Trusted P2P computing environments with role-based access control

J.S. Park, G. An and D. Chandra

Abstract: A P2P computing environment can be an ideal platform for resource-sharing services in an organisation if it provides trust mechanisms. Current P2P technologies offer content-sharing services for non-sensitive public domains in the absence of trust mechanisms. The lack of sophisticated trust mechanisms in the current P2P environment has become a serious constraint for broader applications of the technology although it has great potential. Therefore in this work an approach for securing transactions in the P2P environment is introduced, and ways to incorporate an effective and scalable access control mechanism – role-based access control (RBAC) – into current P2P computing environments has been investigated, proposing two different architectures: requesting peer-pull (RPP) and ultrapeer-pull (UPP) architectures. To provide a mobile, session-based authentication and RBAC, especially in the RPP architecture, lightweight peer certificates (LWPCs) are developed. Finally, to prove the feasibility of the proposed ideas, the RPP and UPP RBAC architectures are implemented and their scalability and performance are evaluated.

1 Introduction

Peer-to-peer (P2P) file sharing has been one of the most popular means of sharing resources in a distributed environment. As is widely known, in a P2P environment, there is no concept of a dedicated centralised server to provide clients with requested resources. Instead, every peer or participant in the system acts as both client and as server, depending on the context. Compared with the client-server model, P2P-based resource management services have several advantages. For instance, in P2P environments, users can share heterogeneous resources residing in various platforms and perhaps in different policy environments. A P2P-based resource management model can provide higher resource availability because of the distributed nature of P2P computing. Unlike the client-server model, each peer, as a service provider, can define the resources provided, the service levels and their conditions. Additionally, each peer, as a service requestor, selects one of the available service providers (peers) based on service levels and conditions. That is, there can be multiple peers for the same resources that are proposing different service levels or conditions (e.g. different service fees). Furthermore, P2P-based service provides higher utilisation of Internet service resources (e.g. bandwidth) because of its distributed routing architecture.

However, the lack of sophisticated trust mechanisms in the current P2P environment has become a serious constraint for broader applications of the technology, especially for non-public applications, although it has great potential [1–5] as described earlier. Suppose there is a content service P2P network where multiple service providers are involved. It provides such resources as on-demand video/audio contents and articles, with different service levels, based on requesting peers' credentials. In such a case, basically, we could stipulate which peer has access to what resources under what conditions. Each service provider may have its own access control policy for the resources it provides. Considering the dynamism of a large-scale P2P environment, where peers from different organisations join and leave P2P communities frequently, the increasing complexity hinders content management. Most research efforts in the field of P2P computing have been focused on the characteristics of different P2P systems and on how services are shared among the peers. Less effort, however, has been focused on establishing control between the peers. We cannot help but address that, in a large-scale P2P computing environment, it is important to provide effective and scalable access control mechanisms. Several types of access control can be enforced for this purpose. However, the conventional identity-based access control is severely undermined by its inability to scale with the population of the P2P network. Therefore, in this work we introduce approaches for providing the scalable and efficient access control mechanism, role-based access control (RBAC [6–12]), in two different architectures: requesting peer-pull (RPP) and ultrapeer-pull (UPP).

2 Operation overview

RBAC has rapidly emerged in the 1990s as a technology for managing and enforcing security in large-scale, enterprise-wide systems. The basic notion of RBAC is that permissions are associated with roles, and users are assigned to appropriate roles, thereby acquiring the roles' permissions. Previously, we identified different RBAC architectures and implemented them on the Web with various mechanisms [12–15]. Our approaches proposed in this study support RBAC mechanisms dynamically based on each peer's current roles. Our work is designed to meet the requirements of environments in which enforcing scalable access control is imperative. Although our approaches are applicable in both the traditional client-server model and

doi:10.1049/iet-ifs:20060084

Paper first received 19th May and in revised form 8th December 2006

The authors are with the Laboratory for Applied Information Security Technology (LAIST), Syracuse University, Syracuse, NY 13244-4100, USA

E-mail: jspark@syr.edu

the P2P model, we apply our ideas to the latter case in this study.

In this work, we introduce two new P2P architectures that are extended from the existing two-tier ultrapeer (super-peer) architecture [16–18]: RPP and UPP architectures. Please note that our approach is not for a public, flat-level content-sharing application, such as music-file sharing on the Internet, where security is not a critical requirement.

Instead, we focus on P2P networks in a private, sensitive environment. Although some public P2P services require each peer to register its identity and authenticate the peer in the beginning of their services, we claim that this does not increase the trust level of the services ultimately because there is no accountability in such services. For instance, any peer can create a fake account and use it in such services. This can occur because there is no true link between a peer and its identity (e.g. account) in a public P2P network. On the contrary, in our approach, we provide a strong link between a peer and its real identity via the first-level authentication (described in Section 4).

Our approaches extend the existing two-tier ultrapeer architecture. In this way, we can make regular peers thinner, more potable and more compatible because a regular peer should only know how to communicate with an ultrapeer. The ultrapeer, then, acts on behalf of its regular peers, including resource search and access control. Furthermore, the entire system can be more scalable in this hierarchical architecture than the traditional flat-level architecture. In this study, for simplicity, we describe only two-tier ultrapeer architecture. However, it is always possible to use our ideas with multiple-tier ultrapeer architectures when the numbers of resources and peers are extremely large.

When a regular node wants to access a resource provided by another peer, it should join an ultrapeer's community after a successful authentication process. It then sends the resource request to the ultrapeer to which it is currently connected. If the resource is under the current ultrapeer's control, the ultrapeer makes the access control decision based on the requesting peer's privilege (i.e. the role in our approach described next). Otherwise, the ultrapeer forwards the request to all of its immediate neighbouring ultrapeers on behalf of the requesting peer. Once the request reaches an ultrapeer that is capable of servicing the request, it checks to see if the requesting peer has the required privilege. Based on the privilege check (i.e. role), a response is sent back to the requesting peer. The operation details in the RPP and UPP architectures are described in the following sections.

