures 4, 5 and 6, we plot the accumulated joining ratio against time. There are four curves in the three figures with each represent a different request size, ranging from 10 to 40. Figure 4 plots a Gothic like approach, while Fig. 5 and Fig. 6 adopt our design, having two and three group authorities respectively.

We contrast their performances with 40 new nodes in Fig. 7. We can see that after 20 seconds, the success joining ratios vary from 12.5% for a centralized scheme, 50% for two authorities, to 75% for three authorities. Figures 8 and 9 show the average joining time

Figure 7. Success ratio for 40 new nodes

Figure 8. Average join time in delegation and centralization

Figure 9. Success ratio for joining in 20 seconds

Figure 10. Success ratio for dynamic group members

Figure 11. Average join time for dynamic group members

Figure 12. Success ratio for dynamic threshold

and the success joining ratio of the CROWN group and the Gothic like scheme by varying the number of concurrent requesters. Clearly, the efficiency of peers concurrently joining is significantly improved by our approach.

6.3.2. Impact of group size and vote threshold

We vary the number of group members from 10 to 100 and plot the accumulated joining ratio versus time in Fig. 10. With a larger group, it takes longer time for new nodes to join. We further show the average joining cost for 20 new nodes when the group size is ranging from 20 to 100 in Fig. 11. From both figures we can see our delegation authorization approach provides an effective mechanism to improve the efficiency of a CROWN group, and control the overhead of the group authorities.

We then explore the impact of the vote threshold to the system performance. Figure 12 plots the joining ratio in 20 seconds when 20 new nodes would like to join. In Fig. 12, x-axis represents the vote threshold, and y-axis represents the joining ratio. We can see a CROWN group is more flexible and efficient. For example, as illustrated in Fig. 12, if we want to have half of the new nodes join the group within 20 seconds, the vote threshold must be less than 24% for a Gothic

like approach, while it could be 50% for our CROWN group.

7. Conclusion and future work

Our key project, CROWN, holds the main goal to empower integration of resources and co-operation of researchers. To meet security requirements of CROWN, we propose a fine-grained and attribute-based access control framework which extends the role-based trust management language. Our approach employs a distributed delegation authorization mechanism to avoid single point of failure. In order to simplify authorization and access control in collaborations, decisions are made based on authenticated attributes of the peers, which improve flexibility of the system. Furthermore, large sensitive data generated during the collaborations are shared and protected.

This design has been successfully implemented in the CROWN grid environment. We evaluate our proposed approach by comprehensive experiments. We believe that wide deployment of the CROWN group will benefit many grid systems.

Future work leads into investigating trust management in our CROWN group to reduce the

spread of malicious content so as to assure reliability as well as availability of services.

