PKI-Based Security for Peer-to-Peer Information Sharing

Anonymous
Anonymous
anonymous@anonymous.org

Abstract

The free flow of information is the feature that has made peer-to-peer information sharing applications popular. However, this very feature holds back the acceptance of these applications by the corporate and scientific communities. In these communities it is important to provide confidentiality and integrity of communication and to enforce access control to shared resources. We present a number of security mechanisms that can be used to satisfy these security requirements. Our solutions are based on established and proven security techniques and we utilize existing technologies when possible. As a proof of concept, we have developed an information sharing system, called scishare, which integrates a number of these security mechanisms to provide a secure environment for information sharing. This system will allow a broader set of user communities to benefit from peer-to-peer information sharing.

1. Introduction

The free flow of information is the feature that has made peer-to-peer information sharing applications popular. However, this very feature holds back the acceptance of these applications by the corporate and scientific communities. In these communities it is important to provide access control mechanisms so that the information owner can clearly and securely define just what is shared and with whom. For example, a group of collaborating scientists would like to share the initial findings of their research within their group, but do not want these findings available to the general audience until they have had a chance to verify them.

Ideally, an information sharing application should allow end users to:

1. Search for information.
2. Make their information available.
3. Download information.

In order to secure this type of information sharing application we need to provide confidentiality and integrity of all communication, and to enforce access control of resources i.e. communication channels and shared information. Existing systems have some combination of these features and security properties, but none provide all of them.

Searching in peer-to-peer systems often involves broadcasting a query to all of the known peers. We refer to this type of communication as group communication. A number of systems provide confidentiality and integrity of group communication [9] [16]. This is typically achieved by building an overlay that encompasses the group and then using TLS to secure the communication between pairs of peers. This solution is inefficient as it requires decryption and encryption of each message that a peer forwards and is exasperated by the fact that a peer often forwards multiple copies of a received message. We use the Secure Group Layer (SGL) [1] to provide confidentiality and integrity of group communication in a more efficient manner. SGL accomplishes this by using a shared group key for securing the messages.

In existing peer-to-peer information sharing systems, authorization and authentication of peers is performed in a purely centralized manner [9] or autonomously by each peer [16]. The first of these is not fault-tolerant and the second may lead to inconsistent enforcement. Our solution distributes the authorization and authentication enforcement while maintaining consistency among the peers. We use SGL to provide authentication enforcement and build extensions for SGL to provide fine-grained authorization. In addition, we allow for trusted peers to invite or escort other peers into the group.

Existing peer-to-peer information sharing systems do not differentiate between resources. That is, an authenticated peer has access to all the shared information of every other authenticated peer. We provide mechanisms that allow each end user to autonomously specify the authentication and authorization requirements for each information item. We leverage previous work on fine-grained authorization for distributed environments by using the Akenti authorization system [14]. We modify Akenti in order to distribute the enforcement mechanism to each peer and develop a group-based authorization model that hides many of the complexities of the Akenti system to the users.

We have developed an information sharing system that satisfies the desired feature set for information sharing and utilizes the security mechanisms described. This system, called *scishare*, is not only a proof of concept, but also a fully functioning information sharing system. Our solutions are based on established and proven security techniques. We have made considerable effort to not reinvent the wheel and have utilized a number of existing PKI-based technologies.

In Section 2, we present an overview of the system and its architecture. Then, in Section 3 we discuss our solutions to securing communication in a peer-to-peer environment. In Section 4, we describe our solutions for fine-grained, distributed, resource access control. Section 5 describes related work in the peer-to-peer information-sharing arena. Finally, we conclude and present possible future work in Section 6.

2. System design

In this section we first provide an overview of our secure information sharing system, including the security model. We then present the architecture and technologies used in our system.