In our approach, the access control decisions, searching for resources and resource management are handled by ultrapeers. This makes the regular peers' platform thinner. The regular peer will simply request a particular resource. The requesting peer's ultrapeer will do the query for the resource on behalf of the regular peer. If there is a peer that has the requested resource, the providing peer's ultrapeer makes an access control decision to determine if the requestor has the required privileges. There is no direct connection required between the regular nodes. All the connections are routed through the corresponding ultrapeers. Also, there can be more than one ultrapeer in a P2P community to facilitate better scalability and performance.

3 Lightweight peer certificates (LWPCs)

In order to successfully implement RBAC in a sensitive, private P2P network, we need foolproof methods to check the identity of a given peer, transfer information about the peer's role with integrity, and verify the link between the role and the peer.

Typically, secret-key schemes are not suitable for this purpose because they are not scalable. One approach for providing scalable security service is the use of X.509 digital certificates [19–21]. Additionally, in order to make an access control decision, we need protected attribute information – in our case, the role of the peer. This could be achieved by an attribute certificate [22] issued by an attribute authority in the implementing organisation. The format of an attribute certificate is still X.509-based, except that it has a link to the corresponding X.509 of its subject instead of containing actual public-key information. In this case, the overhead of managing the two distinct certificates (i.e. attribute certificate and identity certificate) is very high, especially in a dynamic environment such as P2P computing. The process of establishing identity and making access control decisions for the peer must be carried out separately, which involves two sets of information exchange. Typically, in P2P computing environments, peers join and leave the peer groups (i.e. community) dynamically, and their access privileges (i.e. roles) can be temporary (e.g. just for one session). In contrast to the highly dynamic P2P environment, an X.509 certificate usually has a long-lived lifetime and requires maintaining certificate revocation lists (CRLs), which increases the overhead of the system.

In order to overcome the challenges faced in using an X.509-based certificate to transfer the peer's role information with effective maintenance, we introduce a lightweight peer certificate (LWPC). This certificate could be used for both the authentication and authorisation of peers in private, sensitive P2P environments (depicted in Fig. 1). Please note that we need a more effective mechanism for supporting dynamic privilege changes in a P2P environment in a scalable manner. As the name suggests, and as we will demonstrate, an LWPC is much lighter and more flexible than an X.509 certificate, thus, will prove more effective for authentication and authorisation in a P2P environment.

The lightness of an LWPC is the result of including only information sufficient to achieve peer authentication and to check the peer's role information. As shown in Fig. 1, in a typical private, sensitive P2P network, an LWPC is comprised of a serial number, peer's identity, peer's authentication information, peer's role information and the validity period of the LWPC. Additional fields can be added to contain application-specific information. Finally, all those fields are signed by the LWPC issuer such as the application role server (ARS). The peer identity is a unique entity that is assigned to the genuine peer who intends to be part of the P2P network. The peer's authentication information can be temporary hashed or encrypted passwords. Using a hash of the password is simpler, because it does not

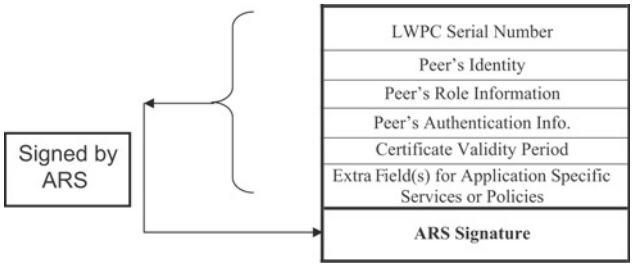

Fig. 1 *Lightweight peer certificate (LWPC) signed by application role server (ARS)*

require additional key management. Alternatively, the ARS can add an encrypted password instead of the hash. Although this can be cryptographically more secure than using the hash, it requires additional maintenance such as sharing the key between the ARS and ultrapeers, which will need to decrypt the encrypted password for peer authentication later. In this case, generally, the public-key-based approach is not an effective solution because the ARS does not know which ultrapeers the LWPC owner (i.e. a regular peer) will connect to later, when the ARS is encrypting the peer's password. In other words, the ARS does not know which ultrapeer's public-key should be used to encrypt the peer's password. The certification authority for an LWPC can be the application's ARS, which maintains the user-role assignment (URA) for the application. An application may run across multiple organisations. It is important to know that an LWPC still uses the public-key cryptography to issue and verify it, which requires only the public-key pair of the ARS. However, unlike the conventional P2P approaches, it does not require each peer's public-key pair. In other words, an LWPC does not need to include the owner's public-key information. Instead, it includes the peer's temporary password information, hashed or encrypted.

The lightness of an LWPC comes at the cost of reduced cryptographic strength as compared with the original X.509, especially the use of hashed passwords instead of public-key cryptography. However, this compromise can be made knowing that an LWPC is valid only for a short period. This contrasts with the nature of an X.509 certificate, which is usually issued with a long lifetime. During this period, the peer's identity would typically remain the same, but the peer's attribute information (i.e. role in our case) could change from time to time in different applications or based on a changed status of the same application. The long-lived, strict certificate does not fit into a typical P2P environment, which should support frequent changes in users and their privileges based on current contexts. An LWPC is issued with a shorter lifetime (i.e. for one session) in order to accommodate the possibility of a peer's changing role from session to session. If a new X.509-like certificate were to be issued for every new session, the cost of managing such a system would be tremendously high. Furthermore, an LWPC is more portable than an X.509 certificate. The only information required to be authenticated using an LWPC is the corresponding password that the user should be able to provide wherever he or she goes. In contrast, to use an X.509, the user needs to carry the corresponding private key stored in a portable machine or device, because a key is not easily memorised by a user. Furthermore, an LWPC does not require maintaining a CRL as does an X.509. Therefore an LWPC has the ability to adapt to the changing roles of a peer, providing greater cost-effectiveness and more scalable maintenance as compared with an X.509 certificate.

