Security and Protection Architectures for Large-Scale Content Distribution

A Thesis
Presented to
The Academic Faculty

by

Paul Q. Judge

In Partial Fulfillment
of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Georgia Institute of Technology
November 2002

Security and Protection Architectures for Large-Scale Content Distribution

Approved:

Dr. Mostafa H. Ammar (Advisor)
(College of Computing)

Dr. Andre Dos Santos
(College of Computing)

Dr. Jun Xu
(College of Computing)

Dr. Ellen W. Zegura
(College of Computing)

Dr. Brian N. Levine
(U. Mass. Amherst)

Date Approved 10/31/2002

Acknowledgements

I give thanks to my advisor, Dr. Mostafa Ammar. Mostafa has become a close friend and provided much needed guidance and encouragement. Without his undying support and advice, I surely would not have made it to this point. Along the way, he has helped me learn not only the science of networking, but also the art of research, and most importantly, many lessons of life.

Thanks to Dr. Ellen Zegura who has provided advise and encouragement. I thank Dr. Jim Xu and Dr. Andre Dos Santos who have provided excellent feedback about many of the information security related questions and ideas. I thank Dr. Brian Levine, of U. Mass. Amherst, for agreeing to serve on my thesis committee and taking time from his busy schedule to provide insightful feedback and attend my thesis defense.

Members of the Networking and Telecommunications Group have provided support throughout my time here. I thank Dr. John Limb for his guidance in my early years here. I also thank Dr. Russell Clark for his encouragement and feedback. The students, specifically Lenitra Clay, Jinliang Fan, Richard Liston, Matt Sanders, and Minaxi Gupta have been my friends and collaborators.

Past and current members of the Georgia Tech Information Security Center, specifically Mustaque Ahamad, Blaine Burnham, Michael Covington, Matt Moyer, and Phyllis Schneck have encouraged and supported me.

Contents

List of Tables

List of Figures

Summary

The Internet is increasingly being depended upon to provide large-scale content distribution. An important need is mechanisms to aid in the scalability of distribution services. These mechanisms include multicast, anycast, peer-to-peer, and overlay-based content distribution networks. Multicast communication provides an efficient transport mechanism for one-to-many and many-to-many communication. Anycast aids in service discovery and replication by providing a mechanism for delivering a single request to one of many servers servicing an address. Peer-to-peer systems allow efficient content location and retrieval among large groups of users. Overlay-based content distribution networks provide reliable and robust distribution frameworks.

As content distribution on the Internet becomes more pervasive and the value of the content being distributed increases, the security of this content and its distribution channels has become a main concern of content creators, owners, and providers. There have been significant advances in maturing the distribution mechanisms; however, there are a number of distinct security issues in these technologies. These issues exist because of two reasons: 1) the issues are introduced by the new distribution mechanisms; or 2) the issues also exist in unicast, but the unicast solutions do not apply. To address these problems, our research aims to develop flexible content protection architectures for large-scale content distribution.

Specifically, the contributions of this work are as follows.

- We developed WHIM, a scalable system that allows multicast content to be securely marked with distinct information for distinct receivers. This system introduces two new concepts: 1) generation of a watermark based on the receiver's location in a tree overlaying the network; and 2) incremental insertion

of the watermark in the content as it traverses an overlay network. We propose and evaluate several forms of this architecture and show how it improves scalability while increasing security. We also develop an implementation of the system that allows a multicast video stream to be watermarked by a hierarchy of intermediaries.

- We generalize the problems of secure IGMP and secure anycast server advertisements into a problem of group access control and propose Gothic, a complete architecture for providing group access control. Gothic centers around a novel authorization architecture. This is complemented by a proposal for a group policy management system that allows the group owner to be authenticated before being allowed to specify the group access rights. This system can be applied to other works that involve group policy. We show how Gothic operates in a number of environments including application-layer multicast, source-specific multicast, application-layer anycast and global IP-anycast. We evaluate the security and scalability of the architecture and show that it improves scalability over previous solutions while maintaining or increasing the level of security. We also propose methods of integrating Gothic with the group key management system and content distribution tree. We propose and evaluate a *group-access-control-aware group-key management* technique that leverages the existence of a group access control system to substantially reduce overhead.

- We describe and implement a rights management architecture for decentralized peer-to-peer file sharing systems called *CITADEL*. CITADEL builds a protected file sharing environment over a normal peer-to-peer network using secured content objects and file sharing software enhanced to perform protection operations. A flexible content importation system that is part of CITADEL allows all users to insert new content as well as additional copies of protected

content. We explain how CITADEL provides the flexibility necessary to support common content distribution business models. We also provide results that show the performance of the system relative to other possible approaches. Finally, we describe an implementation of CITADEL that uses the Gnutella network, LimeWire file sharing software and standards-based security tools.

Chapter 1

Introduction

As content distribution on the Internet becomes more pervasive and the value of the content being distributed increases, the security of this data has become a main concern of content providers. Content distribution has become pervasive in part due to the deployment of scalable networked services and architectures. These mechanisms range from network services such as multicast and anycast to networked architectures such as content distribution networks (CDNs) and peer-to-peer systems. The availability of such systems has sparked the interest of end-users and content providers. End-users are attracted to such distribution method's ability to locate and obtain a wide variety of content. Content distributors are drawn to such system's large distribution channel and low cost.

There is significant interest by content creators and owners in the protection of the content that is distributed in these systems. Content protection in any environment is a formidable challenge due to the complex protection goals and range of possible attacks. Additionally, content protection in large-scale content distribution environments introduces a number of issues that do not exist in client-server systems. There has been previous research in understanding and solving security issues in client-server environments. However, the introduction of decentralized systems, group-based systems, and multicast systems brings about a new set of problems. For example, confidentiality, integrity, trust management, and non-repudiation mechanisms in unicast systems do not translate to large-scale content distribution systems.

1.1 Applications

Innovation and the human imagination have repeatedly led to applications that stretch the limits of the computing capabilities of the day. As the demands of applications increase, we have developed sustaining technologies to increase the lifetime of the current technologies. More significantly, it is said that disruptive applications stimulate disruptive technologies that introduce a new value proposition [45]. For example, many of the advances in website technology is considered sustaining technology while the introduction of peer-to-peer content distribution can be considered a disruptive technology.

There are four forces that drive applications: the types of content that users deisre, the ways that users wish to access the content, the bandwidth that is available to users, and the format in which content is available. Content types include movies, music, pay-per-view events, software, business information such as stock quotes, news data such as weather or world events, and real-time conversations.

Another factor is the formats in which the content has been made available. This is influenced heavily by advances in compression and storage. For example, over the years video compression has advanced from MPEG-1 to MPEG-2 [191] to MPEG-4, MPEG-7, MPEG-21 [1], and Windows Media Video [2]. Similarly audio has progressed from WAV to MP3 and Ogg Vorbis [202]. Such changes have significant impact on the applications. For example, the emergence of the MP3 compression format made it possible to transport audio content over the Internet in a amount of time that users found reasonable. This made applications such as Napster possible. Similarly, MPEG-4 and Windows Media Video codecs allow movies and other video content to be compressed to a size that allows such files to be exchanged over the Internet. Different multimedia streaming formats also have infleunced applications.

The increase in available bandwidth is also a driving force. Over the last few years, the amount of bandwidth available to home users rise from 28.8 *kbps* to 33.6

[1] http://mpeg.telecomitalialab.com/

[2] http://www.microsoft.com/windows/windowsmedia/

kbps to 56 *kbps* and then drastically increase to 1.5 *mbps* and up for some users. This has made significant changes in the number of users capable of participating in Internet content distribution and the expectations of those users.

The ubiquity of devices has changed the places and times that users are able to access content. The recent popularity of wireless devices and portable entertainment devices has influenced applications. We have witnessed computing devices move from the office into the home office with PCs and then throughout the home with laptops and computerized entertainment centers. Now with wireless devices, PDAs, and portable entertainment devices, computing and content devices are able to move with the user outside the home.

1.2 Content Distribution Methods

The Internet has succeeded at supporting wide-scale communication in the form of e-mail and web pages. That success has led to technical growth issues, users that expect more, and providers that want to deliver more. Growth issues are a result of the current size and use of the Internet exceeding the original design goals. Therefore, there are some technical barriers such as limited scalability resource bottlenecks and the lack of inherent quality of service (QoS) due to the best-effort model. Higher user expectations have come as a result of the Internet's success. Users expect higher speeds and higher quality. Providers have seen proof of the Internet's potential and now aim to deliver even richer content and offer new services.

These are not new issues. Every since the invent of the Internet, users have pushed it beyond its intended limits and along the way there are technical barriers. Today, we are again asking more of the Internet than it was designed to give. So again we must overcome the technical issues to meet our demands.

Years ago, there was a need for applications that would sent data to large groups. Multicast was developed as an efficient delivery mechanism to ease that strain. Other work resulted in the use of caches in networks to increase efficiency. Recently, we

have built upon our earlier gains and discovered ways to supplement the Internet to provide the resources to meet our demands. This has resulted in mechanisms such as content distribution networks (CDNs). Other efforts have sought to provide efficient content location and retrieval to end-users; this has lead to the popularity of peer-to-peer systems.

1.2.1 Evolution of Content Distribution

Content distribution as it is today is a result of years of evolution. The evolution is sometimes driven by technology, but often driven by applications. Content distribution consists of storage and caching mechanisms and delivery mechanisms. In this section we briefly discuss the evolution that has taken place in these two areas.

1.2.1.1 Evolution of Caching and Storage Infrastructure

It was realized early on that scalability can be increased by allowing popular requests to be handled with some sort of aggregation [24]. Frequently requested objects could be stored in a manner that did not require a full operation to service a user with that object. Similar concepts have been used in other areas such as memory caching.

As the world wide web increased in popularity, Internet Service Providers (ISPs) began to seek ways to lower their bandwidth costs and many used forward proxy caches to cache web pages and other objects that were requested by their customers. This concept advanced to include mechanisms such as co-operative caching that allow multiple caches to exchange information for increased efficiency [207].

Later, content providers used forward proxies when seeking ways to provide better QoS for their customers. However, many of these were maintained by the content provider rather than the ISP and served that provider's most popular content.

As content storage was pushed further to the edge of the network, it finally reached the actual edge: the end-users' computers. This is the underlying concept behind peer-to-peer content distribution. Files are stored at peer computers and

exchanged directly between peers.

As content became more distributed, mechanisms that allowed the efficient location of these resources became important. These mechanisms include server selection, request redirection, content location, and service location [200]. Server selection involves an enduser choosing a particular server from a group of servers based on some metrics. Request redirection allows a host to send a request to a general address and have it redirected to a specific server. This usually involves some form of server selection algorithm. Anycast is an important mechanism in providing server selection, server location, and request redirection. Content location allows a host to efficiently locate content within a distributed system and is an important component of peer-to-peer systems.

1.2.1.2 Evolution of Delivery Model

Multicast was developed to provide scalability by allowing a server to service multiple requests with a single response. Multicast enables efficient large-scale content distribution by providing an efficient transport mechanism for one-to-many and many-to-many communication. Over the years, multicast has been the topic of many research, engineering, and deployment efforts. These efforts have continued to transform multicast into a technology that can be relied upon by many applications. Work has been done in reliability, manageability, scalability, quality-of-service, address allocation, inter-domain multicast, pricing/billing, and ease of deployment [60].

There have been a number of multicast routing protcols proposed including DVMRP [164], MOSPF [143], PIM-DM [57], PIM-SM [64], and CBT [23]. Recently, single source multicast (SSM) [100, 98] has been proposed that simplfies some of the problems faced by normal any-source multicast (ASM), but it is too soon to determine how widespread its use will be.

The industry has come to require a multicast model that is ready for deployment in commercial applications. Thus, to provide the necessary components, the new

paradigm is to add layers to the infrastructure instead of engineering the mechanisms into the existing layers. This has lead to the use of overlay networks and application-layer multicast to provide the missing pieces. Other reasons for interest in overlay networks is that they can offer increased QoS, more robustness, and better manageability [47]. The use of application-layer multicast does not rule out the use of or replace the need for IP multicast. Application-layer multicast allows rapid deployment and a platform to build additional services on top of the multicast model. However, application-layer multicast is not as efficient of a delivery method as IP multicast. In many environments, application-layer multicast and IP multicast can co-exist.

Another advancement in content delivery is the ability to provide enhanced services during content distribution. This allows distributed applications to act on the content as it is being delivered. Such work includes active services, overlay services, and web services. The added functionality may include video smoothing, transcoding, content personalization, advertisement insertion, or content protection.

1.2.2 Multicast

There are a number of available multicast routing protocols that provide the efficient transport mechanisms of multicast by routing packets with one group destination address to multiple recipients. The routing protocols must be aware of group members in the network in order to deliver packets to them. The mechanism provided for doing this is the Internet Group Membership Protocol (IGMP) [33]. A host uses this protocol to notify an edge router that it should deliver packets from a particular multicast group to that host.

1.2.3 Content Distribution Networks

Content distribution networks (CDNs) are network infrastructures that are deployed to deliver content reliably and quickly [200]. The idea is that on top of network

layer connectivity we utilize transport through application-layer resources to provide improved and novel services.

CDNs utilize various mechanisms to overcome common problem areas of the Internet. These problem areas include network congestion and content server bottlenecks. CDNs allow higher levels of service and new levels of scalability. Due to the nature of placing functionality at higher levels, deployment of new services becomes can occur more rapidly.

CDNs involve a number of components. Content routing systems utilize information known by the application to achieve intelligent routing. Content transport can be achieved with application-layer multicast, IP multicast, or unicast. The content distribution system may also include content caching mechanisms. The proxy execution environment allows services to be deployed within the network. Content peering involves the internetworking of seperately managed CDNs. The redirection fabric maps the client request to the CDN to be served. Content network management involves managing the overlay network including the deployment, routing, and monitoring of the network. This allows a number of powerful features such as monitoring of application usage, integration with the pricing model, application-layer quality of service (QoS).

There have been a number of research efforts that proposed application-layer multicast systems [72, 47, 42] and systems for deploying and managing overlay networks [197]. There have been a number of deployed content distribution networks ncluding commercial networks like Inktomi [103] and Akamai [13] and non-commercical networks like Internet2 Distributed Storage Infrastructure (I2-DSI) [104].

1.2.4 Anycast

Anycast allows multiple servers to provide a service at a single address called the *anycast address*. Each *anycast server* lets the routing protocol know that it is

listening to the anycast address. Then when a host wishes to contact a server providing that particular service, it simply sends a request to the anycast address. The routing system knows which servers said that they are providing that service, so it chooses one of those servers and forwards the request to it. Besides the basic IP model of anycast [155], global IP anycast [116] and application-layer anycast [212] have been proposed.

1.2.5 Peer-to-Peer

Peer-to-peer networks are formed as a logical connection of endhosts over the physical network. Peer-to-peer file sharing systems consist of two components, the file location process and the file retrieval process. In most peer-to-peer systems, the file retrieval process is decentralized. That is, files are transferred directly between peers rather than through a client-server model. However, peer-to-peer systems differ in the file location process. There are two main types of peer-to-peer systems. *Centralized* systems such as Napster [3] provide indexing and searching functions at a centrally managed location (or a set of replicated locations), while *decentralized* systems depend on the peers themselves to manage content indexing and search functions in a distributed manner. Among the decentralized systems, there are naive broadcast query systems such as Gnutella [78] and distributed hash table (DHT)-based systems such as CAN, Chord, Pastry and Tapestry [170, 194, 179, 214].