2.1 System overview

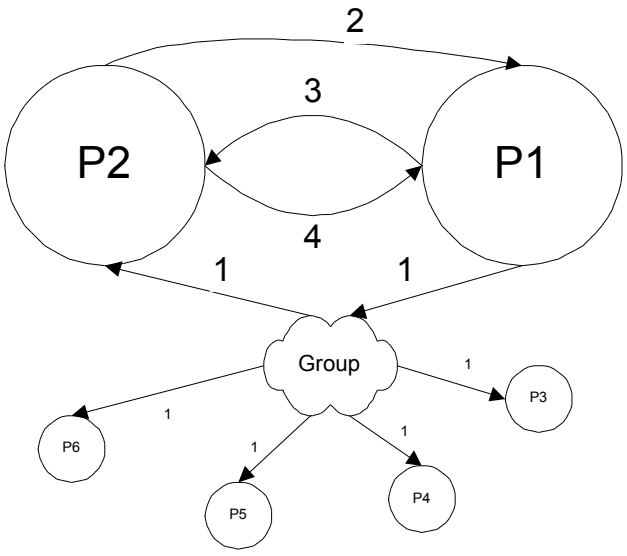

Figure 1. Peer interactions

Figure 1 shows the interactions between peers. A peer, P1, searches for information by sending a *query* message to the group (step 1). Peers P3 through P6 don't have content that matches the query, so they do not respond. Peer P2 has content that matches the query, so it sends a *query response* message (step 2). P1, can then request to download content from P2 by sending a *transfer request* message (step 3). P2 answers this request with the content or an error message (step 4). In this respect, our system is very similar to existing peer-to-peer file sharing systems, such as Gnutella [8]. The difference stems from our security model. In the remainder of this section we discuss the system actions in more detail, the security requirements they impose, and our security model.

Communication primitives. We consider a set of peers, P, that form a peer group, G. The peers communicate by exchanging messages. A peer, P_i, can send a message to any other peer, P_j, or to the peer group, G. Communication between two peers, P_i and P_j, is direct, i.e. we do not consider the scenario where a third peer, P_k, acts as proxy, forwarding messages for the communication. Messages sent to the peer group G are intended for every peer in G.

Searching for information. An important feature of an information sharing system is the ability to search for the desired information. In our system model, a peer P_i searches for information by sending a *query message* to a peer group G, where P_i is a member of G. Each peer, P_j in G, that receives the query message, checks whether any items match the query and responds directly to P_i with a *query response message* that contains the metadata associated with the items that matched the query.

The security requirements for the search are:

1. The query message is confidential to G.
2. The query response message is confidential between a P_j and P_i.
3. Each query response only contains the metadata for items for which P_i is authorized access.
4. P_i is able to verify its trust relationship with each P_j from which it receives a query response.

Transferring information between peers. A peer P_i may request a transfer of information from a peer P_j, by sending a *transfer request message* to P_j. P_j, upon receiving this message checks whether it has the information item associated with the request. If P_j has the item then P_j transfers the requested information to P_i. If the information is transferred to P_i, then P_i becomes the owner of that copy of the information.

The security requirements for information transfer are:

1. The transfer request message and the transfer of the information are confidential between P_i and P_j.

2. P_i and Pj are able to identify each other and thus determine the level of their trust relationship.
3. The information is transferred from P_j to P_i only if P_i is authorized to access that information.

Managing locally shared information. Each end user designates a set of information items that its peer host can share with the other peers in the group. Each information item has a pair of authorization requirements. One is for the metadata associated with the information item and the other is for the actual item. Users should be able to specify these requirements through a simple user interface.

The specified requirements for an information item are autonomous. For example, if peer P_i allows P_j access to an information item I, it indirectly relinquishes the right to fully control access to I. We do not prohibit P_j, once it obtains a copy of I, from specifying its own authorization requirements for I that are independent of those specified by P_i.

Security model. Our security model adapts a number of security solutions built around the X.509 public-key infrastructure (PKI) to provide confidentiality, integrity, authentication, and authorization. We have developed mechanisms that allow X.509 entities to build secure collaborations. These mechanisms do not prevent outsiders from participating and provide a secure environment where members and outsiders can meet and interact. Outsiders can earn the right to become full partners based on their contribution and/or behavior.

We also accommodate peers which do not have certificates. We call these *pseudo users*. *Pseudo users* are provided with automatically generated X.509 certificates and have access to public resources. However, these peers cannot be granted higher levels of trust in the system.

We only allow peers that have obtained X.509 certificates from a verifiable certification authority (CA) to become trusted peers in our system. We made this decision because: (1) our target communities (research labs, universities, and corporations) have the resources to deploy certification infrastructures, (2) in this model peers are shielded from being tricked into giving away too much power to untraceable and potentially dangerous entities, and (3) our trust verification mechanism ensures that certificates presented by peers have not been revoked, a very important component of the security guarantees we provide. In summary, our mechanisms enable all peers to easily determine the trustworthiness of other peers and information, and also provide them with options to disable interactions with outsiders.