4 RBAC architectures in a P2P environment

The following sections discuss the steps involved in a P2P interaction based on our proposed architectures. A resource requesting peer (Alice), who is connected to Ultrapeer1, contacts providing peer (Bob), who is connected to Ultrapeer4, in a sensitive P2P application. We introduce two different architectures – RPP and UPP – for transferring the resource requesting peer's role(s) in a P2P environment. We assume that Alice initially has no idea who has the resources she is looking for (e.g. Resource X). In both RPP and UPP architectures, the initial authentication takes place at the point of joining the network. We call it first-level authentication in the operation. In order for a peer to perform either a resource search or a resource access request, it must be part of the network after successful first-level authentication. To further verify the identity, an ultrapeer needs to check whether the peer is the owner of the LWPC, which contains the owner's role information. We call this as second-level authentication in the operation. Details of the operational procedures, including authentication mechanisms in the different architectures, are described in the following sections.

In a large distributed P2P enterprise, where multiple organisations are involved, each organisation's role structure can be mapped to the enterprise role structure. We could define global roles and assign them to the participating peers. However, we do not recommend the approach in a large distributed P2P environment, where we cannot simply expect a single administrator for the entire enterprise who can directly maintain the mappings between peers and global roles, especially when multiple organisations are involved. Furthermore, those mappings may change frequently in the P2P environment. More details about our previous work on role mappings are described in [14, 15].

4.1 RPP RBAC architecture

Fig. 2 depicts the operational procedures in the RPP architecture. The major procedures are represented in rounded rectangles in the figure. Initially, the resource-requesting peer (Alice) connects to the ARS with her PEER_ID. The ARS then performs first-level authentication via a strong, long-lived authentication mechanism such as an X.509 certificate [19–21], group membership [23], Kerberos [24, 25] or a permanent password. As we are considering a sensitive P2P application with accountability, we assume that each peer (user) possesses at least a genuine account in its organisation with permanent, long-lived authentication information. Different peers may belong to different organisations that may use various authentication schemes. In a system that spans multiple organisations, the ARS will access the authentication servers of the participant organisations to perform this first-level peer authentication. If the authentication by ARS is successful, Alice is required to provide her temporary password (PASSWD). The ARS will include the hash of this password or an encrypted one in the LWPC for Alice (discussed in Section 3). Although using an encrypted password would be more secure than using a hashed password, it requires additional overhead such as key management between the ARS and ultrapeers. For simplicity, we use a hashed password in our description. This temporary password will be valid only for a short period (e.g. for a log-in session). The ARS then uses the PEER_ID to retrieve Alice's role information (ROLE_INFO) from the URA database. The server then issues an LWPC for Alice that holds her PEER_ID, hashed (or encrypted) temporary PASSWD, validity period, ROLE_INFO and other application-specific information depicted in Fig. 1. The newly issued LWPC is transferred to Alice from the ARS, which also maintains a copy of the LWPC. The ARS will overwrite any previously issued LWPC for the same peer to ensure that the LWPC contains the current information for this session.

While her LWPC is valid, Alice connects to an ultrapeer (e.g. Ultrapeer1 in Fig. 2), and presents her LWPC. Ultrapeer1 then performs second-level authentication, requesting Alice to provide the temporary password she

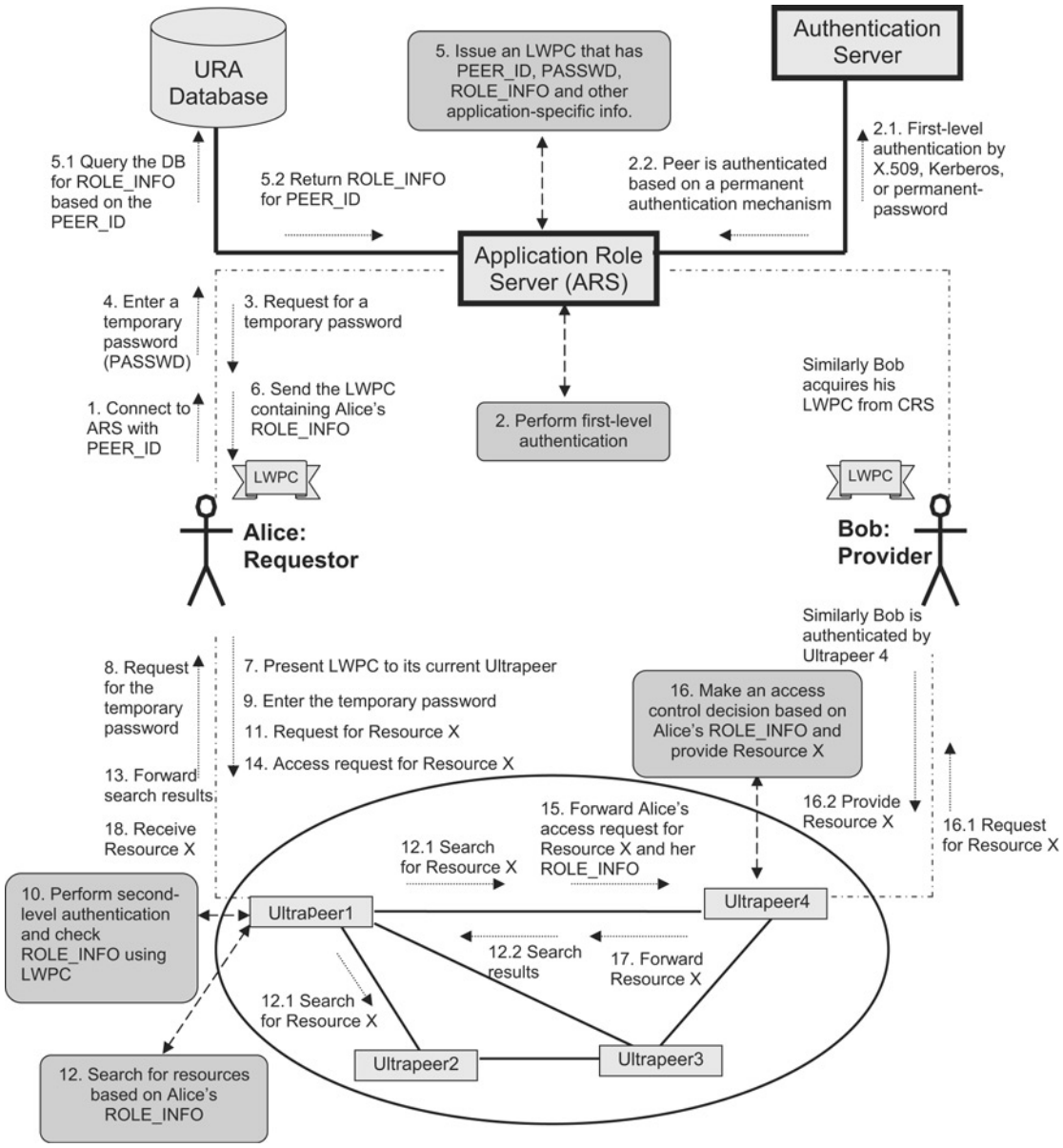

Fig. 2 *Requesting peer-pull (RPP) role-based access control architecture in a peer-to-peer environment*