1.3 Security Goals

Information security has been an active area for the last thirty years. The focus has sometimes shifted depending on the computing environment at the time. Early work focused on system security in multi-user and multi-process systems [125, 25]. Then there was work that offered formal definitions of security properties and models [126]. Later work began to examine security issues in distributed systems such

[3] http://www.napster.com

as encryption, authentication, authorization, and trust management [119, 150, 26]. The focus then shifted to protecting data over large networked systems and also protecting the systems. As new types of networked systems are proposed, we still are working on defining models to represent our goals and threats, methods to perform authentication and authorization, mechanisms to protect the data on the system, and ways to protect the system itself.

In this section we briefly discuss some fundamentals of information security such as the different types of security services that may be desired in a particular environment, the attacks that aim to deny these security services, and the classes of mechanisms to defend against these attacks.

1.3.1 Security Services

A security service is a property that may be desired in a particular environment to enhance the security of information. These services counter the different security attacks and use some security mechanism to do so.

A common classification of security services defines six distinct services [117]:

- Confidentiality: Protects data from release of contents.
- Access control: Limits and controls access to objects.
- Availability: Ensures continuous service for intended users.
- Authentication: Ensures the identity of an entity or of the source of data.
- Integrity: Ensures that the data received is the data that was sent.
- Nonrepudiation: Ensures that neither party participating in communciation can deny the occurence of the communciation.

1.3.2 Security Attacks

The most common classification of security attacks defines four general categories of attack [119]:

- Interruption: A component of the system is damaged or otherwise made unavailable for authorized users.
- Interception: An entity gains access to information that it is not authorized to receive.
- Modification: An entity modifies some component of the system such as message contents or modifying the behavior of a program.
- Fabrication: An entity creates unauthentic objects such as messages or data files.

We highlight the types of attack that are dealt with in this thesis and show the particular security service that they attack. Interception and eavesdropping are attacks on confidentiality. Theft-of-service and redistribution are attacks on access control. Interruption or denial-of-service are attacks on availability. Masquerading is an attack on authentication and integrity.

1.3.3 Security Mechanisms

We classify security mechansims by the role they play in defeating attacks:

- Prevention: These mechanisms aim to protect resources in order to deny the ability to perform a security attack or to provide deterrence to decrease the likelihood of an attack.
- Detection: These mechanisms are designed to detect the presence or occurence of an attack so that some measures can be taken.

- Response: The role of these mechanisms is to respond to an attack in a passive or active manner. Passive responses include increasing security by enforcing a more stringent policy. Active responses may include executing a reciprocal attack.

In this thesis, we concentrate on preventive mechanisms. There are two types of preventive mechanisms: protection mechanims and deterrence mechanisms. Protection mechanisms aim to shield systems from exposure. Deterrence mechanisms aim to discourage attackers from acting. Protection mechanisms can be compared with a lock on a door while deterrence mechanisms are similiar to a security camera.

1.4 Thesis Outline

This thesis seeks to define security and content protection issues in large-scale content distribution and propose architectures to solve a range of the issues. We identify the content protection and security goals and methods used to achieve these goals in traditional client-server environments. We discuss the mechanisms used for large-scale content distribution and examine cases in which the traditional mechanisms cannot be utilized in large-scale systems. Additionally, we identify new issues that appear due to the nature of certain large-scale distribution systems. We then propose solutions for a range of these issues. Our set of solutions provides a flexible architecture for content protection for various distribution methods. Specifically, we address the following issues:

- **Theft deterrence:** In unicast environments, fingerprinting is achieved by watermarking the content at the source then distributing it. In a multicast environment, this approach offers no security since all receivers will share a common fingerprint. We explore methods to securely watermark multicast multimedia content while maintaining the scalability advantages of multicast.

- **Distribution tree protection:** In multicast content distribution, users attach to the distribution tree using the Internet Group Membership Protocol (IGMP). In the current model, any host can use IGMP to become a member of any IP multicast group causing eavesdropping, theft-of-service, or resource utilization leading to denial-of-service. In this work, we explore a method to provide access control within IGMP without introducing heavy loads on the network infrastructure such as routers.

- **Rights management:** The popularity of decentralized peer-to-peer file sharing systems has led to environments that require content protection but lack a central authority to enforce the protection. How can content protection be provided and enforced without a central authority? How can the open peer-to-peer sharing experience be maintained in the presence of a content protection system? In this work, we describe a content protection architecture for decentralized peer-to-peer file sharing systems that is designed to answer these questions.

1.5 Organization of Dissertation

The remainder of this thesis is organized as follows. In Chapter 2 we provide an overview of content protection. Chapter 3 summarizes our work in developing WHIM, a scalable fingerprinting for multicast environments. Chapter 4 describes a group access control architecture for secure multicast and anycast called GOTHIC. Chapter 5 motivates the need for content protection in peer-to-peer systems and discusses CITADEL, an architecture for content protection in peer-to-peer systems. In Chapter 6 we describe practical issues involving CITADEL including its support of common business models, simulation results, and an implementation of the system. Finally, chapter 7 summarizes the contributions of the work presented in this dissertation and outlines some future directions for this research.

Chapter 2

Overview of Content Protection

There are numerous works in content protection that cover many environments, distribution methods, content types, and protection goals. The goals within content protection are sometimes referred to as copy protection, conditional access, or digital rights management. Digital rights management is a more generic term that can be used to describe some set of content protection schemes that compose a particular system for value chain participants from content creators to consumers. Three phases of content protection that are common across different distribution methods and content formats are *protected distribution*, *protected storage* and *output protection*. Protected distribution deals with providing conditional access or enforcing an access policy in the distribution model. This essentially controls access to protected objects. Protected storage deals with controlling access to the actual content in a protected object. This essentially controls playback of protected objects. Output protection deals with protecting content after an authorized user is accessing the content. This focuses on restricting access to the content as it is played by the user.

Output protection work includes Digital Transmission Content Protection (DTCP) [11] for protecting content during transmission between devices using IEEE 1394 or Universal Serial Bus (USB), Macrovision Copy Protection [132], High-bandwidth Digital Content Protection (HDCP) [59] for protecting content during transmission to digital displays, and Microsoft's Secure Audio Path [140] for protecting content on PCs during transmission to audio devices such as sound cards.

Protected storage work includes Content Protection for Pre-recorded Media

(CPPM) [8] for protection pre-recorded DVD-Audio, Content Protection for Recordable Media (CPRM) [9] for protecting content stored on recordable media such as DVD-R or flash memory, Content Scrambling System (CSS) [10] for protecting pre-recorded DVD video, and copy-protected CD solutions such as the Cactus Data Shield [141] for protecting pre-recorded CDs from replication or extraction to files such as MP3s.

Protected distribution work takes many forms differing greatly depending on the distribution method. In cable and satellite, conditional access is provided by set-top boxes enforcing subscription and pay-per-view models; see for example the NDS VideoGuard [149]. In CD and DVD sales, conditional access simply means that the person that pays for the content receives the media containing the content. On websites that sell content, protected distribution is performed in the client-server model of purchasing rights and obtaining content. In multicast or group communications, protected distribution is provided by using group keys to access encrypted content [35] and by controlling access to the multicast distribution tree [110].

In this section, we provide an overview of security issues and research in content distribution. We first discuss the causes behind the multicast security issues in Section 2.1.2. We then provide more detailed explanations of security problems in multicast and proposed solutions. We discuss four areas of multicast security research: receiver access control, group key management, source authentication, and multicast fingerprinting. For each area, we further explain the vulnerabilities that it introduces, outline the objectives of solutions, and survey work in the area. In Section 2.6, we briefly highlight other security issues in multicast content distribution including source access control, secure multicast routing, and group policy specification. Section 2.7 explains work in video watermarking. In Section 2.8 we discuss related network security and content protection work.

2.1 Causes of Multicast Security Issues

Multicast enables efficient large-scale content distribution by providing an efficient transport mechanism for one-to-many and many-to-many communication. Over the years, multicast has been the topic of many research, engineering, and deployment efforts. These efforts have continued to transform multicast into a technology that can be relied upon by many applications. Work has been done in reliability, manageability, scalability, quality-of-service, and ease of deployment. As these areas become more mature, there is increased potential for multicast to be used as the underlying distribution mechanism for content distribution applications. Therefore, security in multicast content distribution is an concern. The maturity of multicast security solutions has the potential to enable the use of multicast for confidential and high-value content and help spark the use of multicast by new applications.

There are a number of security issues in multicast content distribution that are directly related to the properties of multicast that make it efficient and attractive. There has been research that provides solutions to many of these security issues. Some of these solutions are ready for deployment, some are nearing maturity, and others are only in the early phases of research. The maturity and deployment of these solutions will help increase the ability of multicast technology to deliver new applications and more content. In the next few sections, we examine these various issues and solutions for providing secure multicast content distribution. [1]

2.1.1 Properties of Multicast

The definition of the *host group model* [43] provides a summary of the key properties of multicast: "a *host group* is a set of network entities sharing a common identifying multicast address, all receiving any data packets addressed to this multicast address by senders (sources) that may or may not be members of the same group and have no knowledge of the groups' membership." This definition highlights the three main

[1] An appreviated version of this taxonomy will appear [112].

properties of multicast:

- *All members receive all packets sent to the address*: Multicast routing delivers all packets sent to the multicast address to all members of the multicast group.
- *Open group membership*: Multicast provides an open group model and allows group membership to be transparent to the source.
- *Open access to send packets to the group*: Any host can send data to the multicast address and it will be delivered to the multicast group without regard for the source of these packets.

We note that we focus here on the host-group native-IP multicast model which allows so called *Any Source Multicast (ASM)* as the most general multicast model available. As such it also represents the most challenging context in which to provide content-distribution security functions. Other multicast models provide more restrictive frameworks that may make it easier to deal with some security aspects. For example in the *Small Group Multicast* [166] model the source needs to know the identity of the multicast group members. Another example is the use of Source-Specific Multicast (SSM) [99] in which groups are associated with a single source and only that source can transmit to the multicast group. Another example is Application-Layer Multicast[49, 106] that utilizes an overlay network to implement multicast functionality including group management and packet forwarding. These more restrictive models, however, while possibly alleviating some aspect of securing multicast distribution, continue to possess other multicast properties (for example, the lack of distinction of received data among the receivers) and therefore, the security techniques surveyed here continue to be relevant.

2.1.2 Security Issues and Solutions

These properties of multicast lead to security issues and vulnerabilities because of two reasons: 1) the issues are multicast-specific; or 2) the issues also exist in

unicast, but the unicast solutions do not apply. Figure 1 shows how each of the three multicast properties leads to vulnerabilities and it shows the areas of research that provide solutions to these issues.

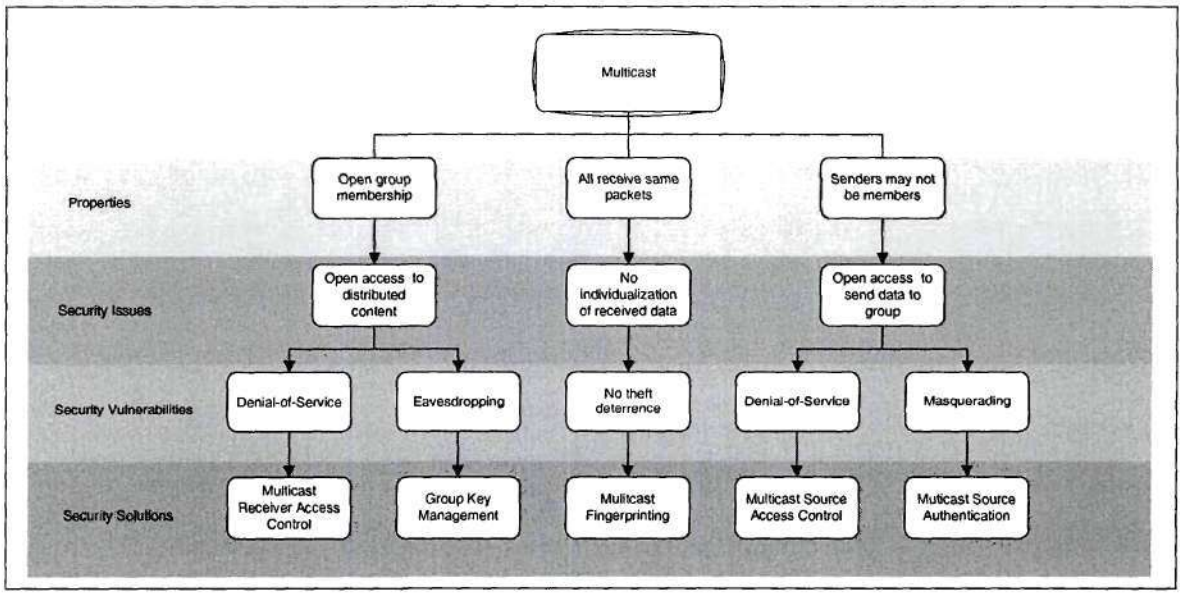


Figure 1: Some Multicast Security Issues and Solutions

- The open group model is beneficial in many environments because it provides a lightweight join operation, the source is not required to maintain state for all group members, and it allows some anonymization for group members. However, this same property of multicast also causes security issues since it is not possible to restrict communication to a set of authorized hosts. In the IP-multicast model, any host can use the Internet Group Membership Protocol (IGMP) [33] to become a member of any IP multicast group—possibly leading to eavesdropping, theft of service, or denial of service. The latter attack can be caused by a malicious host joining a number of multicast groups, thereby utilizing large amounts of bandwidth or router resources. To defend against these threats, two classes of solutions have been proposed: group data encryption with *group key management* and *multicast receiver*

access control.

- The multicast model delivers any traffic sent to the multicast address to the entire group. This means that any host can send data to the multicast group.This leads to two problems. First, group members need to able to verify that messages received are from the intended source. *Multicast source authentication* solutions have been proposed to provide this functionality. Secondly, there should be mechanisms to restrict unauthorized sources from sending data to multicast groups due to the potential for denial-of-service attacks. *Multicast sender access control* solutions are necessary to defend against this threat.

- The fact that all members receive all packets sent to the group is a fundamental feature and benefit of multicast; however, this property also causes some security mechanisms that are used in unicast to not work in multicast environments. One reason for this is that there is no individualization of the received data. Traditionally, this individualization has sometimes been used to provide security. For example, *fingerprinting* [203] is the embedding of receiver identifying information in content to deter unauthorized duplication and propagation. However, fingerprinting techniques used in unicast environments do not work in multicast environments because all users receive the same data. Therefore, *multicast fingerprinting* [46, 32, 198, 109] solutions have been proposed to achieve unique fingerprinting in a multicast environment while maintaining the efficiency of multicast.