2.2 Architecture

Figure 2 shows the architecture at a peer. *SGL* is used to send and receive queries within a group and *TLS* [14] is used for all other communication. The *authentication managers* are used to provide coarse-grained access control to the communication channels. The *authorization manager* and the *Akenti engine* are then used to provide fine-grained access control to the communication channels and shared information.

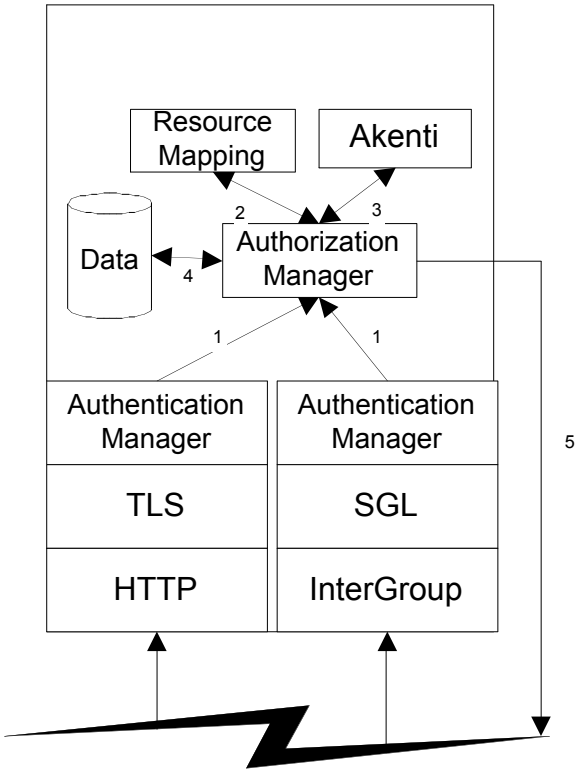

Figure 2. Peer architecture

When a query message or transfer request message is received by the peer it is first processed by the appropriate authentication manager. Queries are processed by the authentication manager associated with SGL and transfer requests are processed by the one associated with TLS.

If the message is approved by the authentication manager, it is passed on to the authorization manager (step 1). If the message is a transfer request, the authorization manager then determines the policy associated with the content requested from the *resource mapping* (step 2). The resource mapping component maintains a mapping between a resource and its associated policy. The policy and identity of the requestor are then handed to the Akenti engine, which

returns the actions that the requestor is allowed to perform on the resource (step 3). If the requestor has read access, the data is retrieved from the data store (step 4) and sent to the requestor (step 5).

The remainder of this section describes SGL and Akenti. The authentication managers are described in Section 3 and the authorization manager is further described in Section 4.

SGL. SGL [1][7] provides the security services required by applications utilizing reliable group communication in wide-area environments. SGL establishes secure multicast channels among application components. An SGL secure multicast communication channel is established by first exchanging a session key among the legitimate application components. This key is then used to achieve multicast message confidentiality and/or multicast data integrity within the group.

SGL provides the application with a security context within which messages multicast over the wire can be cryptographically protected. The essential building block for setting up a secure multicast context is a distributed key exchange protocol that allows the participants to exchange a session key as equals and, therefore, treats them as peers. The first step in solving this problem was to design an algorithm that allows a set of participants to agree on a session key [4]. We refer to this kind of group key genesis as the initial group Diffie-Hellman key exchange. Alone, the group Diffie-Hellman key exchange is of relatively little practical use. A mechanism to enforce restrictions on who can participate in the key exchange and, therefore, the multicast group is needed. SGL integrates the Diffie-Hellman key exchange and authentication mechanisms into a security layer.

In our system, we layer SGL on top of the InterGroup protocols [3] to provide secure and reliable group communication. InterGroup provides membership services, reliable message delivery, and ordered message delivery for many-to-many communication. The InterGroup protocols are designed to scale to wide-area environments such as the Internet. SGL secures InterGroup in much the same way that the TLS secures TCP.

Akenti. Akenti [14] provides scalable authorization services in highly distributed network environments. Stakeholders, i.e. resource owners are able, in a flexible and secure way, to remotely and independently define and deploy distributed resource policies used for fine-grained access control.