entered (Step 4 in Fig. 2) when the ARS issued her LWPC. Ultrapeer1 authenticates Alice by comparing the hash of the password she presents against the one included in her LWPC. If they match, the authentication is successful and Ultrapeer1 trusts the information contained in the LWPC, including her role information (ROLE_INFO). If the password in the LWPC was encrypted by the ARS, Ultrapeer1 decrypts it using the corresponding key. In the RPP architecture, peers never use their permanent, long-lived passwords for their second-level authentication with ultrapeers. Consequently, Alice requests for services (e.g. search for Resource X in Fig. 2) via her current ultrapeer, Ultrapeer1, which provides services to Alice based on her roles defined in the LWPC. Ultrapeer1 acts on behalf of Alice, searching for resources in the P2P environment and receiving search queries from other ultrapeers who maintain the resource availability tables (RAT) for their leaf-nodes. For instance, in Fig. 2, let us say Alice's role (e.g. Manager) is authorised to search for resources in the peers under Ultrapeer1, 2 and 4, but not under Ultrapeer3. As a result, the search results show the list of peers who have the resource that Alice seeks (i.e. Resource X) and optional description about the providers such as policies, conditions, reputation, cost, quality of service and so on. Alice considers the resource providers' quality of service and selects one of those resource providers. When Alice requests access to Resource X from Bob, Alice's current ultrapeer, Ultrapeer1, forwards her access request as well as her ROLE_INFO to Ultrapeer4. Now, Ultrapeer4 makes an access control decision based on Alice's ROLE_INFO and its access policy including permission role assignment (PRA). If she is allowed to access, Ultrapeer4 fetches Resource X from Bob and forwards it to Alice through Ultrapeer1. Depending on the policy and application, an ultrapeer may have its own PRA or share the same PRA with others. The former case is more flexible and autonomous, but the PRAs as a group should be consistent in granting or denying access. For instance, typically, the same role, Manager, should not be permitted access to Resource X under Ultrapeer4 if other ultrapeers would deny access. However, there are some exceptional cases that allow for such a conflict. For instance, if those ultrapeers are competing resource providers in a commercial application, allowing access to Resource X could attract more customers.

Alternatively, once Alice received the search results through her current ultrapeer, Ultrapeer1, she can directly access the resource providing ultrapeer. In our example,

Alice can directly request Resource X from Ultrapeer4. However, in this case Alice should present her LWPC to ultrapeer4, which performs another second-level authentication using the LWPC. If the authentication is successful, Ultrapeer4 provides Resource X to Alice based on her ROLE_INFO included in the LWPC. In this way, we can distribute the network traffic and decrease the overhead of the requestor's ultrapeer, but it is less convenient to the requestor because the requestor must present its LWPC to each resource-providing ultrapeer.

Similarly, Ultrapeer4 acts on behalf of Bob, receiving all search queries and requests from other peers through their ultrapeers and responding with hits for all Bob's available resources as well as the resources of other peers under it. In such a scenario, the resource-providing side needs to verify the authenticity of the requesting peer. The providing peer must also verify that the requester has the required privileges to access the requested resource. Finally, Ultrapeer4 provides the resources that Alice is looking for based on her roles defined in the LWPC. All communications between Alice and Bob are relayed via their ultrapeers. For this session, Alice will never directly communicate with any regular peer. A regular peer will always communicate through either her current ultrapeer or resource-providing ultrapeer, or both. The same holds true for Bob in our example. In this manner, ultrapeers act as proxies for their leaf-nodes and can provide more effective and scalable access control in such a dynamic computing environment. Furthermore, we can dramatically decrease the burden of regular peers.

4.2 UPP RBAC architecture

In the UPP architecture depicted in Fig. 3, the first-level authentication is done in the same manner as in the RPP architecture; however, it is performed by the peer's ultrapeer – not by the ARS. In fact, there is no direct communication between regular peers and the ARS in this architecture. Therefore when a regular peer is joining a community, the ultrapeer authenticates its leaf-node by using a strong, long-lived authentication mechanism. Furthermore, the ultrapeer pulls the peer's role information from the ARS when it is needed. Unlike the RPP architecture, the peer's role information is transferred from the ARS to the role-verifying ultrapeer. For the UPP architecture, although the use of LWPCs is technically possible, it is not effective in most cases because we assume that in this architecture, in the course of verifying a peer's role information, the ultrapeer can retrieve the requesting peer's role(s) directly from the ARS. Therefore we