As stated above, most of these issues exist across the different multicast models. However, some of the multicast schemes may be immune to some of these issues due to their design. For example, Single-Source Multicast inherently provides some source access control since the group address is based on the source's unicast address [2]. Small Group Multicast provides some receiver access control since the source

[2]This is actually a side effect of reverse path forwarding, not intentional security.

knows the group membership. In Application-Layer Multicast, the receiver access control problem differs since group management may not be based on IGMP.

In addition to the various models, multicast content distribution involves a number of potential environments composed of different Internet Protocol (IP) versions, routing protocols, address allocation schemes, and inter-domain requirements. The security issues that we discuss are relevant to these many flavors of multicast, but may vary slightly across the particular environments.

In the following sections, we discuss these areas of multicast security research: receiver access control, group key management, source authentication, and multicast fingerprinting. For each area, we further explain the vulnerabilities that it introduces, outline the objectives of solutions, and survey work in the area. In Section 2.6 we briefly highlight other security issues in multicast content distribution including source access control, secure multicast routing, and group policy specification.

2.2 Multicast Receiver Access Control

There are a number of available multicast routing protocols that provide the efficient transport mechanisms of multicast by routing packets with one group destination address to multiple recipients. The routing protocols must be aware of group members in the network in order to deliver packets to them. The mechanism provided for doing this is the Internet Group Membership Protocol (IGMP) [33]. A host uses this protocol to notify the routing system that it should deliver packets for a particular multicast group to this host. In the current model, any host can use IGMP to become a member of any IP multicast group causing eavesdropping or theft of service. The traditional method used to protect the information is to encrypt the multicast data and provide decryption keys only to authorized members (as discussed in Section 2.3). In some cases, encrypted communication is not possible for any number of reasons including legal issues or technical reasons. Even if encryption is used, there are still risks involved with unauthorized users receiving encrypted data such

as traffic analysis and possibly cryptanalysis. The current model is also vulnerable to a denial-of-service attack in which malicious hosts join a number of multicast groups utilizing large amounts of bandwidth or router resources.

Solving these problems requires controlling the ability of hosts to join the multicast group. We call this *multicast receiver access control.* The need for a solution to these problems is well-known and was first stated in [81]. The term *secure IGMP* has been used to refer to the protocol that would provide the solution.

2.2.1 Objectives

The primary objective of a multicast receiver access control system is to provide a means of restricting hosts' ability to join the multicast group. The secondary objective is to maintain scalability.

- Security: The system should be able to effectively restrict unauthorized receivers from joining the multicast group. This means restricting the ability of these users to access the data being delivered to the multicast group as well as stopping the users from establishing any state in the multicast routers.

- Message overhead: The system must also minimize communication overhead for each of the entities involved as well as minimze the overall network traffic overhead that is introduced.

- Computational overhead at the routers: In order to achieve scalability and to remain a lightweight system, the system must minimize the amount of computational overhead that is required of the routers.

2.2.2 Proposed Solutions

Figure 2 shows a classification of multicast receiver authorization solutions based on how they provide revocations. Some systems do not provide revocation, some systems leverage the authorization state maintained by some outside system, some

systems must query a centralized server to maintain authorization state, other systems distribute access control lists to routers, and some systems efficiently provide revocation using time-limited authorizations.

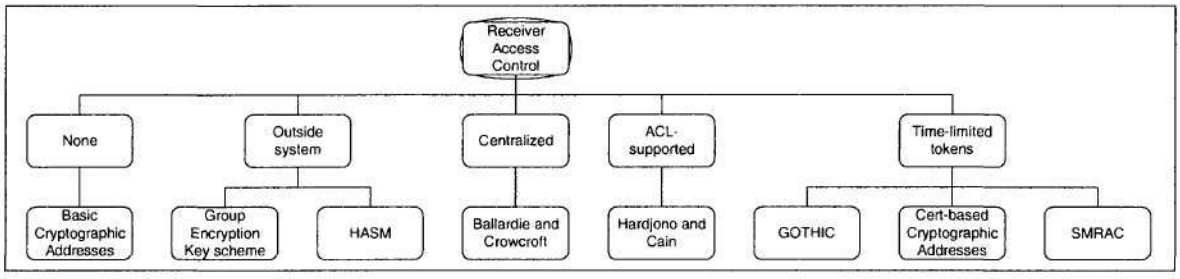


Figure 2: Multicast Receiver Authorization Systems

- Hardjono and Cain

  Hardjono and Cain [87] present a method for delivering keys to enable IGMP authentication and suggest a method of authorizing group members. The authorization server provides capability-like access-tokens to group members and access control list-like token lists to the routers. The host sends a join request including the access-token to the router that verifies that the access-token is in the token list. There are two vulnerabilities in the system: 1) Malicious users can perform a replay attack by presenting another user's access-token because the access-tokens are not related to the identity of the user. The system attempts to protect against this by having each router only accept a particular access-token once; however the same access-token can be used on any other router. 2) Malicious users can cause the router to accept fake access tokens because the issuer signature is not verified by the router. One inefficiency of this system is that all membership changes require distributing new token lists to all routers because of the use of a distributed ACL-based design for revocation.

- Ballardie and Crowcroft

  Ballardie and Crowcroft [22] provide an early survey of multicast security threats and present some countermeasures. Within the discussion, they present a version of IGMP that allows receivers to be authorized before joining the group. The architecture includes authorization servers that possess ACLs distributed by an initiator. The host sends a request to an authorization server to obtain an *authorization stamp*(AS) that is included in the join request sent to the router. The router forwards the host's request to the authorization server for approval. There are two vulnerabilities in the system: 1) An unauthorized user can obtain an authorization stamp by authenticating as itself, but then providing the spoofed address of an authorized user for authorization. This vulnerability is due to the fact that the AS uses the distinguished name to authenticate the host, but uses the IP source address to authorize the host. 2) An unauthorized user can cause the AS to accept an invalid authorization stamp such as one from a different group or one for a different user. This is because the AS only verifies the signature of the authorization stamp without verifying the information in it. One inefficiency of this system is that many of the authorizations and verifications are unnecessary because the authorization server actually only uses the router's interface address to authorize the request. This does happen to limit the damage of the two flaws mentioned above and causes them to not directly lead to unauthorized access.

- Standards work in progress

  Recently, there have been a number of efforts within the Internet Engineering Task Force(IETF) to standardize a multicast receiver authorization system. Castelluccia and Montenegro [36] propose the use of *cryptographically generated addresses* to restrict access to the multicast group. The authors propose a basic scheme that provides no revocation and a certificate-based scheme that provides time-limited revocation. He, et al. [93] discuss the *simple multicast*

receiver access control(SMRAC) system that also uses time-limited tokens. Coan, et al. [50] describe *HASM*, a multicast receiver and sender access control system that utilizes Kerberos tokens.

2.3 Group Key Management

In unicast, two users can provide confidentiality by encrypting data with a shared key. In multicast, *group key encryption* is used in which the multicast traffic is encrypted with a symmetric key and every authorized member of the group is given the decryption key. This becomes complicated by the case in which group membership is dynamic. Upon a change in membership, it is often necessary to change the group key so that the leaving member cannot access new broadcasts or so that the new member can not access old broadcasts. The term *leave* is used to describe the act of a voluntary leave or a forced leave. It is necessary to reduce the cost of updating the group key in these situations. When a new member joins, the new group key can be sent to the original group members using the old group key. However, when a member leaves, the solution involves more work. The simplest approach is, upon each leave, compute a new group key and send it to each user encrypted with its individual key. This is not acceptable because it requires n separate encryptions and transmissions for each join or leave. A simple improvement is to encrypt the new key with each user's individual key (resulting in n encryptions), but send all of the keys in one message to the entire group. This reduces transmission costs, but still requires n encryptions and causes the users to be able to detect their key among the group of keys in the received message. Work in *group key management* aims to provide efficient rekeying schemes for dynamic group memberships.

2.3.1 Objectives

- Scalability: A group key management solution should be able to handle large Internet groups. This requires low requirements for support infrastructure as well as low message and computational overhead.

- Forward and backward secrecy: Forward secrecy is the ability to keep leaving members from accessing future communication. Backward secrecy is the ability to keep new members from from accessing past communication. Some systems require forward secrecy but not backward secrecy.

- Collusion resistance: Collusion is when a set of authorized or unauthorized members work together to gain access to communication that they are not authorized to access. A scheme should be able to state its resistance to collusion of a group of c members.

- Message overhead for rekeying: Schemes should aim to provide efficient rekeying by reducing the message overhead necessary for rekeying on a join or leave to less than $O(n)$ as in the naive approach.

- Computational overhead for rekeying: In addition to reducing the message overhead, schemes should maintain minimum computational overhead for rekey operations.

- Storage overhead: Some schemes add storage requirements in order to reduce message overhead. Such schemes should maintain reasonable storage requirements for group members as well as for the group key controller.

- Reliability requirements: Many schemes make use of multicast to rekey the group. Schemes should account for the fact that multicast is unreliable by being robust to packet loss.

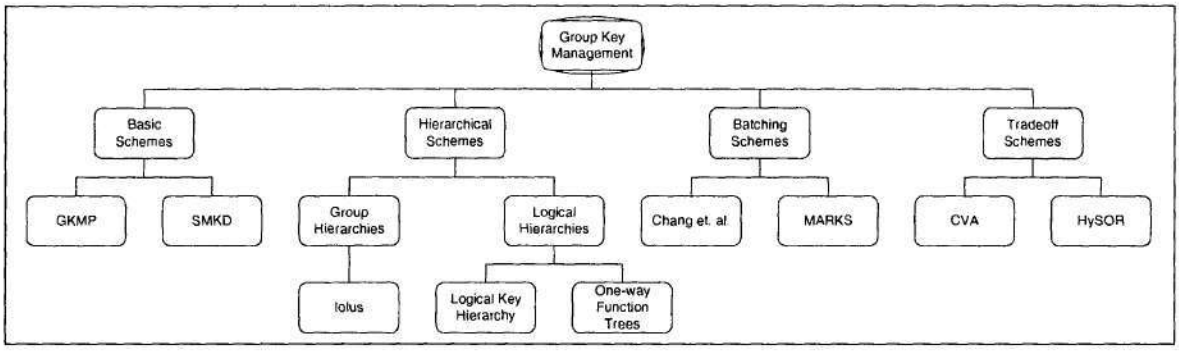


Figure 3: A Taxonomy of Group Key Management Solutions

2.3.2 Proposed Solutions

Work in group key management schemes includes basic schemes, hierarchical schemes, batching schemes and tradeoff schemes as shown in Figure 3. Basic schemes include the earlier work in group key management and did not focus on efficient rekeying. Hierarchical schemes include the first attempts at reducing rekeying overhead. Batching schemes attempt to further reduce rekeying overhead by not changing the key on every join or leave, but instead batching a number of joins or leaves before rekeying. It has been generally accepted and recently proven that $O(\log n)$ is the lowest overhead achievable by a group key management scheme if strict non-member confidentiality and non-collusion are required [193]. Tradeoff schemes attempt to provide lower than $O(\log n)$ overhead by trading off some collusion resistance.

- Basic Schemes
  - *Group Key Management Protocol (GKMP)* [89] assigns a group controller (GC) to manage the keys for each multicast group. The controller generates and maintains symmetric keys for each member. The GC selects a group member to generate the keys with. Then, it validates each group member's permissions and sends the group key encrypted by the individual key. This scheme does not provide a solution for efficient rekeying. It simply provides a method to avoid a single central key controller by

allowing a group controller per multicast group.

- *Scalable Multicast Key Distribution (SMKD)* [21] is based on the Core-Based Tree (CBT) routing protocol. This scheme takes advantage of the hard-state of the core based tree to provide secure joining for the CBT group tree. CBT is hard-state since each router knows its neighbor and the configuration does not change. The core of the tree performs the duties of a group controller and generates the group session keys and the key distribution keys. As the tree expands and new routers are added, the key distribution tasks are assigned to the routers. They are given the ability to authenticate joining members and give them the group key. This technique only works with core-based tree and requires trusted routers. It does not propose a solution to efficient rekeying.

- Group Hierarchies

  - *Iolus* [142] is a framework that divides the multicast group into a hierarchy of subgroups. The central or top group is managed by a group security controller (GSC) and group security intermediaries (GSIs) are used to manage the other subgroups. Each subgroup uses a separate subgroup key. Since there is no single group key and only subgroup keys, it is only necessary to generate a new subgroup key for the subgroup that is involved in the membership change. Each user in the subgroup must be transmitted a new subgroup key using its individual key. Iolus can also be used for encrypting and delivering the traffic. The GSI for each subgroup knows the keys for the neighboring subgroups. To send a message, the member sends the message to its local GSI using its individual key, the GSI sends the message to the group using the group's subkey and sends the message to the GSIs of any neighboring groups using that group's subkey. Instead of actually re-encrypting the message to send it to other subgroups, a random key is chosen to encrypt the message

and the GSI encrypts the key and sends it along with the message to the neighboring GSI. This reduces the computation costs of a message being sent through multiple GSIs.

- Logical Hierarchies
  - *Logical key hierarchies* [204, 209] use a hierarchy of keys to obtain a scalable solution rather than a hierarchy of groups. Each user that joins the group receives a secret key shared with the group key controller. The controller maintains a k-ary tree structure in which the root is the group key, the leaves are the n individual keys of the group members, and the intermediate keys are auxiliary keys used for key updates. Each member stores the set of keys in the path from its individual key to the root key This scheme allows the number of rekey messages to be reduced by allowing the new keys to be encrypted with subgroup keys rather than individual keys for the majority of the group. The operation of rekeying upon a join is similar. This scheme reduces overhead to $klog_k n$ messages for a rekey operation and requires members to store $log_k n$ keys.
  - *One-way Function Tree* (OFT) [20] is also based on tree hierarchy but uses a different method to generate keys for the logical subgroups. Keys at interior nodes in the logical hierarchy are derived from other keys using one-way functions and mixing functions. This allows group members to compute the new subgroup keys upon a group rekey. This scheme reduces the message overhead to $O(\log_2 n)$ for a key update, but requires members to store up to $O(2\log_2 n)$ keys.
- Tradeoff Schemes
  - *HySOR* [68] considers a range of protocols with varying message costs and vulnerability to collusion. In one extreme is logical key hierarchy that has $O(\log n)$ overhead and is resistant to collusion. On the other

extreme is a protocol based on the Linear Ordering of Receivers (LORE), which requires $O(1)$ messages for rekeying, but is vulnerable to any two receivers colluding. LORE uses two sets of auxiliary keys: forward keys denoted by f_i and backward keys denoted by b_i. All users are ordered and assigned a rank between 1 and n. A receiver with rank i, u_i, holds keys f_1 to f_i and keys b_i to b_n. In order to rekey the group when u_i leaves, the new group key is multicast to the group twice: once encrypted with f_{i+1} and once encrypted with b_{i-1}. Thus, all users with ranks higher than i and lower than i can decrypt the rekeying message. The authors present a scheme using a hybrid structuring of receivers(HySOR) which is tunable between the logical key hierarchy and LORE. HySOR uses a key graph where each leaf is a division of receivers and LORE is used to manage keys within each division. The authors show how an operator can tune the performance and collusion resistance by changing the number of divisions.