The Akenti engine allows for multiple and distributed resource owners and supports very powerful ways to express the constraints on the use of resources. Furthermore, it uses minimal centralized knowledge to securely collect distributed policy in the form of certificates during the evaluation phase. This dynamic policy collection enables Akenti to scale well to large and cross-domain populations both in terms of policy management and in terms of users joining and leaving organizations. After evaluating the policy for a resource, Akenti can issue a signed authorization token to the holder of a X.509 public key, authorizing him/her to perform actions with respect to the resource.

3. Securing communication

We use multicast to send and receive queries and unicast for all other communication. We use SGL to secure the multicast and TLS to secure the unicast. Both support authentication services and trust decisions based on X.509 certificates.

Our security model enables secure interactions among peers that do not necessarily know each other and encourages a gradual build-up of trust. Peers with certificates from verifiable CAs have the potential of becoming full partners in the system, while peers with no certificates are provided pseudo certificates by the application software which can be used to participate in the SGL and TLS session key generation. However, these pseudo certificates can never be used to establish the holder as a fully trusted partner. In our system, *pseudo users* have access to world readable files and can express policies to protect their own files. The remainder of this section presents the approach we have taken to allow for gradually building trust in peers [2] and to promote sharing in the overall system.

3.1 Multicast communication

SGL allows a group of peers to communicate securely. The peers participate in a handshake protocol that results in the derivation of a group shared secret key to encrypt and protect messages.

The SGL specification [7] allows extensions to its authentication mechanism. Pluggable authentication managers are used during the initial phase of the SGL handshake protocol, called the HELLO phase. Every peer has an authentication manager that presents the peer's credentials to SGL, collects the credentials of the other peers attempting to join the group, and finally determines the set of eligible peers that may proceed with the handshake.

We define a secure multicast channel as the 4-tuple, <IP address, port, locator, public key>. The channel's administrator, who may not be a user of the information sharing application, creates a channel configuration policy (CCP). The administrator then signs the CCP with a 'well secured' private key. Each peer interested in joining a group obtains the channel 4-tuple using offline methods,

uses the locator to retrieve the CCP, and finally verifies the CCP using the public key. It is important to note that each peer performs these steps before attempting to join the channel, as the CCP is part of the authorization data sent in the HELLO phase.

The CCP contains a serial number, which allows each peer to detect the case where more than one CCP is received from the group. This scenario, which can occur when some of the peers have not downloaded the latest CCP, is resolved by simply picking the CCP with the highest serial number. Consequently, all peers use the same security and operational parameters. In particular, when the X.509 authentication method is used, they all use the trusted CAs listed in the CCP to determine the trustworthiness of other peers. Normally, untrusted peers are not part of the next phase of the SGL handshake, and thus cannot join the group.

In order to allow a gradual build-up of trust our solution uses two SGL channels, each with a separate CCP. The first channel is secure (encrypted) and only allows trusted peers to know the group key (referred to as the trusted channel), while the other is secure but allows anyone with a X.509 or pseudo certificate to join (referred to as the untrusted channel).

Only peers that satisfy policies defined by the administrator are admitted into the trusted channel. In order to distinguish peers by their privileges, this channel uses Akenti capabilities, which contain the X.509 certificate of the peer, rather than the X.509 certificate alone in order to verify their right to join the group. Section 4.1 describes this authentication method in detail.

Every peer is admitted into the untrusted channel. This channel's CCP uses the X.509 authentication method but with a minor modification. At the SGL level, the authentication managers accept any peer that presents a valid X.509 chain and proof of possession of the corresponding private key. However, at the application level, after the handshake is complete, each peer performs the trust check based on the CAs it trusts. Each non-*pseudo user* for which this check fails is added to a list of untrusted peers. Peers have access to their list of untrusted peers and may add these peers to their trust world. Queries sent by untrusted peers get marked as such by the application allowing the peer to not respond to such queries.

3.2 Unicast communication

Peers may play the traditional role of a client in some cases and the traditional role of a server in others. In our system each peer plays the role of a client when sending a query response or transfer request message, and the role of a server when providing content. TLS, at a minimum, requires that the server side present a certificate chain. We use TLS in Mutual Authentication mode where the client authentication is optional. In other words the server side of a connection will always present a certificate (a pseudo one if necessary) and the client will only present one if it has a real certificate. The use of pseudo certificates does not in any way compromise the security of the sever-client connections. The pseudo private keys are securely generated and stored. In fact, users are not even aware that these certificates exist.