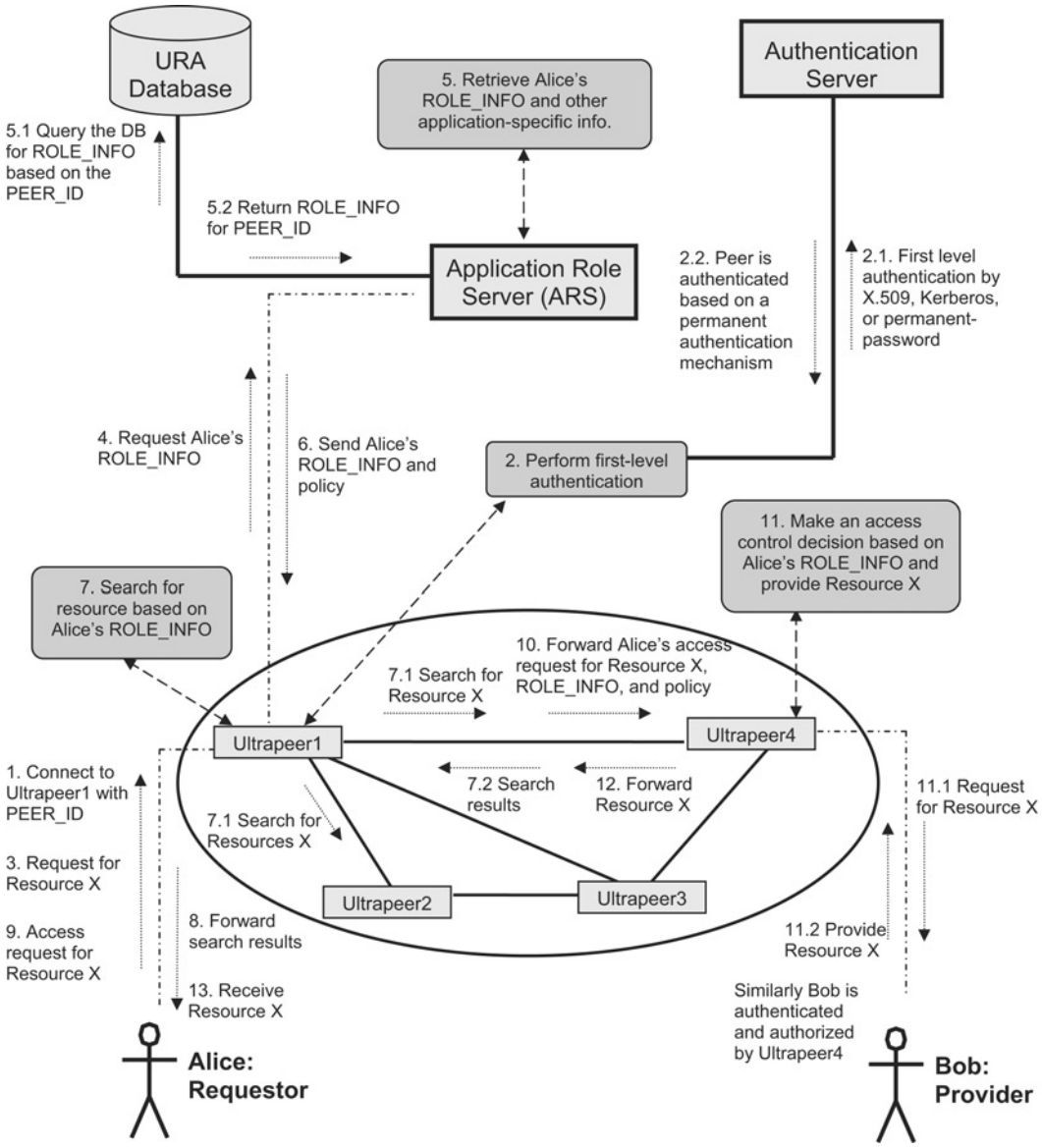

Fig. 3 *Ultrapeer-pull (UPP) role-based access control architecture in a peer-to-peer environment*

recommend the use of LWPCs only for the RPP architecture. This makes the UPP architecture more convenient and transparent to regular peers because they do not need to provide their temporary passwords for second-level authentication. However, this requires ultrapeers to perform more services on behalf of regular peers. It is not simple for the UPP architecture to implement mobile policies and constraints in a distributed manner. For instance, in the RPP architecture, an LWPC can contain such policies and constraints (e.g. validity period), as well as role information. When an ultrapeer received an LWPC from a regular peer, it can make an access control decision based on ROLE_INFO and application-specific policy described in the LWPC. However, in the UPP architecture, in order to consider the corresponding application-specific policy, an ultrapeer should look up the policy server whenever it needs to make an access control decision.

Let us take the same service that we considered in the UPP architecture. Initially, the resource-requesting peer (Alice) connects to its ultrapeer, Ultrapeer1, with her PEER_ID. As we discussed earlier, different peers may belong to different organisations, where each may use a different authentication scheme. In such heterogeneous environments, ultrapeers must have access to each participating organisation's authentication servers in order to authenticate the peer and allow it to join the network. (This procedure does not perform role verification yet.) In Fig. 3, Ultrapeer1 performs first-level authentication via a strong, long-lived authentication mechanism based on Alice's X.509 certificate, group membership, Kerberos or a permanent password housed in an authentication server accessible to the backbone of ultrapeers. If the first-level authentication is successful, Alice requests for services (e.g. search for Resource X in Fig. 3) through her current ultrapeer, Ultrapeer1.

Ultrapeer1 then retrieves Alice's ROLE_INFO from the URA database through the ARS using Alice's unique PEER_ID, and provides services to Alice based on her roles. For instance, in Fig. 3, suppose Alice's role (e.g. Manager) is authorised to search the resources in the peers under Ultrapeers1, 2, and 4, but not under Ultrapeer3. As a result, the search results show the list of peers who have the resource that Alice seeks (i.e. Resource X) and optional description about the providers such as policies, conditions, reputation, cost, quality of service and so on. Alice considers the resource providers' quality of service, and selects one of those resource providers. When Alice requests access to Resource X from Bob, Alice's current ultrapeer, Ultrapeer1, forwards her access request as well as ROLE_INFO to Ultrapeer 4, which also looks up ARS for additional policies and constraints. Ultrapeer4 then makes an access control decision based on the ROLE_INFO, policy, constraints and its PRA. As we discussed earlier, an ultrapeer may have its own PRA or share the same PRA with others. If Alice's role is authorised to access Resource X, Ultrapeer4 fetches Resource X from Bob and forwards it to Alice through Ultrapeer1.

Alternatively, just like we did in the RPP architecture, once Alice received the search results through her current ultrapeer, Ultrapeer1, she can directly access the resource-providing ultrapeer. In our example, Alice can directly request Resource X from Ultrapeer4. However, in this case Ultrapeer4 should perform another first-level authentication to check the requesting peer's identity. Furthermore, it should retrieve the requesting peer's ROLE_INFO and application-specific policy from ARS. If the authentication is successful, Ultrapeer4 provides Resource X to Alice based on her roles. In this way, we can distribute the network traffic and decrease the overhead of the requestor's ultrapeer, but it is less convenient to the requestor because the requestor must follow a first-level authentication in each resource-providing ultrapeer.