- *Complementary Variable Approach* (CVA) [204] is able to reduce message overhead to $O(1)$ but is vulnerable to collusion attacks. The controller generates n complementary variables j. Each member is assigned a rank i, $1 \leq i \leq n$. Each member i receives the group key and all complementary variables except j_i. To remove a member i from the group, a message is sent to all members stating "remove member i". The current group key and complementary variable j_i are used to create a new group key with some deterministic key variable generation process. Thus, all members except i are able to compute the new group key. This scheme reduces message overhead to $O(1)$, but is vulnerable to collusion and requires a storage overhead at the group members of n.

- Batching Schemes

  - *Boolean function minimization* technique [38] batches membership changes

to reduce rekey overhead. The authors refer to this as *cumulative member removal.* The authors also present a new logical key hierarchy algorithm for rekeying that uses a set of auxiliary keys and dynamically generates a logical key hierarchy by composing different keys. This rekeying scheme achieves the same $O(\log n)$ overhead as the other logical key hierarchy approaches, but reduces the storage overhead at the group controller to $O(\log n)$ as opposed to $O(n)$.

- *MARKS* [31] divides the group session into time slots, assigns one key for each slot, and changes the group key every time slot. MARKS also presents the technique of generating condensable key space with binary hash trees. MARKS involves no message overhead for rekeying during the group session. However, MARKS is limited in that it requires receivers to determine their leave time when they join the group; this cannot be met in many applications. Also, MARKS has problems with situations in which a receiver may join and leave a group multiple times over a given session.

2.4 Group Source Authentication

Source authentication is the ability of group members to verify the identity of the sender of a received packet. In unicast, a shared secret-key message authentication code (MAC) is used to provide authentication. In multicast, the group key provides a shared secret-key; however, performing message authentication with this key only verifies that the sender is a member of the group, but not necessarily the intended source. Many applications require a level of authentication that allows a receiver to identify the individual sender of a message. There has been work that aims to efficiently provide this level of source authentication.

2.4.1 Objectives

The design objectives of a source authentication scheme should include the following:

- Authenticity: The receiver must be able to verify the identity of the data's source. One level of functionality is that the receiver can verify that the data is from a group member. The next level of functionality is that the receiver can verify that it is from an authorized sender. The most precise functionality is that the receiver can determine the exact identity of the sender.

- Integrity: The receiver should be able to verify that the received data has not been modified. Some schemes provide only authentication without integrity checking.

- Non-repudiation: Non-repudiation requires the ability to prove that a host sent a particular message. This prevents the sender from later denying the transmission of the message.

- Efficiency: The efficiency of the solution is based on communication, storage, and computation overhead at the source and the receivers.

- Collusion resistance: The scheme should provide protection against collusion or at least be able to state in a provable manner the level of protection against c-collusion.

- Minimal latency: Some schemes require a certain number of packets to be stored before they can be signed or verified. For some real-time applications, this can introduce an intolerable delay.

- Robustness against unreliable communication: Some designs are based on an assumption of reliable communication. Some multicast environments do not provide reliable multicast communications; therefore such schemes are unsuitable for these environments.

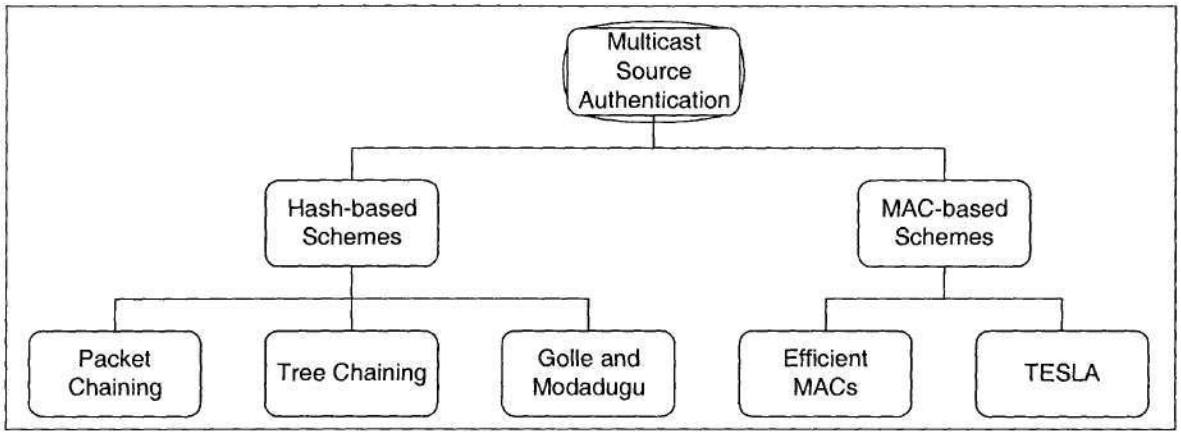


Figure 4: A Classification of Multicast Source Authentication Schemes

2.4.2 Proposed Solutions

As shown in Figure 4, there have been two approaches in multicast source authentication schemes: hash-based schemes and MAC-based schemes.

- Hash-based Schemes

  Digital signatures provide a simple method of individual authentication. However, due to the computational costs of computing and verifying digital signatures, signing each packet is not a practical solution.

  - Packet Chaining

    Gennaro and Rohatgi proposed packet chaining, a solution to efficiently authenticating digital streams [77] that allows only the first block to be signed and contain an association with subsequent packets. The stream of data packets is partitioned into chains and each packet in the chain contains a hash of the next packet in the chain. Thus only the first packet in the chain must be signed. This works for streams that are finite and in which the data is known in advance. For infinite streams, multiple one-time signatures are used.

- Tree chaining

  Wong and Lam [210] proposed tree chaining, a technique that partitions the stream of data packets into blocks and forms a tree structure to perform authentication. Each block of n messages can be authenticated with one signature. It differs from Gennaro and Rohatgi's approach because the association made between packets is a tree-based association rather than a linear one. Each leaf node is a message digest of a data packet and the parent nodes are message digests of the two children nodes. The root node is the message digest for the block which is signed once for the entire group. The data packet is sent along with the block signature, the packet position in the block, and the siblings of each node in the packet's path to the root. In order for the receiver to verify the received packet, it recreates the path from the received packet up to the root. The digest of the received packet is computed and is used to recreate each node along the path. If the root that is computed by the receiver is the same as the signed one that was received with the packet, then the packet is verified. The receiver can cache the nodes so that it is possible to verify all the packets in the tree by computing each node in the authentication tree no more than once.

- Golle and Modadugu

  Due to the association between packets, the above approaches are sensitive to data loss. Golle and Modadugu [79] proposed a hash-based scheme that aims to be robust against bursty packet loss. It achieves robustness by replicating packet signatures across multiple packets in the stream. The final packet also includes a signature. The authors provide results that show the burst tolerance of the scheme based on the efficiency resources.

- Hybrid Signatures

  Rohatgi later proposed a scheme that makes use public key digital signatures as well as faster one-way function-based k-time signatures [178]. The scheme creates sets of k-time key pairs offline and uses the normal digital signature to certify the public k-time keys. Message signatures are created online using a k-time private key and the certified k-time public key. The scheme avoids the need for reliable communication by sending the k-time keys more than once.

- MAC-based schemes

  There have been schemes proposed that use message authentication codes to provide authentication rather than digital signatures to increase efficiency.

  - Efficient MACs

    Cannetti, et al. proposed a scheme that makes use of efficient MACs [35]. In this scheme, the sender holds a set of l MAC keys and each group member holds a subset of the l keys. Each message is then MACed with each of the l keys and the recipient verifies the MAC with the keys that it holds. The authors show that appropriate choice of subsets provides a high probability of protection against c-collusion.

  - TESLA

    Perrig, et al. proposed TESLA, a MAC-based scheme that provides authentication without regards for the packet loss rate [158]. The scheme involves the source signing the first packet and providing notification of a chain of MAC keys. Each packet P_i is authenticated with a MAC using a key K_i. Later packets reveal each K_i. The scheme requires some time synchronization between the sender and the receivers since each packet must be received before the next packet is sent.

2.5 Multicast Fingerprinting

Encryption is generally used to safeguard the content while it is being transmitted so that unauthorized persons can not read the stream from the network, but this offers no protection after the intended receiver receives the data. There is no protection against unauthorized duplication and propagation by the intended receiver. Watermarking can provide protection in the form of *theft deterrence*. *Watermarking* is the embedding of some identifying information into the content in such a manner that it can not be removed by the user but it can be extracted or read by the appropriate party. Watermarks can be used for copyright protection or for identification of the receiver. Copyright protection watermarks embed some information in the data to identify the copyright holder or content provider, while receiver-identifying watermarking, commonly referred to as *fingerprinting* [203], embeds information to identify the receiver of that copy of the content. Thus, if an unauthorized copy of the content is recovered, extracting the fingerprint will show who the initial receiver was.

In multicast environments, traditional fingerprinting or embedding the receiver's identification as the watermark at the source will not work since all the receivers will share the same watermark. It is necessary to watermark content with unique information for distinct receivers of the same multicast stream. A simple method to achieve unique watermarks for each receiver would be to watermark the stream differently for each receiver and to unicast the watermarked streams. Of course, the inefficiency of such a scheme calls for a better solution. The goal is to maintain the security of this approach while achieving scalability.

2.5.1 Objectives

The design objectives of a system to fingerprint multicast content should be security and scalability. We outline the concepts involved in achieving these goals. The features and components of the system necessary to accomplish these goals should

be designed into the solution.

- Security:
  - Robustness of the fingerprinting method: The fingerprint is what distinguishes one user from another. This can be a particular pattern of frames or a particular pattern embedded in a frame. The method used must be robust to efforts of a user to remove this distinguishing information. There has been significant work in multimedia watermarking. A scheme extending these efforts into fingerprinting multicast content is desirable since it assures a robust fingerprinting method.
  - Collusion problem: Collusion is when a set of group members work together to use the set of differently watermarked streams to create a copy of the content which cannot be determined to contain the fingerprint of any of those receivers. The solution must be based on a fingerprinting scheme that is not susceptible to collusion.
  - Asymmetric fingerprinting: Schemes should be able to provide asymmetric fingerprinting. This allows the sender to identify the receiver of a recovered copy of data without previously knowing the fingerprinted data. Thus, the sender is not capable of distributing the data and accusing an innocent receiver [161].
  - Protection Granularity: The granularity of protection is the amount of content that is needed for the protocol to be able to determine the receiver of the content. Schemes should be able to provide the smallest possible protection granularity but also be flexible so that this can be changed depending on the needs of the application.
- Scalability:
  - Logging Requirements: Logging is necessary because once the content is recovered and the fingerprint is extracted, there must be some record of

what receiver was represented by the ID recovered from the watermark at that instant in time. The storage and processing overhead of logging should be minimum.

- Efficiency: The efficiency of the solution is based on the amount of data that the source must transmit and encrypt and the amount of data introduced into the network.

2.5.2 Proposed Solutions

Figure 5 shows that there have been four classes of multicast fingerprinting solutions depending on where the watermarking takes place. Client-side marking schemes involve some client software that watermarks the content. Application-level schemes add logic to the application to deliver unique versions of the content. Network-level schemes involve computation in the network that causes each user to receive a unique version of the content. Overlay-based schemes involve intermediaries in the content distribution path that uniquely watermark the content for receivers.

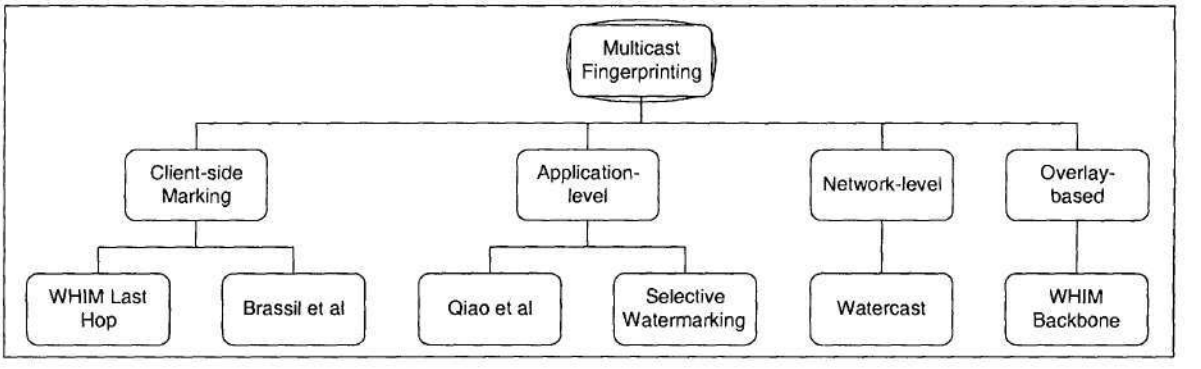


Figure 5: Multicast Fingerprinting Solutions

- Text Content

  Brassil, Low, and Maxemchuck [30] proposed one of the first solutions for multicast fingerprinting. This system is designed for text documents and involves

multicasting the text documents and marking them at the client's machine. The source then unicasts to each user a decryption program that contains the user's unique identification number. The program decrypts the document and inserts the identifying mark for that user. The authors note that since it is possible to reverse engineer the program and extract the decryption key, the key must be changed periodically which means that the entire program must be changed. One inefficiency is that in order to rekey the group, the documents are encrypted with a new key and programs with the new key built in must be unicast to every group member.

- Application-based approach

  Chu, Qiao, and Nahrstedt [46] proposed a protocol to provide a different version of a multicast video stream to each group member. The protocol creates two watermarked MPEG streams, assigns a unique random binary sequence to each user, and uses this sequence to arbitrate between those two watermarked streams. For the ith watermarked frame in stream $j(j = 0, 1)$, a different key KEY_i^j is used to encrypt it. Then user n is given either KEY_i^0 or KEY_i^1 depending on the random bit sequence of user n. The efficiency of this protocol is hampered by the need to watermark, encrypt, and transmit two copies of the stream and by the significant amount of key messages sent. The ability of the protocol to detect a collusion is dependent on the length of the retrieved data stream. Even with a retrieved data stream of sufficient length, the algorithm to determine a collusion is so complex that there is not a known length of retrieved stream that can guarantee a c-collusion detection, where c is the number of colluders. The protection granularity of this protocol is large since it is based on the number of receivers.

- Watercast

  Brown, Perkins, and Crowcroft [32] proposed a technique that has each group member receive a slightly different version of the multicast video stream. For

a multicast group with a tree of depth d, the source creates n differently watermarked copies of each packet such that $n > d$. On receiving a transmission group of packets, each router forwards all but one of the packets. The last hop router then forwards exactly one packet to the subnet with the receiver(s). The goal is that each receiver then receives a stream that consists of a unique combination of watermarked packets. The original receiver of a recovered stream can be determined by simulating the operation of various network components during the time that clip was originally transmitted. This makes the logging requirements high since the log must keep the state of the entire network from the start to the end of the transmission. The requirement that the source watermark, encrypt, and transmit n copies of the stream makes this solution inefficient. The scheme does not offer a solution for having multiple receivers on the same subnet since they will have the same User ID. The protection granularity is large because as the length of the clip increases, the probability of being able to specify a single receiver increases. Also, the ability to determine collusions is dependent on the length of the clip and requires extensive computation to determine what users could have possibly had access to the frames in the recovered stream.