8. References

- [1]The Gnutella protocol specification 0.6, <http://rfc-gnutella.sourceforge.net>.
- [2]Sun Microsystems Project JXTA v2.0: Java Programmer's Guide, <http://www.jxta.org/>.
- [3]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, Hammamet, Tunisia, 2001.
- [4]J. E. Altman, Sun Microsystems, Project JXTA : PKI Security for JXTA Overlay Networks, <http://www.jxta.org/docs/pki-security-for-jxta.pdf>.
- [5]M. Blaze, J. Feigenbaum, J. Ioannidis, and A. D. Keromytis, "The KeyNote trust management system version2, IETF RFC 2704," 1999.
- [6]Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making Gnutella-like P2P Systems Scalable," in Proceedings of ACM SIGCOMM, 2003.
- [7]I. Foster and A. Iamnitchi, "On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing," in Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS), 2003.
- [8]L. Gong, Project JXTA: A Technology Overview, <http://www.jxta.org/project/www/docs/TechOverview.pdf>.
- [9]L. Guo, S. Chen, S. Ren, X. Chen, and S. Jiang, "PROP: a Scalable and Reliable P2P Assisted Proxy Streaming System," in Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS), 2004.
- [10]H. Harney, A. Colegrove, and P. McDaniel, "Principles of policy in secure groups," in Proceedings of Network and Distributed Systems Security, San Diego, CA, 2001.
- [11]W. Hong, M. Lim, E. Kim, J. Lee, and H. Park, "GAIS: Grid Advanced Information Service based on P2P Mechanism," in Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing (HPDC-13), 2004.
- [12]J. Huai, Y. Liu, X. Li, and C. Hu, "Early Experiences with CROWN Grid," *Technical Report, School of Computer Science, Beihang University*, 2005.
- [13]P. Judge and M. Ammar, "Gothic: A group access control architecture for secure multicast and anycast," in Proceedings of INFOCOM, 2002.
- [14]Y. Kim, D. Mazzocchi, and G. Tsudik, "Admission control in peer groups," in Proceedings of IEEE International Symposium on Network Computing and Applications (NCA), 2003.
- [15]D. Li, X. Lu, and J. Wu, "Fission E: A Scalable Constant Degree and Low Congestion DHT Scheme Based on Kautz Graph," in Proceedings of IEEE INFOCOM, 2005.
- [16]N. Li, W. H. Winsborough, and J. C. Mitchell, "Distributed Credential Chain Discovery in Trust Management," in Proceedings of the 8th ACM Conference on Computer and Communications Security, 2001.
- [17]N. Li, J. C. Mitchell, and W. H. Winsborough, "Design of a role-based trust management framework," in Proceedings of the 2002 IEEE Symposium on Security and Privacy, 2002.
- [18]N. Li and J. C. Mitchell, "Datalog with constraints: A foundation for trust management languages," in Proceedings of the 15th International Symposium on Practical Aspects of Declarative Languages, 2003.
- [19]N. Li and C. Nita-Rotaru, "A Framework for Role-Based Access Control in Group Communication Systems," CERIAS Tech Report 2003.
- [20]Y. Liu, X. Liu, L. Xiao, L. M. Ni, and X. Zhang, "Location-Aware Topology Matching in Unstructured P2P Systems," in Proceedings of IEEE INFOCOM, 2004.
- [21]Y. Liu, Z. Zhuang, L. Xiao, and L. M. Ni, "A Distributed Approach to Solving Overlay Mismatch Problem," in Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS), 2004.
- [22]P. McDaniel and A. Prakash, "Methods and Limitations of Security Policy Reconciliation," in Proceedings of the IEEE Symposium on Security and Privacy, Oakland, California, USA, 2002.
- [23]P. McDaniel, "On Context in Authorization Policy," in Proceedings of the 8th ACM Symposium on Access Control Models and Technologies (SACMAT), Como, Italy, 2003.
- [24]B. C. Neuman and T. Ts'o, "Kerberos: An authentication service for computer networks," in *IEEE Communications Magazine*, 1994, pp. 33-38.
- [25]N. Saxena, G. Tsudik, and J. H. Yi, "Admission Control in Peer-to-Peer: Design and Performance Evaluation," in Proceedings of ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN), Virginia USA, 2003.
- [26]V. Sunderam, J. Pascoe, and R. Loader, "Towards a Framework for Collaborative Peer Groups," in Proceedings of the Third IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), 2003.
- [27]M. R. Thompson, A. Essiari, and S. Mudumbai, "Certificate-based authorization policy in a PKI environment," *ACM Transactions on Information and System Security*, 2003.
- [28]V. D. Gligor, H. Khurana, R. Koleva, V. Bharadwaj, and J. Baras, "On the Negotiation of Access Control Policies," in Proceedings of the 9th Security Protocols Workshop, Cambridge, UK, 2001.
- [29]L. Xiao, Z. Xu, and X. Zhang, "Low-cost and Reliable Mutual Anonymity Protocols in Peer-to-Peer Networks," *IEEE Transactions on Parallel and Distributed Systems*, 2003.