As we have done for the SGL untrusted channel, we defer the trust check and only do authentication at the TLS level. This way, untrusted peers are able to complete the TLS handshake. The trust check is performed at the application level immediately after the completion of the TLS handshake protocol. If the trust test fails at the application level, we flag the metadata or the content as untrusted. In our system, untrusted peers only have access to world readable files. Trusted peers, on the other hand, may have access to other files depending on the policy described in Section 4.2.

We also allow the peers to elect to only communicate with trusted peers. In that case, the trust check is performed immediately at the TLS level. If the trust check fails, the handshake fails and the connection is aborted. Peers electing this option prevent untrusted peers from even opening a communication channel.

Upon establishing a secure unicast channel with an untrusted entity, the peer is prompted to add this entity to their trust domain. If the peer declines, that entity is added to the list of untrusted peers. The peer can later add these untrusted entities to their trust domain.

4. Fine-grained access control

In this section we present fine-grained access control mechanisms that allow trusted peers to have different privileges. We have identified two resources in the system. These are the multicast communication channel and shared information (content and its metadata).

The sharing of resources within rather large and dynamic peer communities, that may span a number of trust domains, must be highly controlled. Each end user should be able to define in a clear and a secure manner just what is shared and with whom. We have already seen in the previous section how SGL and TLS can be used to perform coarse-grained access control to provide equal access to authenticated peers.

We use the Akenti authorization system as a basis for our work on fine-grained access control since it scales well to very large and dynamic populations. The Akenti engine allows for multiple and distributed resource owners and supports very powerful ways to express the constraints on the use of resources. Furthermore, it uses minimal centralized knowledge to securely collect distributed policy in the form of certificates during the evaluation phase. This dynamic policy collection enables

Akenti to scale well to large and cross-domain populations not only in terms of policy management, but also in terms of peers joining or leaving (perhaps forcibly) organizations.

In this section we first introduce a design for fine-grained access control to multicast communication. Then, we present the fully integrated group-based authorization model we have developed to secure access to shared content and its metadata.

4.1 Multicast channel

In Section 3.1, we showed how the CCP is used to make sure that all peers have a consistent way of configuring the SGL channel at runtime as well as how X.509 certificates may be used for authentication. Here we make minor modifications to the authentication method and use Akenti capabilities to provide peers different levels of access to the trusted channel.

The owners of the channel, which are not necessarily peers, generate and deploy the authorization policy. If the system is set up correctly, only minimal policy elements are kept at a central site. The policy elements that change frequently, such as resource usage constraints, are deployed at sites conveniently accessible to the owners. The Akenti engine uses this minimal central knowledge to collect the dynamic policy elements at runtime. Once the policy elements are collected, the engine evaluates the privileges of each peer and produces capability certificates for trusted eligible peers.

Capability certificates signed by the Akenti engine contain the name associated with a peer (typically the end user), the peer's public key, the name of the channel, and a list of rights or actions that can be awarded to the named peer. The actions may have additional, application-specific conditions attached to them such as the time period during which the action is valid. The channel's authentication managers present the CCP and capability certificates to SGL on behalf of their peers. They verify the integrity of all received capabilities by using the Akenti engine's public key, conveniently contained in the CCP, and verify the identity of all other peers by using public keys contained in the capability certificates.

The possible actions granted to peers are *join*, *invite*, *escort*, *attend*, and *reject*. The *join* action allows a peer to join the SGL group as a full-fledged member, while the *attend* action allows a peer to join the SGL group as a guest. A peer must have a *join* or *attend* action listed in its capability in order to access the channel.

A peer who has been granted the *invite* action, called the inviting peer, may allow other peers to join the SGL group as guests. The inviting peer provides the potential guest with a signed invitation in the form of a capability certificate listing *attend* as one of the actions along with its own capability. The guest then presents this chain of two (or more if invitation chaining is allowed) certificates during the SGL handshake. Each peer is then able to verify the validity of the invitation(s) without having to contact the inviting peer.