- Selective Watermarking

  Wu and Wu [198] proposed a technique that multicasts most of the video and uniquely watermarks and unicast a portion of the video. Depending on the specific selection scheme used, the chosen segments could be from 90% to less than 1% of the original video. There is a tradeoff between efficiency and security. As smaller amounts of the video are chosen for encryption and watermarking, the ability of persons outside of the group to obtain the video increases due to the proposal of not encrypting the video that is not watermarked and the ability of group members to obtain video that is not watermarked increases due to the fact that if only I frames are watermarked, then unwatermarked

I-blocks found in P and B frames can provide some degree of quality video. As larger percentages of the video are chosen to be watermarked, encrypted, and unicast, the security increases, but the efficiency of the protocol begins to resemble that of the simple unicast model. Since only I frames are watermarked, the protection granularity is each set of the I-frame pattern.

2.6 Other Multicast Security Research Areas

- Secure Multicast Routing

  Shields and Garcia-Luna-Aceves [188] proposed Keyed HIP (KHIP), a secure hierarchical multicast routing protocol. The authors show that multicast routing protocols are vulnerable to attacks against the routing infrastructure that can cause denial-of-service by creating routing loops or blackholes. KHIP provides authentication mechanisms that allow only trusted routers to join the multicast tree. The authors also state the need for a multicast receiver access control architecture and explain that it would complement KHIP.

  There has been work that aims to add security mechanisms to the PIM-SM multicast routing protocol [199]. This work is still in progress, but aims to provide protection for PIM-SM similar to that provided by KHIP.

- Sender Access Control

  The problem of controlling which hosts can send data to a group is a separate problem from receiver access control. This is because IGMP is not used to register multicast senders.

  Ballardie and Crowcroft [22] proposed a scheme to detect and prevent unauthorized multicast traffic. This scheme requires each packet to include a timestamp and an authorization stamp. Upon noticing multicast traffic from a new source, a router forwards a copy of the packet to the authorization service that verifies that the authorization stamp was created by a host that has the rights

to send data for that particular multicast group and verifies that the timestamp is current. If the verifications fail, the router is notified and is required to send an alert upstream towards the source in order to have all routers block traffic from the unauthorized source.

One viewpoint is that sender access control is becoming less of a problem with recent multicast schemes such as source-specific multicast (SSM) [99] that inherently provide sender access control. Recent discussions within the Internet Engineering Task Force (IETF) have maintained that receiver and sender access control should be solved separately but have considered a scheme similar to secure IGMP for sender access control.

- Group Security Policy

  Multicast group policy is an important element of securing multicast content distribution. It deals with specifying the parameters and mechanisms involved with securing the group.

  McDaniel, et al. [136] presented requirements for policy management in secure groups. This work explains that requirements include the specification, distribution, evaluation, and enforcement of policy. The authors show that previously there were two types of systems with regard to group policy. Trust management systems specified and evaluated policy in a well-defined manner, but lacked the ability to enforce them. Policy directed secure group communication systems defined and implemented policies, but do not always maintain secure distribution and composition of the policies.

  Another problem in group security policy is verifying the entity that is allowed to specify the group's policy. This entity is usually the group owner, but determining who is the group owner and authenticating an entity to be the group owner can be a complex task. In [110], the authors examine this problem and propose two solutions for a *group owner determination and authentication system*(GODAS).

2.7 Video Watermarking

This section reviews proposed video watermarking schemes and mentions the advantages and disadvantages of each. This section is divided into two sections based on the domain in which the watermark is inserted for each scheme. The first section, compressed, reviews schemes that embed the watermark in the compressed video stream. The second section, uncompressed, reviews schemes that embed the watermark in a raw or uncompressed video stream. Another classification of watermarking systems is based on the information that is needed to retrieve the watermark. Private marking systems require at least the original unwatermarked image. Whereas, public marking systems do not require the original unwatermarked images. Petitcolas, et. al. [160] provide a survey of information hiding.

2.7.1 Compressed Techniques

Dittman, Stabenau and Steinmetz [61] proposed a technique that inserts a robust watermark in MPEG video while avoiding artifacts. First, a position sequence is generated from the user key as a seed with a secure random number generator. This sequence is used for hiding the watermark in the frame. Second, smooth and edge blocks are detected to improve the visual quality, then the watermark information is coded with error corrections and redundancy. Finally, three coefficients are selected to be used to embed each bit of the watermark information in a block as in the Zhao-Koch algorithm [123]. The Zhao-Koch scheme involves selecting two or three coefficients in the block and modifying the values of these coefficients so that the relationship between the values denotes the bit that is embedded in that block. This scheme has the advantages that it is a public marking system, artifacts are avoided by using smooth block and edge recognition schemes, and error correcting codes and redundancy is used to increase the robustness of the watermark. It also has the disadvantage that StirMark [159, ?] causes high error rates.

Hartung and Girod [90] proposed an idea that embeds the watermark in compressed video. Let $a_j(-1 or 1)$ be a sequence of information bits we want to hide in the video stream. We then make the spread sequence $b_i = a_j, j * C_r <= i < (j+1) * C_r$, where C_r is the chip-rate. The watermark is constructed as $w_i = Alpha * b_i * p_i$, where $Alpha$ is an amplitude factor and $p_i(-1 or 1)$ is a binary pseudo-noise sequence. In order to add a watermark, we process the encoded video block by block. For each block, we do a zigzag scan, yielding a $1x64$-vector of re-scanned DCT coefficients. For the DC-coefficient, we add the DC-coefficient of the watermark block to that of the encoded video block, obtaining the DC-coefficient of the watermarked block. For the AC- coefficients, we do the same add operation as long as the number of bits to transmit for the watermarked AC-coefficient does not increase. Hence, usually only few DCT coefficients of the watermark can be incorporated per $8x8$ block. The scheme has the advantages that the watermark does not increase the bit rate. It has disadvantages such as the vulnerability to collusion and limited amount of embedded information due to the bit-rate constraint. Also, since it is a private marking system, recovery of the hidden information requires the use of the same pseudo-noise sequence p_i that was used in the coder,

In Holliman, Memon, Yeo, and Yeung [101] proposed an adaptive scheme to embed watermark information in DCT blocks of image data. The scheme makes use of an algorithm to select the appropriate blocks for watermark insertions and a block-dependent seed generation algorithm to determine which coefficients to modify in a particular block. The algorithm attempts to reduce artifacts by not marking smooth and edge blocks. Smooth blocks are those that the number of non-zero coefficients in the lower right half of the DCT block are less than a threshold min_Z. Edge blocks are those that contain any unquantized coefficients with an absolute value exceeding a threshold min_E. To determine which coefficients of a particular block are modified, the authors suggest selecting bits of that block and concatenated with bits from some previous blocks and the private key, to be used as the seed to the pseudo-random number generator. The bit is embedding in the block in the same

way as the Zhao-Koch scheme using two coeffiecients [123]. Advantages of this scheme are that it is a public marking system, it avoids artifacts by using smooth block and edge recognition schemes, and does not significantly increase the bit rate. The disadvantage is that it is only shown to be robust against JPEG compression attacks

2.7.2 Uncompressed

Hartung and Girod [91] propose an algorithm that allows public retrieval of the watermark. One problem for most watermarking techniques based on spread spectrum communications is that retrieval of the watermark requires the same pseudo-noise sequence pi used for embedding of the watermark. Hence, decoding of the watermark is not public since this would potentially allow attacks on the watermark. The algorithm resolves this problem by making only parts of the pseudo-noise sequence pi public. At the same time, the hidden information can be retrieved in the same manner. A modified pseudo-sequence is public where each n-th coefficient is taken from the original pseudo-sequence pi and all other coefficients are arbitrary random values with the same distribution as pi. The advantage of this scheme is that the watermark can be retrieved and verified publicly. However, the disadvantage is that the robustness of the publicly decodable watermark is lower than the robustness of the non-publicly decodable watermark.

Dittman, Stabenau, and Steinmetz [61] suggested a way to embed a watermark in the spatial domain of an image. The algorithm overlays a $8x8$ pattern over every $8x8$ block of the frame. First, a position sequence is generated to determine the blocks will be modified. Second, for each block a user key dependent pattern is made based on the inserted bit. Lastly, the created pattern is added to the original block. The advantages of this scheme are that it is a public marking system, it is resistant to the collusion attack, and it can embed a large amount of bits. It also has the disadvantage that StirMark [159, ?] causes high error rates.

Qiao [165] proposed a solution to embed a watermark in an uncompressed video stream that resolves the rightful ownership problem. Given an original image V, a key KEY is chosen. Then the watermark $W = (w_i)$ is created by applying a standard encryption function such as DES, i.e., $w_i = DES_{key(v_i)}$, where v_i is the i-th pixel of V. Watermarked V_w is constructed as: $vw_i = v_i x(1 + Alphaxw_i), Alpha = 1$. This solution has the advantage that it is non-invertible and it resolves the rightful ownership problem. The rightful ownership problem is when an attacker can manipulate the watermarked video and claim that he/she also is the original owner. The disadvantage of the scheme is that it is a private marking system.

Hartung and Girod [90] proposed a solution to embed a watermark in raw video using ideas from direct-sequence spread spectrum communications. Let $a_j(-1or1)$ be a sequence of information bits we want to hide in the video stream. The spread sequence $b_i = a_j, j * C_r <= i < (j + 1) * C_r$, where C_r is the chip-rate is created. The watermark is constructed as $w_i = Alpha * b_i * p_i$, where $Alpha$ is an amplitude factor and $p_i(-1or1)$ is a binary pseudo-noise sequence. Then the watermark is added to the line-scanned digital video signal v_i yielding a watermarked video signal $vw_i = v_i + w_i$. The advantages of this scheme is htat it is more robust than the scheme presented in this same paper that inserts the watermark in the compressed domain. The disadvantages are that it is a private marking system and it does not solve the collusion problem.

2.8 Related Network Security Work

There has been previous work in rights management or content protection for centralized peer-to-peer system, but not for decentralized peer-to-peer system. Outside of peer-to-peer, there has been work that does not share the same goal as CITADEL but is somewhat related. Related work includes authentication, authorization, and trust management systems for distributed environments as well as other rights management work.

- Centralized Peer-to-Peer Content Protection: The content protection system that was implemented as part of Napster was one of the most well-known [122]. This system relied on the central authority that maintains the indexing and location functions to provide content protection. Content identification was done based on the file name of the content. This approach used a blacklist of forbidden content. The system controlled the sharing of blacklisted files by not allowing users to locate these files. This was accomplished by either not allowing users to add these files to the index or not responding to queries for these files. This approach proved to be easily bypassed by users simply changing the file names. Ultimately, the content providers insisted that this system did not provide adequate protection and Napster was forced to shut down until it can provide adequate content protection functions [51].

- Peer-to-Peer Security: There have been a few different types of work in security for peer-to-peer security. These systems either focus on protecting the system, the file retrievers, or the file providers. Work in security of the peer-to-peer infrastructure includes work by Sit and Morris [192] and Castro, et al. [37]. Secure Overlay Services [121] provides a proactive system for preventing denial of service attacks and is also relevant to peer-to-peer systems. Work in anonymous systems include Anonymous Peer-to-Peer File Sharing (APFS) [185] and Freenet [74]. Work in censorship resistant peer-to-peer systems include Publius [131] and Eternity [15].

- Digital Rights Management (DRM) A number of commercial DRM solutions are offered such as Microsoft Rights Manager [3]; however, the details of most of the systems are not published. Park, et. al. [154] provide a taxonomy of architectures for controlling the dissemination of digital information. Judge and Ammar [111] discuss how watermarking technology can be used to achieve various DRM goals in peer-to-peer systems. Feigenbaum, et. al. [70] discuss

[3] http://www.microsoft.com

privacy issues in DRM systems.

- Protected Content Formats: The idea of protection labels and attaching protection labels to the objects that they describe has been in the security literature for some time [25]. Slightly more recent work extended this into the concept of a secure package for storing content and its controls [113, 190].

- Authentication and Authorization in Distributed Systems: There has been much work in access control models including traditional mandatory and discretionary access control (MAC and DAC) [162] and works that extend beyond MAC and DAC such as role based access control [181] and the dissemination control model [135]. Neumann proposed an authorization system for distributed systems [150]. Wong and Lam describe a distributed authorization service [211]. Hayton, et al. proposed the Oasis architecture for access control in distributed environments [92].

- Representation of Authorization Information in Distributed Systems: Approaches to representing authorization information in distributed system include extensible rights markup language (XrML) [67], extensible media commerce language (XmCL) [66] and generalized access control list (GACL) [211].

- Trust Management in Decentralized Systems: The trust management problem involves creating security policies, verifying that certain credentials are adequate based on the security policy, and deferring trust to third parties. KeyNote [26] provides a comprehensive system for trust management.

Chapter 3

Theft Deterrence using Fingerprinting in Multicast Environments

Encryption is generally used to safeguard the content while it is being transmitted so that unauthorized persons cannot read the stream from the network. However, end-to-end encryption offers no protection against unauthorized duplication and propagation by the intended receiver. This additional protection can be obtained by *watermarking* the content. Watermarking is the embedding of some identifying information into the content in such a manner that it cannot be easily removed by the user but it can be extracted or read by the appropriate party. Watermarks can be used for copyright protection or for identification of the original receiver after the data is propagated. *Copyright protection watermarks* embed some information in the data to identify the copyright holder or content provider, while receiver-identifying watermarking, commonly referred to as *fingerprinting* [203], embeds information to identify the receiver of that copy of the content. Thus, if an unauthorized copy of the content is recovered, extracting the fingerprint will show who the initial receiver was.

Problems arise when attempting to fingerprint content in a multicast environment that do not arise in copyright protection watermarking. If copyright protection watermarks are embedded in the data at the source, then the watermarked data is multicast to the group of receivers. For fingerprinting, embedding the receiver's identification as the watermark at the source will not work since all the receivers will share the same watermark. It is necessary to watermark content with unique

information for distinct receivers of the same multicast stream. A simple method to achieve unique watermarks for each receiver would be to watermark the stream differently for each receiver and to unicast the watermarked streams. Of course, the inefficiency of such a scheme calls for a better solution. We aim to maintain the security of this approach while achieving scalability.

We propose WHIM [109], a scalable system that allows multicast content to be securely marked with distinct information for each receivers. This system introduces two new concepts: 1)generation of a watermark based on the receiver's location in the network; and 2) incremental insertion of the watermark in content as it traverses the network. WHIM makes use of a hierarchy of intermediaries for creating and embedding the fingerprint. This allows security and scalability. The use of a hierarchy allows a new type of security by having a User ID based on the user's location in an overlay network. Security is also maintained by using proven watermarking algorithms to embed this User ID. The hierarchy leads to scalability by capitalizing on the efficiency of multicast distribution and by distributing the watermark embedding load from the source to the different intermediaries.