5 Experiment

5.1 Implementation summary

We proved the feasibility of our ideas by implementing the RPP and UPP architectures. For our implementation, we selected Windows XP as the operating platform and the JXTA (Juxtapose) framework [26] to provide an existing P2P system with our proposed RBAC services in the two different architectures. We used the JXTA security package to provide the required cryptographic tools, employing the digital signature functions to sign the LWPCs generated by the ARS in the RPP architecture. For the prototype implementation we did not use the standard communication system of JXTA. Instead, we developed our own P2P architecture, which uses sockets from the Java networking package. Developing our own communication subsystem enabled us to add the XML-based communication protocol used for interaction between peers. A customised communication subsystem is also easier to understand and manipulate. To establish a communication protocol between the interacting peers, we use XML-messaging, wherein all messages exchanged between peers are passed in the XML format.

In our approach and its implementation, the URAs are maintained in a centralised location (in the role server), while the PRAs are maintained in a distributed manner (in the ultrapeers). This approach affords several administrative benefits. First, it provides efficient maintenance because all the URA mappings for the participating peers in the enterprise are in the same place. URAs need to be frequently changed in real applications, so centralised URA maintenance provides effective real-time update and synchronisation. Second, our implementation supports easy coordination between URAs, such as an enterprise's inter-community constraints. For instance, if we need to enforce the separation of duties through the peers' roles in different communities, we can easily manipulate URAs when they are located in the same place. Third, our implementation supports better performance in finding a peer's URA information because it is faster and more accurate to search one centralised information site than a number of widely distributed places. In real P2P environments, a URA search across communities is frequently requested. Distributing PRAs to their corresponding community policy servers makes sense, because, unlike URAs, PRAs are usually stable and do not require dealing with community interdependencies. Although supporting separation of duties can be difficult through distributed PRAs, the same effect can be achieved through centralised URAs. Therefore we believe that for P2P computing environments in which a number of heterogeneous communities participate, it is a good strategy to centralise URAs while distributing PRAs to their corresponding peer nodes.

5.2 Performance evaluation

Fig. 4 shows the results of our experimental performance evaluation. For the experiment, we have implemented the RPP and UPP architectures, and the traditional ID-based scheme described in this work. The experimental networks

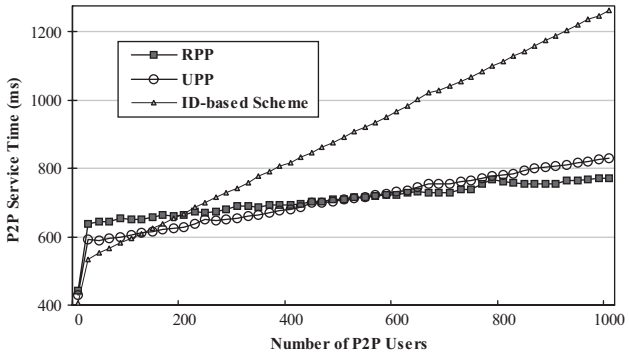

Fig. 4 *Results of experimental performance evaluation*

consists of five routers, 1000 P2P nodes, three ultrapeers, an AS (authentication server), an ARS and a PRAS (permission-role-assignment server). Each link's delay between network nodes is 7 ms. The number of roles and permissions are ten, respectively. In the experiment, the search time for authentication and role information stored in the AS, the ARS and the PRAS is minimal because we have used a binary search. However, the registration for the information to those servers is longer because it requires the construction of index information. In Fig. 4, the *x*-axis indicates the number of all the P2P users registered in AS. The *y*-axis indicates the total P2P service time for a P2P user, from when the peer joins the network until it receives the resource it seeks.

The performance of the ID-based scheme is very sensitive to the number of the registered P2P users, as shown in Fig. 4. This shows that the ID-based scheme suffers from a scalability problem. However, the RPP and the UPP are barely influenced by the scalability problem, as shown in the figure. In our experiment, the performance of the UPP architecture is better than that of the RPP architecture until the number of P2P users is fewer than about 500. On the contrary, the RPP architecture is better then the UPP architecture in situations that involve more than 500 P2P users. This is because the RPP architecture has a shorter response time by using mobile LWPCs in ultrapeers while the UPP architecture requires ultrapeers to connect external servers for peer authentication and role verification (Table 1). Overall, in our experiment the RPP architecture shows the best performance in a large P2P environment of more than 500 peers.

6 Trade-offs

In order to implement RBAC in a P2P environment, the RPP architecture requires more effort than the UPP architecture because each peer needs to communicate with the ARS and using an LWPC is recommended. In the pure UPP architecture, providing mobile policy for authentication and authorisation is more complex than in RPP architecture because such schemes require serious overhead in ultrapeers, especially in the providing side. However, in RPP architecture, mobile policy can be simply provided by using LWPCs. Therefore although UPP is simpler to implement than RPP, it may not be a reasonable approach when we need a session-based security service via mobile policy, which usually is more secure than non-session-based schemes. Overall, the ultrapeers' overhead in RPP architecture is lower because ultrapeers do not need to interact with the ARS, which provides users with more convenience. Furthermore, in RPP architecture, the response time required to obtain a requested resource is quicker as all transactions are between peers and do not involve the ARS. Conversely, in UPP architecture, the ultrapeers (on both the requesting and receiving sides) connect to the ARS at different times once the peer has requested a resource. However, the RPP architecture requires each peer to obtain its LWPC before it connects to ultrapeers.

The UPP architecture provides greater mobility because a peer can directly access any ultrapeer in the application, while a peer in the RPP architecture requires an LWPC for access to an ultrapeer. Finally, the RPP architecture provides higher reusability because a regular peer maintains a copy of its LWPC, which is reusable in multiple ultrapeers as long as the LWPC is valid. However, once an LWPC has been issued and transferred to a peer, the information, such as role defined in the LWPC, cannot be simply or dynamically updated. This can be more effective in UPP architecture because the peer's fresh role information is transferred whenever the peer makes a new connection with an ultrapeer, even during the same session.