A peer who has been granted the *escort* action, called the escorting peer, may also allow other peers to join the SGL group as guests. However, in this case, the escorting peer must also participate in the SGL handshake for the guest to be admitted into the group. The escorting and guest peers follow the same steps as in the case of an invitation. The only change is that each peer needs to also make sure that the escorting peer(s) are present during the handshake.

A peer who has been granted the *reject* action may remove another peer from the SGL channel by forcing a re-key operation and presenting a 'negative' capability certificate of the peer to be removed. This 'negative' capability certificate names the peer and does not list any action.

In summary, the channel owners specify which peers can join the trusted channel. Some of these peers may have the capability to invite untrusted peers to the trusted channel, based on their experience. This flexibility is essential when forming secure and spontaneous collaborations.

4.2 Information

We hide many of the complexities of the Akenti policy language by providing a group-based authorization model layered between the user and Akenti. Currently**,** a policy is simply a named container that holds a set of groups and a set of rejected peers. The Akenti engine asserts that a peer satisfies a policy if the following conditions are satisfied:

1. The peer is not listed in the rejected peer list.
2. The peer belongs to *at least* one of the listed groups or the policy does not list any groups.

A group is a named set of peers, which is managed by one or more authorities that assert which peers belong to the group. Peers can create their own groups or reference groups created by other peers or third party entities. A peer can make a group available for reference by storing that group's membership certificate, in public directories.

Peers can reference groups in the policies they create by simply supplying the owner of the group and the location of the public directories containing the group information. The Akenti engine asserts that a peer belongs to a referenced group *if and only if* the engine locates, in one of the listed public directories, the peer's valid group membership certificate. Consequently, peers can be added and removed from groups without any communication among the peers.

Finally, when a peer wants to access a resource we simply retrieve the relevant policy from a one-to-one resource-policy mapping managed by each peer and evaluate the peer's privileges based on that policy. In our system, resources are hierarchical and the peer must satisfy the first policy found in the hierarchy.

5. Related work

A number of peer-to-peer information sharing systems have been developed in the last few years (e.g. [8], [11], [12]). The focus of these systems has been on usability and scalability, rather than security. Our system focuses on all three of these aspects, particularly security.

The majority of security research in peer-to-peer has focused on providing anonymity to the user and data (e.g. [5], [6]), and automating the construction of trust relationships in systems utilizing untrusted identities (e.g. [10], [13]). Our system allows users to participate without having to prove their identity, but does nothing to enhance the anonymity of the users or the data. We also leave the construction of trust relationships to the users, providing them with historical data about their interaction with other users, trusted or untrusted.

Waste [16] is a secure file-sharing system that provides security of information within a small trusted group of peers. It secures all communication at the link-level using TLS and builds a PKI web of trust between the trusted peers. Waste assumes that all of the trusted peers are equal. In addition, peers are forced into trust relationships that are commutative and associative. Thus, any peer that is allowed into the system has full access to all the information in the system. Our system provides similar properties, but also allows for building autonomous trust relationships, fine-grained authorization control, and for sharing of public data with peers outside the trusted group.

Groove [9] allows a small group of collaborators to form spontaneous shared spaces in which they exchange information. It essentially implements PKI without the certification framework to build trust among users and it is not clear if any of the implemented security protocols have been proven secure. Groove did put a lot of effort into making the system usable, and has automated a number of protocols to hide the key management issues from the users. It does not support fine-grained access control, but provides support for incremental trust through its built-in invitation protocol.

6. Conclusion and future work

Providing security in peer-to-peer environments is difficult due to the distributed and autonomous nature of peers. The major challenges are providing confidentiality and integrity of group communication in an efficient manner, and distributing the authentication and authorization enforcement such that security is not compromised.

Secure group communication is most efficient if the group members have a common, shared key for securing the communication. This is difficult to achieve in dynamic groups, as a new key must be generated every time a membership change occurs to preserve forward secrecy. SGL provides efficient, distributed, mechanisms for re-keying in this environment. We make use of these mechanisms to provide efficient, secure delivery of queries in our system.

SGL also provides basic authentication mechanisms for the group. However, SGL does not specify the manner in which policies are specified or used. We have developed a group policy and associated mechanisms to guarantee that every peer in a group proceeds with the same policy. This results in every peer in the group providing the same authorization enforcement for that group. In order to provide fine-grained access control for the group, additional mechanisms that leverage of the Akenti authorization system were put in place. We used techniques similar to the ones for authentication to guarantee consistency, allow for peers to invite guests into the group, and allow for peers to invoke re-keying in order to remove peers whose privileges have been revoked from the group.