This chapter proceeds as follows. In Section 3.1 we enumerate the design objectives of WHIM. Section 3.2 gives an overview of the WHIM architecture. Section 3.3 discusses the WHIM-Backbone component which is based on a hierarchy of intermediaries that provide an efficient distribution architecture that fingerprints the streaming content. Section 3.4 describes the WHIM-Last Hop component, a secure protocol that fingerprints and distributes content between an intermediary and a group of receivers. Section 3.5 presents an analysis and simulation results of the efficiency of WHIM, and a comparison with previous solutions. Finally, Section 3.7 presents conclusions.

3.1 Objectives

The design objectives of a system to fingerprint multicast content should be security and scalability. We outline the concepts involved in achieving these goals. The features and components of the system necessary to accomplish these goals should be designed into the solution.

Security:

Robustness of the fingerprinting method:

The fingerprint is what distinguishes one user from another. This can be a particular pattern of frames or a particular pattern embedded in a frame. The method used must be robust to efforts of a user to remove this distinguishing information. There has been significant work in video watermarking see for example [61, 165, 90, 101]. A scheme extending these efforts into fingerprinting multicast content is desirable since it assures a robust fingerprinting method.

Collusion problem: Collusion is when a set of group members work together to use the set of differently watermarked streams to create a copy of the content which cannot be determined to contain the fingerprint of any of those receivers. The solution must be based on a fingerprinting scheme that is not susceptible to collusion.

Asymmetric fingerprinting: Schemes should be able to provide asymmetric fingerprinting. This allows the sender to identify the receiver of a recovered copy of data without previously knowing the fingerprinted data. Thus, the sender is not capable of distributing the data and accusing an innocent receiver. [161]

Protection Granularity: The granularity of protection is the amount of content that is needed for the protocol to be able to determine the receiver of the content. Schemes should be able to provide the smallest possible protection granularity but also be flexible so that this can be changed depending on the needs of the application.

Scalability:

Logging Requirements: Logging is necessary because once a video is recovered and the fingerprint is extracted, there must be some record of what receiver was represented by the ID recovered from the watermark at that instant in time. The storage and processing overhead of logging should be minimal.

Efficiency: The efficiency of the solution is based on the amount of data that the source must transmit and encrypt and the amount of data introduced into the network.

3.2 WHIM Architecture Overview

Our system has two components, WHIM Backbone (WHIM-BB) and WHIM Last Hop (WHIM-LH). WHIM-BB introduces a hierarchy of intermediaries into the network and forms an overlay network between them. Figure 6 illustrates how the hierarchy is formed as an overlay network in the physical network. We distinguish each intermediary by its unique path from the source. This Path ID composed of intermediary IP addresses is embedded into the content to identify the path that it traveled. Each intermediary embeds its portion of the Path ID into the content it forwards the content. This embedding is performed using modified versions of existing video watermarking algorithms. This is along the lines of the recent trend towards introducing a hierarchy of entities into the network to provide active services, such as reliable multicast [130, 156], Internet caching [69, 39, 114], multimedia proxy servers [187], and layered video multicast [128].

Each intermediary can have a set of child intermediaries and receivers. We call this set of child receivers the intermediary's domain. A watermark embedded by WHIM-BB identifies the domain of a receiver. Some literature suggests that identifying the domain of the receiver or the last hop before the receiver is adequate protection [32]; however, we feel that it is necessary in many applications to identify

the individual receiver. So, we propose WHIM-LH, which allows intermediaries to mark the content distinctly for any children receivers that they might have. WHIM-LH forms a domain-wide secure distribution and fingerprinting system including key distribution and logging.

A central component of WHIM-LH is a secure client-side fingerprint insertion program that communicates with the intermediary for registration and to receive the decryption keys and the stream. The security of this component can be achieved by using techniques such as Mobile Cryptography [180] and Time-Limited Blackbox Protection [97]. Clients join and register for the group at the domain level. This type of control is ideal for applications in which domains are responsible for the activity of its members. For example, a university might subscribe to a site-wide license for a broadcast then have students subscribe individually to receive it.

WHIM-LH is a building block that when merged with WHIM-BB forms a robust layered solution for fingerprinting multicast content distinctly for each receiver in the group. Used together, WHIM-BB and WHIM-LH allow content to be marked to pinpoint the location of the receiver in the overlay network as well as to identify the individual receiver. WHIM protects against attacks in which receivers join a group using a fake IP address or name. Even if the WHIM-LH registration fails to lead to the actual receiver, the WHIM-BB Path ID will pinpoint the responsible domain. It should be noted that either of these can be used alone as a suitable fingerprinting system. WHIM-BB, alone, offers a fingerprinting system that identifies the domain of the receiver, but not the individual receiver. WHIM-LH can be used between the source and the group of receivers to fingerprint the content uniquely for each receiver. However, it lacks the scalability of the combined solution due to the lack of the distributed architecture and it does not provide any information regarding the location of the receiver.

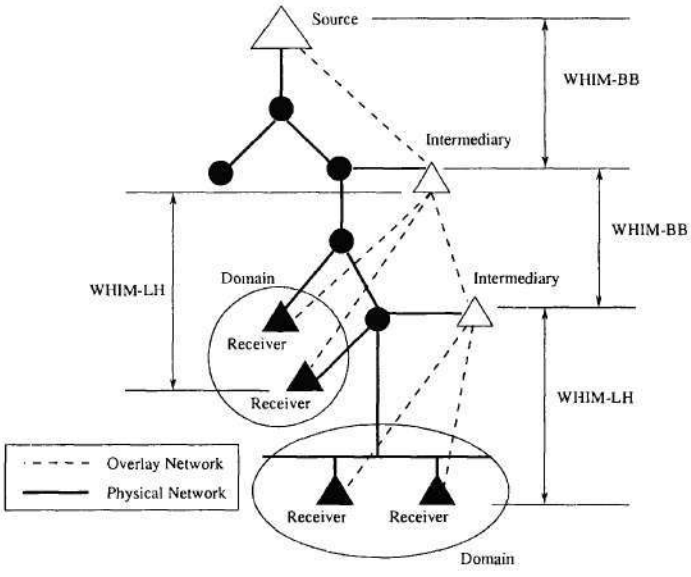


Figure 6: The Hierarchy of Intermediaries as an Overlay Network

3.3 WHIM Backbone (WHIM-BB)

WHIM-BB makes use of a hierarchy of intermediaries for creating and embedding the fingerprint. The fingerprint is based on the path from the source to the intermediary. This increases the security of the system and the scalability of the watermark embedding. Use of a hierarchy allows a new type of security by having the user's fingerprint based on the user's location in the network. Security is also maintained by using proven watermarking algorithms to embed this identifying information. The hierarchy allows scalable watermark embedding by distributing the embedding load from the source to the different intermediaries and by easing logging requirements. This section first describes the architecture of intermediaries, then discusses the distributed watermarking algorithms used by the intermediaries, and finally, discusses the logging necessary to maintain the path information.

3.3.1 Architecture

Our architecture consists of a hierarchy of intermediaries positioned as end systems in the network. Each intermediary is assigned a unique ID either manually or

using some prefix labeling algorithm [19]; so to identify the intermediary, there exists a unique ID that identifies each path from the source to each intermediary. As the content traverses the network, every intermediary through which it passes concatenates its ID to the Path ID already embedded in the content.

The amount of computation required to insert the watermark is more than routers today are capable of and possibly even more than the amount of processing power proposed by advocates of active networking [34, 195]. Therefore, WHIM-BB places a hierarchy of intermediaries as end-systems in the network and forms an overlay network between them. This overlay architecture lends itself to end-system or application-layer multicast [72, 47, 42]. There has been research that makes a case for application-layer multicast stating that it can help avoid many of the problems involved in using an IP multicast distribution model such as congestion control and end-to-end reliability and even increase security. These works have proposed protocols for enabling application-layer multicast. Other works such as the X-bone [197] propose systems for dynamically deploying and managing overlay networks. WHIM's architecture can use application-layer multicast rather than rely on global IP multicast support while still using IP multicast where available, especially within domains.

This idea can be extended to allow the intermediaries to be coupled with existing machines in the network that perform computation. Infrastructures in place for multimedia applications [48], multimedia proxy servers [187], server replication, and caching [39, 114, 69] provide ideal locations for WHIM intermediaries to be located.

3.3.2 Distributed Watermarking Algorithms

The fingerprint is the information embedded into the content to uniquely identify the recipient. The identifying information consists of a timestamp and the concatenation of all the IDs of the intermediaries on the path. This identifying information is embedded into each frame of the multimedia content. WHIM-BB embeds the

fingerprint incrementally at each intermediary. Existing watermarking methods are designed to embed an entire watermark at once. We propose distributed watermarking algorithms that allow existing watermarking algorithms to be used in a distributed manner securely and efficiently.

Example 1 The watermarking algorithm described by Dittmann, et al. [61] works as follows. For each frame, a pseudo random sequence is calculated to determine the order in which the blocks will be marked. In the determined order, the blocks are discrete cosine transformed; smoothness and edge detection is done; and the blocks are quantitized with Qm/Qf accordingly. For each block, the information is embedded as in the Zhao-Koch algorithm [123].

Our distributed version of this algorithm performs as follows. The source creates the pseudo random sequence in which the blocks will be watermarked, does smooth and edge detection for each block, and quantitizes with Qm/Qf. The watermark begins with a timestamp inserted by the source. It then sends the new frame and the sequence towards the receivers. As each intermediary receives the stream, it uses the sequence to determine the next blocks to watermark, adds its ID to the watermark, and sends the remainder of the sequence and new frame towards the group.

Example 2 The watermarking algorithm proposed by Holliman, et al. [101] works as follows. An adaptive scheme is used to choose the blocks to be watermarked. Smooth and edge detection is done to determine the blocks that can withstand watermarking. Also, within each block, coefficients to be used to embed the bit are chosen pseudo-randomly based on properties of the block. The information is embedded by modifying these chosen coefficients based on the Zhao-Koch algorithm.

Modified to perform in a distributed environment, the algorithm operates

as follows. The source does smooth and edge detection and selects coefficients for each block. After beginning the watermark with a timestamp, the source sends the sequence of blocks to be watermarked and which coefficients are to be changed along with the stream towards the receivers. As each intermediary receives the stream, it uses the sequence to determine the next blocks to watermark and which coefficients in that block to use. The intermediary then adds its ID to the watermark and sends the rest of the block sequence and coefficient information along with the altered frame towards the group.

Though each frame contains the entire string of identifying information, it does not imply concentration of the watermark. It simply means that the entire piece of identifying information is embedded into each frame. The embedding algorithm is still based on a secure watermarking algorithm that effectively hides the embedded information inside of that frame data. Therefore, this results in no reduction in the level of security.

If there is not a need to safeguard single frames or very short clips, selective watermarking [198] can be used to increase the performance. This involves a tradeoff in the strength of the security because the length of video clip that is necessary to extract the watermark is increased. Instead of inserting the fingerprint in every frame, it can be inserted in every n-th frame. This translates into about a n-fold increase in performance with a tradeoff of n times the length of the clip that is necessary to extract the watermark. For example, with an MPEG stream, it is possible to fingerprint only the I frames. If the MPEG stream has the repeating IBBPBBPBB pattern, this will reduce the computational overhead by reducing the numbers of frames that are fingerprinted by 89%.

The information exchanged by the intermediaries is encrypted with an intermediary group key, I_k, while the content data is encrypted with some session key, G_k, as shown in Figure 7. In cases in which the intermediary does not already have

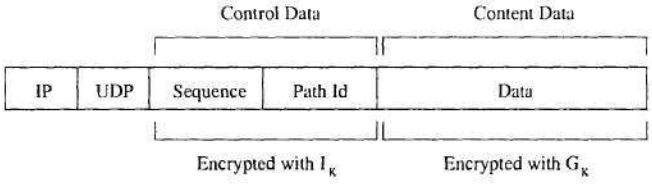


Figure 7: Packet transmitted between Intermediaries.

```
Repeat
        Extract Picture ?
        Repeat
                Extract Slice ?
                Repeat
                        Extract Macroblock ?
                        Repeat
                                Extract Luminance Block ?
                                Transform VLC To Integers
                                Embed Bit By Altering Coefficients For This Block
                                Convert Integers Back To VLC
                        Until (Finished Inserting Bits)
                        Or (No More Luminance Blocks In This Macroblock)
                Until (Finished Inserting Bits) Or (No More Macroblocks In This Slice)
        Until (Finished Inserting Bits) Or (No More Slices In This Picture)
Until (Finished Inserting Bits) Or (No More Pictures In This Sequence Layer)
```

Figure 8: The Bit-Embedding Algorithm at the Intermediary.

the compressed video data available, it will need to perform the necessary decapsulations, possibly including RTP [186], UDP, and IP, to extract the video data. Once the video data is available, the intermediary must perform the steps to locate the necessary blocks and embed the watermark. An example of this algorithm for MPEG video is shown in Figure 8.

3.3.3 Logging

To determine the domain of the receiver from retrieved watermarks, the log must have enough information so that it can determine which nodes were represented by that Path ID at that particular instant in time. Previously, there has not been much attention to the logging aspect of such a watermarking system. We have identified

it as a key requirement of the system and an important factor in the scalability of the system with regard to actually determining the party associated with a copy of content. While previous schemes for fingerprinting multicast video require extended periods of the fingerprinted video in order to extract enough information about the embedded fingerprint to determine the recipient, WHIM requires only one frame since the entire label is inserted in each frame. Thus, WHIM can safeguard each frame of a video. With some other schemes, if a user illegally redistributes a single image or a very short clip from a video, there is no way of determining the perpetrator. Also, our logging system requires only minimal information and uses a simple and straightforward algorithm to determine receivers.

Our logging system operates as follows. Each intermediary sends to the logging system, the Path ID that has accumulated in the packet (including its own ID). This Path ID also includes the timestamp inserted by the source. Depending on the overlay management used, the intermediary might also send its IP address or some other identifying information. This includes some authentication information and a timestamp so that the logging system is assured that the information is being received from a legitimate intermediary. This logging information is sent to the logger every time that the Path ID of the intermediary changes. Therefore updates are only sent when the overlay topology changes, not every time the underlying routing topology changes. When a watermark has been extracted and the receiver must be determined, only a simple table lookup algorithm is necessary to access this information from the log.