Table 1: Trade-offs in RPP and UPP RBAC architectures

Characteristics	RPP architecture	UPP architecture
Implementation simplicity	More complex	Simpler
Mobile policy	More effective	Less effective
Ultrapeer overhead	Lower	Higher
Peer convenience	Lower	Higher
Response time	Shorter	Longer
Reusability	Higher	Lower
Mobility	Lower	Higher
Dynamic update	Less effective	More effective

RABC, role-based access control; RPP, requesting peer-pull; UPP, ultrapeer-pull

7 Related work

Fenkam *et al.* developed an access control system for P2P mobile teamwork environments [27]. The teamwork infrastructure relies on a P2P middleware. A user is required to present his or her authorisation certificate to service providers. An authorisation certificate provides information about access right, user ID, object ID, expiration date and the signature of a special peer who delivers the certificate. However, there is no strong verification mechanism for checking whether an authorisation certificate belongs to the requesting peer, because the certificate does not include any authentication information about the owner. We believe, by using X.509-based certificates as in the traditional P2P authorisation or LWPC as proposed in our work, we can provide and check the link between the certificates and the owners.

Sandhu and Zhang proposed a trusted computing architecture to enforce access control policies in P2P environments based on an abstract layer of trusted hardware [28]. They consider the integrity and trust of platforms and applications that are used by a user to access an object. They also integrate user attributes such as roles into the architecture by using identity and attribute certificates. They apply the approach of Park and Sandhu [29] for binding a role certificate with an identity certificate. The original work identified three different binding mechanisms with respect to monolithic, autonomous and chained signatures. It provided generic architectures of the different binding mechanisms, and was not application-specific. Technically, it is possible to apply those binding mechanisms in different

ways to various applications. In [28], as one possible way, checking the link between an attribute certificate and the owner requires verification of the corresponding identity certificate, which makes the entire P2P environment tightly dependent on PKI. This can be too expensive and restrictive for a dynamic, distributed environment such as the P2P environment. On the contrary, however, the approach proposed in this work can make the role verification process more effective in P2P environments by using LWPCs.

Park and Hwang introduced a controlled P2P computing architecture by extending the concept of Web services [30, 31] through a middleware [15]. Each providing peer makes the access control decision based on the requesting peer's role. They design and develop a middleware platform that works as a broker between peers. The middleware retrieves the requesting peer's role information from the role server and sends it to the requestor. Also, it looks up the policy server and generates policy metadata, which is transferred to the providing peer. Later, the providing peer makes an access control decision based on the policy metadata it received and the role information presented by the requesting peer. According to our approaches described in this work, their work belongs to the RPP architecture, while our approach is more scalable and requires fewer overheads in the regular peers because ultrapeers work on behalf of their regular peers.

Winslett *et al.* introduced the PeerAccess framework in distributed systems [32]. The framework can be used in reasoning about the behaviour of resource owners, their clients and the authorisation service in supercomputing grids. Bertino *et al.* provided cryptographic access control to their P2P framework by using field encryption in XML files [33]. Zhang and Kindberg introduced an authorisation infrastructure in the CoolTown project for nomadic computing [34]. Kim *et al.* developed a secure platform for P2P computing in the Internet [35]. Tan *et al.* identified access control requirements in P2P environments and proposed a trust-based access control framework in such environments by integrating aspects of trust and recommendation models [36]. All these models support secure communications and authorisation in P2P environments. However, as they are identity-based approaches, their management schemes may not be scalable in large, dynamic P2P environments. By inserting the concept of RBAC into our P2P architectures, we make our approaches more scalable.

8 Summary and future work

In this work we have introduced an approach for securing transactions in the P2P environment and we have investigated ways to incorporate an effective and scalable access control mechanism, RBAC, into current P2P computing environments, proposing two different architectures: RPP and UPP architectures. To provide a mobile, session-based authentication and RBAC, especially in the RPP architecture, we develop LWPC. Finally, to prove the feasibility of our proposed ideas, we implement the RPP and UPP RBAC architectures and evaluate their scalability and performance. By considering roles, instead of identities, for access control decisions, we can dramatically increase the scalability in a large distributed P2P environment. However, identification of each peer (i.e. authentication) is still needed in any system in order to provide the security principle of accountability, which is required in all existing non-public systems. Traditional approaches consider identities not only for identification but also access control. This is why we claim that they are not scalable. On the contrary, our approach considers only roles for access control once the identification is successful.

Our future research will also explore how this framework can be expanded for adoption into large, complex organisations through the use of a composite RBAC approach [37]. A composite RBAC's separation of organisational and system-level role structures allows for organisational roles to be reused across various target systems within the same organisation (system neutrality), while simultaneously enabling the reuse of target system roles across different organisations (organisation neutrality). We will also look for ways to improve access control flexibility by introducing multiple concurrent roles into our framework so that users may be able to activate a set of non-conflicting roles, as opposed to just one role, each session.