In our system, peers can build their trust relationships autonomously. An underlying associative trust still exists since peers essentially become owners of data they download. The enforcement of a stricter policy that prevents a peer from modifying the authorization requirements for data it has downloaded is outside the scope of this paper. It is a possible avenue of future work that would need to take into further consideration the area of digital rights management.

By adding authentication and authorization to an application users are often required to obtain credentials from a centralized authority to gain access to even the most basic functionalities. Often, this is unnecessary, as only a subset of the application functionalities needs to be secure. We allow users immediate access to the application by automatically generating pseudo certificates for users that do not have credentials. This allows these *pseudo users* access to the basic functionality of the system without compromising the security model.

We have presented a number of security mechanisms that can be used to provide PKI-based security for peer-to-peer information sharing. Our solutions are based on established and proven security techniques and we utilize existing technologies when possible. As a proof of concept, we have developed an information sharing system, called scishare, which integrates a number of these security mechanisms to provide a secure

environment for information sharing. At the moment, the security mechanisms related to SGL are not yet implemented.

We are currently working on implementing and incorporating the security mechanisms related to SGL into scishare. We are also currently evaluating the security mechanisms described in this paper.

The ideas presented in this paper could be used in other types of peer-to-peer applications. We plan on refining and further simplifying our security mechanisms so that security can be incorporated into peer-to-peer applications in a way that adds value and not inconvenience.

7. References

[1] D.A. Agarwal, O. Chevassut, M.R. Thompson, G. Tsudik, "An Integrated Solution for Secure Group Communication in Wide-Area Networks", *Proceedings of the 6th IEEE Symposium on Computers and Communications*, Hammamet, Tunisia, July 3-5, 2001, pp 22-28.

[2] D. Agarwal, M. Lorch, M. Thompson, and M. Perry, "A New Security Model for Collaborative Environments," Proceedings of the Workshop on Advanced Collaborative Environments, Seattle, WA, June 22, 2003.

[3] K.Berket, D.A. Agarwal, O. Chevassut, "A Practical Approach to the InterGroup Protocols", *Future Generation Computer Systems, Vol. 18 (5)*, Elsevier Science B.V., 2002, pp. 709-719.

[4] O. Chevassut, "Authenticated Group Diffie-Hellman Key Exchange: Theory and Practice", PhD Dissertation, Universite Catholique de Louvain, Belgium, October 2002.

[5] I. Clarke, O. Sandberg, B.Wiley, and T.W. Hong. "Freenet: A distributed anonymous information storage and retrieval system", Lecture Notes in Computer Science, 2001.

[6] R. Dingledine, M. J. Freedman, and D. Molnar. "The free haven project: Distributed anonymous storage service", In Proceedings of the Workshop on Design Issues in Anonymity and Unobservability, July 2000.

[7] G. Egles, O. Chevassut, K. Berket, A. Essiari, D. A. Agarwal. "The Secure Group Communication Protocol version 2", Work in progress.

[8] "Gnutella", http://www.gnutella.com/

[9] "Groove", http://www.groove.net/.

[10] S. Kamvar, M. Schlosser, and H. Garcia-Molina. "Eigenrep: Reputation management in p2p networks", In Twelfth International World Wide Web Conference, 2003.

[11] "Kazaa Media Desktop", http://www.kazaa.com

[12] "Limewire: The Most Sophisticated File-Sharing Application", http://www.limewire.com.

[13] A. Singh and L. Liu "TrustMe: Anonymous Management of Trust Relationships in Decentralized P2P Systems", in Proceedings of The Third International IEEE Conference on Peer-to-Peer Computing, September 2003.

[14] M.Thompson, A. Essiari, S. Mudumbai , "Certificate-based Authorization Policy in a PKI Environment", ACM Transactions on Infomation and System Security (TISSEC), Volume 6, Issue 4, pp: 566-588, November 2003.

[15] "The TLS Protocol Version 1.1", http://www.ietf.org/internet-drafts/draft-ietf-tls-rfc2246-bis-06.txt.

[16] "Waste", http://waste.sourceforge.net/.