3.4 WHIM Last-Hop (WHIM-LH)

Whereas WHIM-BB marks the content to identify the last hop intermediary of a receiver, WHIM-LH allows a single intermediary to embed distinct User IDs for each of its children receivers. This section first explains the WHIM-LH architecture and the variations that are allowed by the different types of User IDs. Then, the

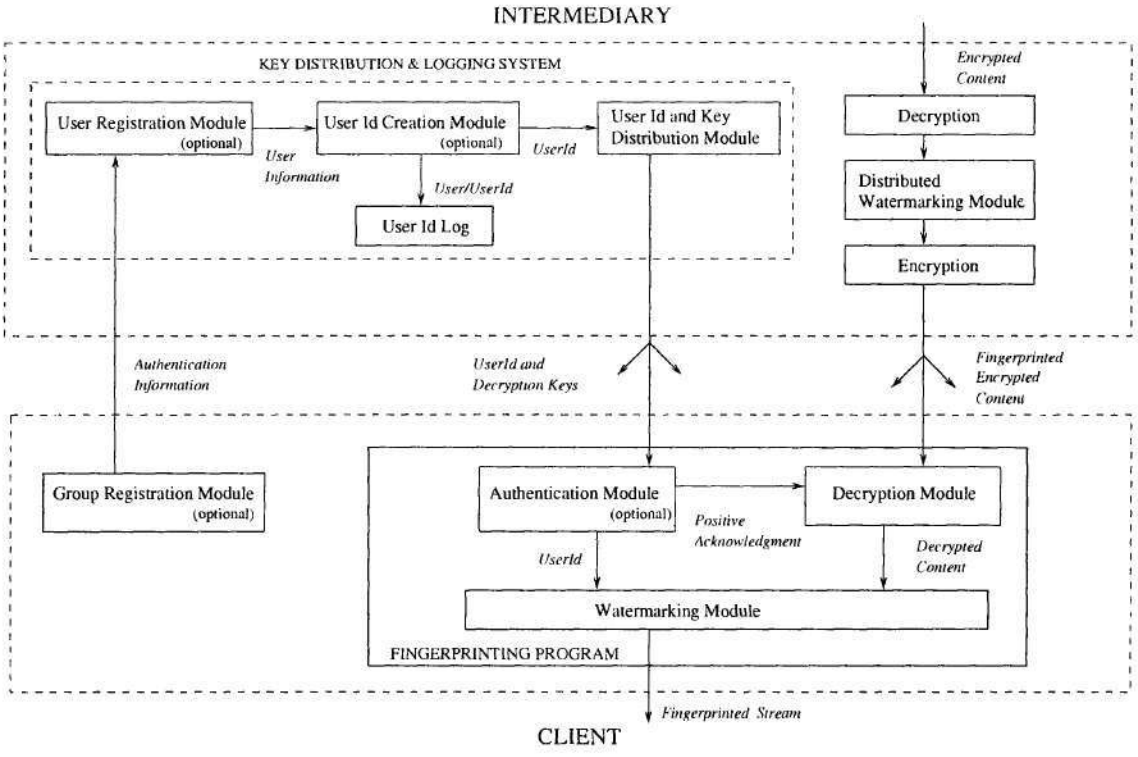


Figure 9: WHIM-LH Architecture

different methods that are available for choosing User IDs are explained.

3.4.1 Methods of Transporting the Video Data

The architecture we describe here maintains the efficiency of multicast routing while enforcing necessary security at endpoints, intermediaries and client. There is significant research in the area of video watermarking, so we provide a framework that allows any watermarking algorithm to be used to fingerprint multicast streams efficiently. We introduce a secure client-side fingerprint insertion program that contains a watermarking module that can be based on any chosen watermarking algorithm. Figure 9 shows the interaction between the modules of the architecture. The intermediary distributes the fingerprinting program with a built-in decryption key, which we subsequently denote as program[$K_{internal}$]. The client registers with the logging and key distribution system to join the group and receives decryption keys

and possibly a User Id. The client program then receives the stream encrypted with the session key, denoted as $\{\text{stream}\}K_{play}$, from the intermediary and securely adds the watermark before making the stream available to the user. The remainder of this subsection explains the variations of this architecture that depend on the type of User ID used.

The *Assigned User ID* scheme has the intermediary communicate with the group using the following steps:

Intermediary to Receivers:

Multicast: $\{\text{stream}\}K_{play}$

Multicast: $\text{program}[K_{internal}]$

Multicast: $\{\{K_{play}, \text{User ID1}\}K_{internal}\}K_{user1}$,
$\{\{K_{play}, \text{User ID2}\}K_{internal}\}K_{user2}$, ...
$\{\{K_{play}, \text{User IDn}\}K_{internal}\}K_{usern}$

Each User ID and key packet is encrypted with the user's public key or symmetric key shared by the logging system and the user, so the same level of security is achieved as if they were unicast. A significant portion of the traffic that is sent is the User ID information.

For applications that would benefit from the decrease in traffic that would result from not sending this information, we propose a method that allows the user to provide her own User ID information to the program. This *Local User ID* method only requires the intermediary to send the following messages to the group:

Intermediary to Receivers:

Multicast: $\{\texttt{stream}\}K_{play}$

Multicast: $\texttt{program}[K_{internal}]$

Multicast: $\{K_{play}\}K_{internal}$

The Authentication module authenticates the user and signals the decryption module. This approach is used when the logging system already has a mapping between the User ID and the actual receiver or can determine the receiver based on the User ID, such as when the User ID is derived from the public key as explained in the next subsection.

3.4.2 Methods of Choosing User ID

The User ID information that is embedded by the intermediaries as the watermark uniquely identifies each receiver. While previous literature simply refers to the User ID as some unique identifier, perhaps randomly assigned, we propose a new technique for creating User IDs. By using cryptographic means, we compose a User ID that is more closely bound to a user than a randomly assigned User ID. As shown in the previous subsection, this also allows a more efficient distribution method. Possible methods of forming a User ID include the following ways:

- **Assigned User ID:** This simple scheme involves each user registering with the source, authenticating with the source, and the source assigning some unique value as a User ID.

- **Public Key-based User ID:** This approach allows the User ID to be based on the public key of the receiver. This requires the user to have a public key certificate [124], a signed message from a trusted certification authority (CA) that specifies the user's name and the corresponding public key, such as a

X.509 certificate [102]. The fingerprinting program must be assured that the public key used is the one assigned to this user by the CA. We suggest two methods of doing this. The fingerprinting program requests the user's public key from the CA and then uses a nonce to confirm that the user knows the corresponding private key. The second method is that the user provides the program with the public key certificate and signs it with the private key. Thus, the program can verify the public/private key pair and that it was assigned by the CA.

3.4.3 Discussion

WHIM-LH provides a framework that allows proven watermarking algorithms to be used efficiently in a multicast environment. It allows efficient rekeying, introduces a new type of secure User ID construction, and has the smallest possible protection granularity. It also is capable of being used with selective watermarking [198] to increase its efficiency at the cost of an increase in protection granularity. Figure 10 shows the how the WHIM-LH architecture is combined with WHIM-BB.

We propose means of preventing the risk of the fingerprinting program being reverse engineered to reveal the decryption key or otherwise altered to disallow the desired results. There are a number of attacks that malicious users can perform against mobile agents including spying out code and data and manipulation of code and data [96]. Mobile Cryptography can be used to guard against these attacks [180]. This involves executing encrypted functions to guarantee code privacy and code integrity. Time Limited Black box Protection [97] can be used to protect the code and data of a mobile agent from being read or modified for at least some minimal time interval.

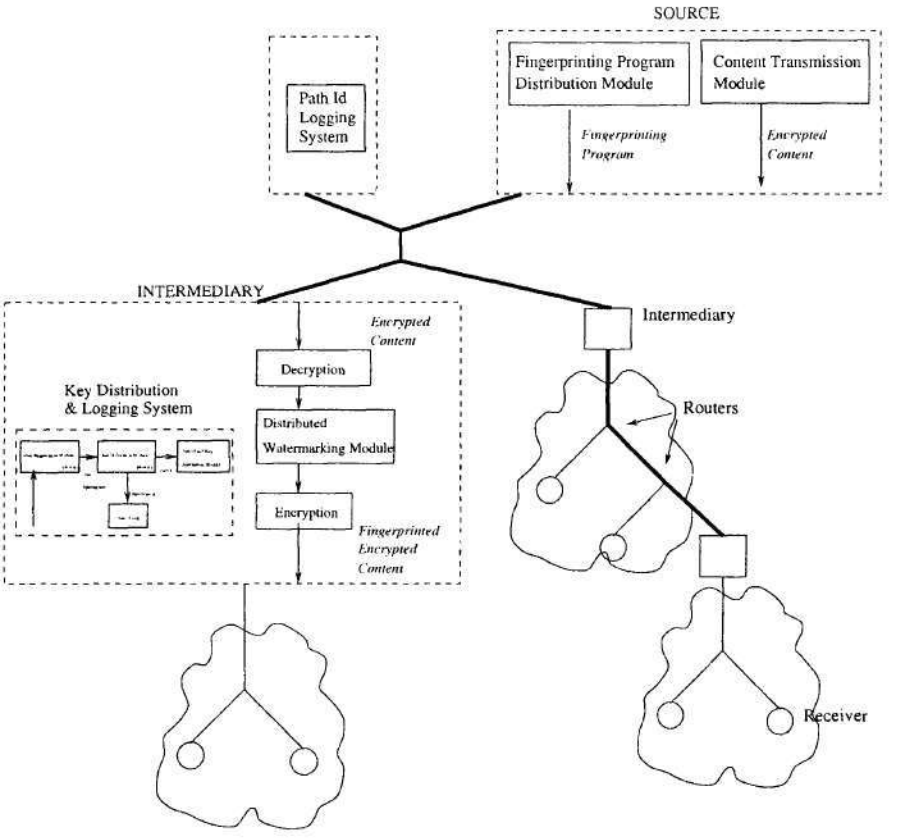


Figure 10: WHIM

3.5 Analysis

In this section, we examine the efficiency of WHIM in terms of data transmission and encryption overhead. We look at this relative to the performance of some of the other multicast watermarking schemes reviewed in the related work section; in particular Brown, et al, and Chu, et al. Figure 11 shows the definitions of variables used in this section.

In WHIM, the source transmits $s + p + cku$ bytes and encrypts $s + (n)(ku)$ bytes. The overhead of the Chu, et al. scheme involves the sender transmitting $nf[2(f) + 2(kf)]$ bytes, then the group leader transmits $nf[(n)(uid + bit + kf)]$ bytes. This system also has significant encryption overhead, $nf[2(f) + kf + msg]$ bytes for the sender and $nf[(bit+kf)(n)+msg]$ bytes for the leader. In the protocol of [32], the amount of transmitted data is increased substantially by the amount of

s	= stream
nf	= number of frames in the stream
f	= frame
p	= program
n	= number of group members
ku	= key/User ID message
cku	= combined key/User ID messages
k	= decryption key
uid	= User ID
bit	= signifies which stream the user receives
kf	= decryption key for a particular frame

Figure 11: Definition of Variables Used in Analysis

necessary redundant data. For a stream of size, s, the amount of data that is transmitted is at least ns, where $n > d$ and d is the depth of the multicast tree.

We seek to analyze the performance of these schemes with two different types of group behavior, theater style and dynamic. Theater style involves all of the group members arriving or joining the group and leaving the group at approximately the same time, as at a movie theater. This allows all of the set up overhead to be multicast to the entire group at once. Dynamic groups involve users joining and leaving the group at anytime throughout the session and may involve members leaving and re-joining. This also involves rekeying of the group.

To analyze the performance for theater style groups, we created multicast groups within transit-stub internetwork topologies using GT-ITM [213]. We performed the simulation with group sizes of 1,000, 5,000, 10,000, and 20,000 receivers. For each group size, the depth of the tree used in our data is based on the average depth of the 10 random shortest path multicast trees that were created. These calculations are based on the source multicasting a one hour session of MPEG-2 video at 4 $Mbps$ at a framerate of 30 fps. The size of the keys in our simulation are 128 bits for WHIM as well as for the scheme of Chu, et al. In our simulation of Chu, et al., the source is also the group leader. The size of the insertion program in WHIM was determined by adding the size of a common decryption program and the size of a watermarking program to be 1 MB; however, the total amount of data transmitted