9 References

1 Androutsellis-Theotokis, S., and Spinellis, D.: 'A survey of peer-to-peer content distribution technologies', *ACM Comput. Sur.*, 2004, **36**, (4), pp. 335–371
2 Yu, B., Singh, M.P., and Sycara, K.: 'Developing trust in large-scale peer-to-peer systems'. Proc. IEEE Symp. on Multi-Agent Security and Survivability, August 2004, pp. 1–10
3 Bailes, J.E., and Templeton, G.F.: 'Technical opinion: Managing P2P security', *Commun. ACM*, 2004, **47**, (9), pp. 95–98
4 Good, N.S., and Krekelberg, A.: 'Usability and privacy: a study of Kazaa P2P file-sharing'. Proc. SIGCHI Conf. on Human Factors in Computing Systems, Lauderdale, FL, April 2003, pp. 137–144
5 Schechter, S., Greenstadt, R., and Smith, M.: 'Trusted computing, peer-to-peer distribution, and the economics of pirated entertainment'. Proc. 2nd International Workshop on Economics and Information Security, MD, USA, May 2003
6 Ahn, G.-J., and Sandhu, R.S.: 'Role-based authorization constraints specification', *ACM Trans. Inf. Syst. Secur.*, 2000, **3**, (4), pp. 207–226
7 Ferraiolo, David, F., Sandhu, R., Gavrila, S., Kuhn, D.R., and Chandramouli, R.: 'Proposed NIST standard for role-based access control', *ACM Trans. Inf. Syst. Secur.*, 2001, **4**, (3), pp. 224–274
8 Li, N., Mitchell, J.C., and Winsborough, W.H.: 'Design of a role-based trust management framework'. Proc. IEEE Symp. on Research in Security and Privacy, Oakland, CA, May 2002, pp. 114–130
9 Li, N., and Tripunitara, M.V.: 'Security analysis in role-based access control'. Proc. 9th ACM Symp. on Access Control Models and Technologies, Yorktown Heights, NY, June 2004, pp. 126–135
10 National Institute of Standards and Technology (NIST): 'The economic impact of role-based access control', Planning Report 02-1, March 2002
11 Sandhu, R., Coyne, E.J., Feinstein, H.L., and Youman, C.E.: 'Role based access control models', *Computer*, 1996, **29**, (2), pp. 38–47
12 Park, J.S., Sandhu, R., and Ahn, G.-J.: 'Role-based access control on the web', *ACM Trans. Inf. Syst. Secur.*, 2001, **4**, (1), pp. 37–71
13 Park, J.S., Ahn, G.-J., and Sandhu, R.: 'RBAC on the web using LDAP'. Proc. 15th IFIP WG 11.3 Working Conference on Database and Application Security, Ontario, Canada, July 2001, pp. 19–30
14 Kang, M.H., Park, J.S., and Froscher, J.N.: 'Access control mechanisms for inter-organization workflow'. Proc. 6th ACM Symp. on Access Control Models and Technologies, Chantilly, VA, May 2001, pp. 66–74
15 Park, J.S., and Hwang, J.: 'Role-based access control for collaborative enterprise in peer-to-peer computing environment'. Proc. 8th ACM Symp. on Access Control Models and Technologies, Como, Italy, June 2003, pp. 93–99
16 Nejdl, W., Wolpers, M., Siberski, W., Schmitz, C., Schlosser, M., Brunkhorst, I., and Löser, A.: 'Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks'. Proc. 12th Int. Conf. on World Wide Web, Budapest, Hungary, May 2003, pp. 536–543
17 Pyun, Y.J., and Reeves, D.S.: 'Constructing a balanced, (log(N)/loglog(N))-diameter super-peer topology for scalable P2P systems'. Proc. 4th Int. Conf. on Peer-to-Peer Computing, Zurich, Switzerland, August 2004
18 Singla, A., and Rohrs, C.: 'Ultrapeers–another step towards scalability', Lime Wire LLC, Version 1.0, November 2000
19 draft-ietf-pkix-ipki-part1-11: 'Internet X.509 public-key infrastructure certificate and CRL profile', 1998
20 ISO/IEC9594-8:1993: 'ITU-T recommendation X.509. Information technology–open systems interconnection – the directory: authentication framework', 1993

21 ITU-T recommendation X.509: 'Information technology–open systems interconnection – the directory: authentication framework', 1997
22 RFC3281: 'An internet attribute certificate profile for authorization', 2002
23 Saxena, N., Tsudik, G., and Yi, J.H.: 'Admission control in peer-to-peer: design and performance evaluation'. Proc. 1st ACM Workshop on Security of Ad Hoc and Sensor Networks, Fairfax, VA, 2003, pp. 104–113
24 Neuman, B.C.: 'Kerberos: an authentication service for computer networks', *IEEE Commun. Mag.*, 1994, **32**, (9), pp. 33–38
25 Steiner, J., Neuman, C., and Schiller, J.: 'Kerberos: an authentication service for open network systems'. Proc. Winter USENIX Conf., Berkeley, CA, 1988, pp. 191–202
26 'Project JXTA, http://www.jxta.org/', accessed 2006
27 Fenkam, P., Dustdar, S., Kirda, E., Reif, G., and Gall, H.: 'Towards an access control system for mobile peer-to-peer collaborative environments'. Proc. 11th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, Pittsburgh, PA, June 2002, pp. 95–102
28 Sandhu, R., and Zhang, X.: 'Peer-to-peer access control architecture using trusted computing technology'. Proc. 10th ACM Symp. on Access Control Models and Technologies, Stockholm, Sweden, June 2005, pp. 147–158
29 Park, J.S., and Sandhu, R.: 'Binding identities and attributes using digitally signed certificates'. Proc. 16th IEEE Annual Computer Security Applications Conference, New Orleans, LA, Dec 2000, pp. 120–127
30 Shirky, C.: 'Web services and context horizons', *Computer*, 2002, **35**, (9), pp. 98–100
31 W3C Working Group: 'Web services architecture', http://www.w3.org/TR/ws-arch/, accessed February 2004
32 Winslett, M., Zhang, C.C., and Bonatti, P.A.: 'Access control: PeerAccess: a logic for distributed authorization'. Proc. 12th ACM Conf. on Computer and Communications Security, November 2005
33 Bertino, E., Ferrari, E., and Squicciarini, A.C.: 'Trust-X: a peer-to-peer framework for trust establishment', *IEEE Trans. Knowl. Data Eng.*, 2004, **16**, (7), pp. 827–842
34 Zhang, K., and Kindberg, T.: 'An authorization infrastructure for nomadic computing'. Proc. Seventh ACM Symp. on Access Control Models and Technologies, Monterey, CA, June 2002, pp. 107–113
35 Kim, W., Graupner, S., and Sahai, A.: 'A secure platform for peer-to-peer computing in the internet'. Proc. 35th Annual Hawaii Int. Conf. on System Sciences, January 2002
36 Tran, H., Hitchens, M., Varadharajan, V., and Watters, P.A.: 'A trust based access control framework for P2P file-sharing systems'. Proc. 38th Annual Hawaii Int. Conf. on System Sciences, January 2005
37 Park, J.S., Costello, K.P., and Diosomito, J.A.: 'A composite RBAC approach for large, complex organizations'. 9th ACM Symp. on Access Control Models and Technologies, Yorktown Heights, New York, June 2004, pp. 163–172