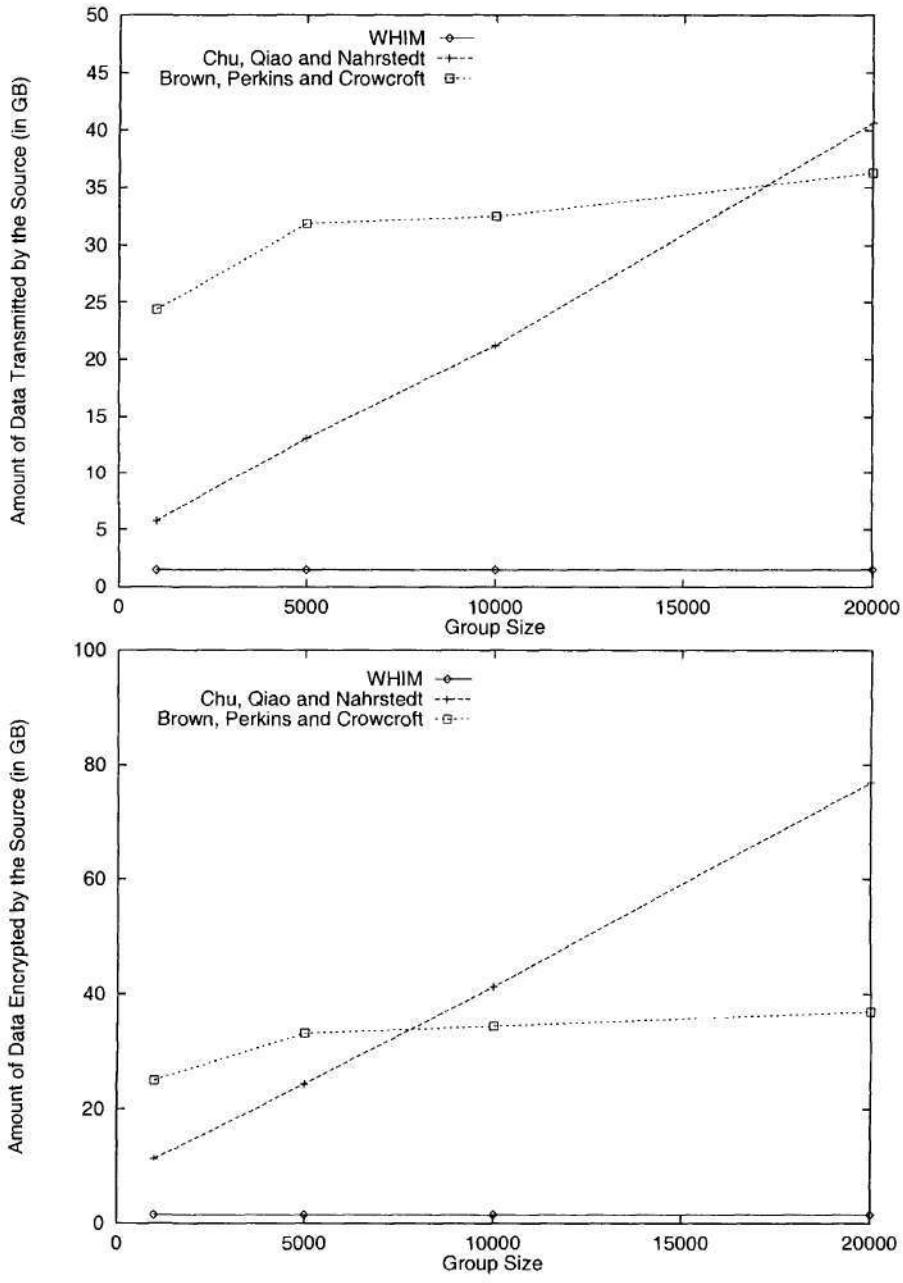


Figure 12: Multicast Fingerprinting Data Overhead at the Source

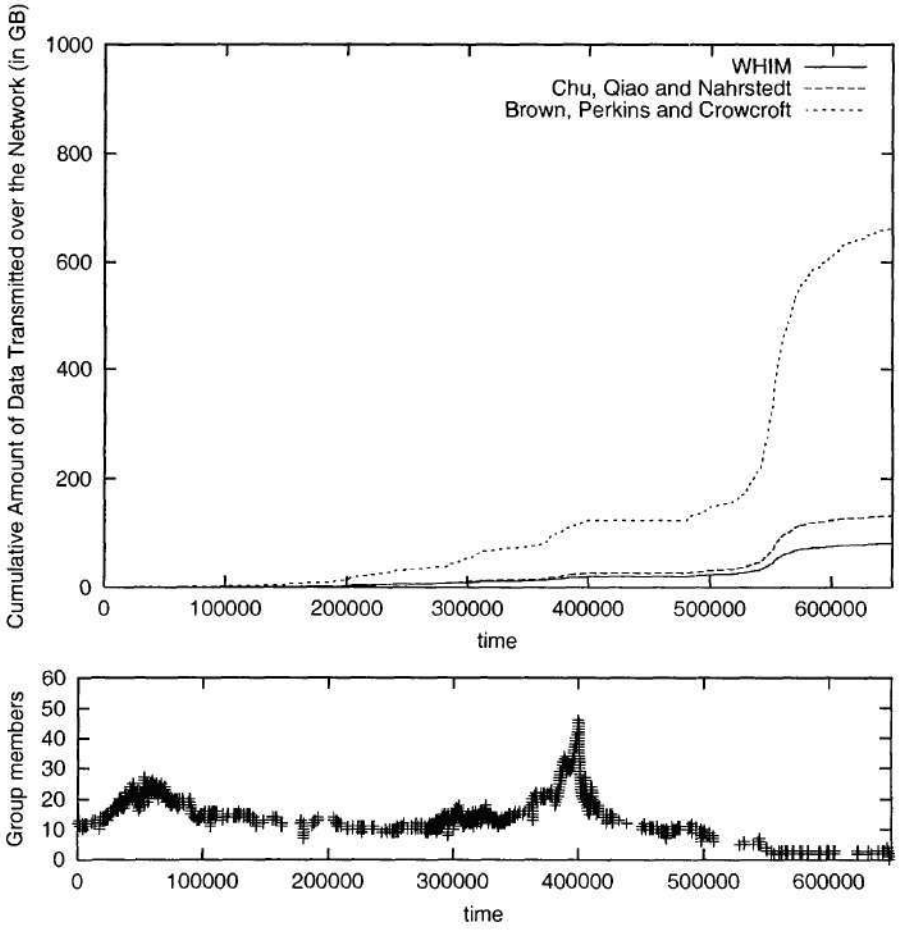


Figure 13: Multicast Fingerprinting Cumulative Data Overhead

and encrypted by the architecture is orders of magnitude above the size of the program so the accuracy of this value becomes insignificant. We compared the total amount of data transmitted and encrypted by the multicast source in WHIM with the schemes of Brown, et al. and Chu, et al. and the results are shown in Figure 12. The amount of data transmitted by the source in WHIM is about 1.5 GB for all group sizes. In the other schemes, the amount of data increases linearly as the size of the group increases. In Brown, Perkins, and Crowcroft's scheme, the amount of data is almost 25 GB for the smallest group size and continues to increase as the group size increases reaching over 35 GB for the 20,000-member-sized group. In Chu, Qiao, and Nahrstedt's scheme, the amount of data transmitted is a little over 5 GB for the 1,000 member group and increases sharply to about 40 GB for the 20,000 member group.

For dynamic groups, we used data collected by the Mlisten tool [14] over several days for the Mbone multicast of the Space Shuttle Mission STS-80 in November 1996. This session has a duration of 13 days and has over 1600 join requests. We used these traces to simulate the performance of the fingerprinting solutions. Figure 13 shows the cumulative amount of data transmitted over the network by these schemes and the number of receivers in the group over time. The cumulative amount of data transmitted by WHIM is about 80 GB while the total for Chu, Qiao, and Nahrstedt's scheme is about 120 GB. The total amount of data for Brown, Perkins, and Crowcroft's scheme is about 650 GB. There is a sharp increase in the amount of data transmitted using Brown, Perkins, and Crowcroft's scheme between 500,000 and 600,000 seconds. This is because a group member joined that was a considerable number of hops from the source and the number of copies of the content that is transmitted by the scheme is based on the depth of the tree.

One factor that allows WHIM to provide greater scalability than previous solutions is the intermediaries that are used to provide security functionality. In our analysis, we do not consider the cost of deploying these intermediaries. This cost includes not only the monetary cost but also may possibly include delay. We do

not provide quantitative results that show the incurred delay; however, we implemented a prototype of the system to examine this and other issues. As we discuss in subsequent sections, the delay incurred was reasonable. We believe that with appropriate buffering, the delay is negligible. Also, many systems utilize content distribution networks or infrstructures that have introduced intemediaries to perform some functionality. Ideally, WHIM's intermediaries will leverage an architecuture that is already in place.

3.6 Implementation

We developed a prototype implementation of WHIM-BB. The architecture involves having multicast video delivered through a hierarchy of intermediaries and marked uniquely for each recipient. Our prototype used H.261 [4] video and RTP [186]. The rtpplay [1] tool was used as the video source to multicast the original video. Vic was used as the client to receive the multicast stream and play the video [134].

The WHIM intermediaries utilized rtpgw [18], an application level RTP gateway, to provide basic video proxying. The prototype implemented our distributed version of the watermarking algorithm presented by Dittmann, et al. [61] to mark the content.

We developed a C++ class that performs watermarking as a filter on the video content. This allows our functionality to be used within other frameworks such as Open Mash [133]. We implemented a base class called WM_Filter and two more specific classes called InsertWM_Filter and ExtractWM_Filter.

We changed the rtpgw's H261Transcoder class to pass the video through the watermarking filter. Within the H261Transcoder class, wm_filter() is called from the recv() function before the video is re-encoded for retransmission.

The following code is added to the recv() function to access the watermark filter class:

[1] http://www.cs.columbia.edu/IRT/software/rtptools/

```
InsertWM_Filter *iwmf;
iwmf = new InsertWM_Filter
            (outw_, outh_, decimation_, wmspot_, wmid_);
iwmf->wm_filter(decoder_->frame());
delete iwmf;
```

wm_filter() steps through the group of blocks in a frame and for each group of block, traverses each of the macroblocks. For each macroblock, wm_filter_mb() is called. wm_filter_mb calls wm_filter_blk() for each block of the four luminance blocks in the macroblock. wm_filter_blk() implements Dittmann's watermarking scheme. It chooses the three coefficients in the block to be marked. It determines the bit that must be encoded in this block by shifting the string to be embedded by the number of bits that it has inserted in previous blocks. The new coefficients are determined based on a Zhao-Koch type algorithm. The coefficients are then re-quantitized and the new values are placed in the block. wm_filter() returns after a given frame is successfully marked and the watermarked frame is then passed to the encoder to be forwarded.

We also implemented a watermark extraction tool that can be used to identify embedded information within a recovered video copy. It was also useful for debugging. It is called from the recv() function as follows:

```
ExtractWM_Filter *ewmf;
ewmf = new ExtractWM_Filter
            (outw_, outh_, decimation_, wmpathlength_);
ewmf->wm_filter(decoder_->frame());
delete ewmf;
```

We were able to multicast video to a number of receivers and have the content watermarked uniquely based on the receivers location in the network. Notable observations were that the watermark insertion did not require buffering and did not introduce any noticable delay.

3.7 Conclusions

There has been a significant amount of work geared toward developing algorithms to securely embed watermarks into multimedia content. The work presented in this paper complements these efforts by providing an architecture that allows these algorithms to be used in multicast multimedia. We have presented two architectures, WHIM-Backbone, a hierarchy of intermediaries that provides an efficient distribution architecture that fingerprints the streaming content, and WHIM-Last Hop, a secure client/server protocol that fingerprints and distributes content between a single entity and a group of receivers, which form WHIM. Our analysis shows the efficiency gains of WHIM over previous solutions.

Transmission of Video	Unicast	Multicast			
Marking Location	Source	Source	Client (WHIM-LH)	Intermediary (WHIM-BB)	Intermediary and Client (WHIM)
Trust	High	High	Medium	High	High
Scalability	Low	High	Medium	High	High
Resolution	High	Low	High	Medium	High

Table 1: Comparison of trust, scalability, and resolution provided by different methods of fingerprinting content to a group

Table 1 compares the trust, scalability, and resolution achieved by solutions based on the type of transmission of the video and the marking location of the data. The first column shows the simple case of marking at the source and unicasting. This achieves high trust and resolution but low scalability. The next column shows

multicast video that is marked at the source. This results in high trust and scalability but low resolution. The third column shows WHIM-LH which multicasts the video and marks at the client. This achieves medium trust and scalability and high resolution. The fourth column shows WHIM-BB which multicasts the video and marks at the intermediaries. This achieves high trust and scalability and medium resolution. The last column shows WHIM which combines WHIM-LH and WHIM-BB to achieve the scalability of multicast with the trust and resolution of a unicast approach.

In addition to the architecture presented in this chapter, the idea of identifying a user by his position in the network can be carried over into other applications to offer increased security and the use of a trusted hierarchy to provide scalable security functionality can be used in other areas including group key management, firewalls, and defending denial-of-service attacks.

Chapter 4

Group Access Control for Content Distribution Tree Protection

The Internet is increasingly being viewed as a medium providing not just connectivity but also services. This is due to the increase in mechanisms within the network to support networked services. An important need is mechanisms to aid in the scalability of networked services. Two such mechanisms have received considerable attention over the years—multicasting and anycasting. Multicast communication provides an efficient transport mechanism for one-to-many and many-to-many communication [56]. Anycast provides a means for a host to send a request to one address and have it serviced by one of many servers servicing that address [155]. This aids service discovery. There has been significant advances in maturing both of the paradigms. There are distinct and significant security problems in both the multicast and anycast models including denial-of-service, theft-of-service, eavesdropping, and masquerading.

We first explain the multicast problem. There are a number of available multicast routing protocols that provide the efficient transport mechanisms of multicast by routing packets with one group destination address to multiple recipients. The routing protocols must be aware of group members in the network in order to deliver packets to them. The mechanism provided for doing this is the Internet Group Membership Protocol (IGMP) [33]. A host uses this protocol to notify the routing system that it should deliver packets for a particular multicast group to this host. In the current model, any host can use IGMP to become a member of any IP multicast

group causing eavesdropping or theft of service. The common method used to protect the information is to encrypt the multicast data and provide decryption keys only to authorized members. In some cases, encrypted communication is not possible for any number of reasons including legal issues or technical reasons. Even if encryption is used, there are still risks involved with unauthorized users receiving encrypted data such as traffic analysis and possibly cryptanalysis. The current model is also vulnerable to a denial-of-service attack in which malicious hosts join a number of multicast groups. This causes potentially large amounts of data to be forwarded to it utilizing network resources.

The anycast paradigm has a different security problem that can be equally as damaging. Anycast allows multiple servers to provide a service at a single address called the *anycast address.* This is accomplished by each of these *anycast servers* letting the routing protocol know that it is listening to the anycast address. Then when a host wishes to contact a server providing that particular service, it simply sends a request to the anycast address. The routing system knows which servers said that they are providing that service so it chooses one of those servers and forwards the request to it. Besides the basic IP model of anycast [155], global IP anycast [116] and application-layer anycast [212] have been proposed. The problem in each of these models is that any system can pretend it is providing a service by telling the routing system that it is listening to that anycast address. This problem has two potential outcomes: denial-of-service or masquerading. The fake server can simply attract requests and ignore them causing a denial-of-service attack. Or, the fake server can actually respond to the request with false information which can lead to a number of additional problems.

Solving the problems described in the multicast model requires controlling the ability of hosts to join the multicast group. We call this *multicast group access control.* The need for a solution to these problems is well known. Gong and Shacham first stated the need [81], and the need has been restated by Ballardie and Crowcroft [22] Shields and Garcia-Luna-Aceves [188], and Hardjono and Cain [87].

The term *secure IGMP* has been used to refer to the protocol that would provide the solution. Solving the problems described in the anycast model requires controlling the ability of a host to advertise itself for the anycast address. This requires controlling membership to the anycast server group. We call this *anycast server group access control.* The need for a solution to the anycast problems is also well known. Partridge, et al. [155], and Katabi and Wroclawski [116] state the need for a solution. Previously, the multicast and anycast problems were viewed as separate problems requiring separate solutions. In reality, the problems in multicast and anycast can be generalized the same group access control problem.

In this work, we propose Gothic, a comprehensive architecture for providing group access control. The design goals are to maintain security while providing a scalable system that involves low computation overhead at the routers, low message overhead, and low support infrastructure requirements. The architecture combines some novel techniques with some known systems security concepts. We evaluated our system relative to two previously proposed systems and find that Gothic maintains or increases the level of security relative to previous work while increasing scalability. We also propose a group policy management system that allows the group owner to be authenticated before being allowed to specify the group access rights. This system can be applied to other group policy work. Finally, we propose and evaluated *group access control aware group key management* (GACA-GKM), which is a protocol that leverages trust built into an group access control system to reduce the requirements of group key management (GKM) and obtain substantial overhead reductions.

For each of the multicast and anycast problems, there are a number of potential environments composed of different Internet Protocol (IP) versions, different routing protocols, different address allocation schemes, and different inter-domain requirements. We call the particular combination of these the *implementation environment.* The proposed architecture is relevant to many flavors of multicast and anycast on the Internet such as Global IP-Anycast (GIA) [116], application-layer anycast [212],

source-specific multicast [99], and application-layer multicast [49, 106]. This chapter freely uses standard terminology from the network and systems security literature without further definition [126].

This chapter is organized as follows. Section 4.1 gives an overview of the Gothic architecture and discusses the two subsystems: the group member authorization system and the group policy management system. Section 4.2 describes the authorization system. Section 4.3 discusses the group policy management system specifying it for multicast and for anycast. Section 4.4 discusses group access control aware group key management and Gothic's interaction with the routing system. Section 4.5 presents an evaluation and simulation results of the architecture and a comparison with previously proposed solutions. We also provide simulation results comparing traditional GKM to our group access control aware GKM technique. Finally, section 4.6 presents conclusions and discusses possible future work.

4.1 Overview of Gothic

Several functions are necessary to provide controlled access to a group including the following:

1) Group policy specification functions: These involve a host requesting to specify a *group policy*, authenticating the host, and verifying that the host is the *group owner*. The group policy is an access control policy that specifies which hosts have access rights to become members among other characteristics. The group owner is the entity that has been assigned ownership of the multicast group and is therefore authorized to specify the group policy.

2) Access request functions: These involve a host notifying the system that it wishes to become a member of a certain group.

3) Access control functions: These involve receiving a host's request, authenticating the host and performing *authorization*. Authorization requires checking the group policy to determine if that host has the access rights to become a member of the requested group.

Gothic controls the group of hosts that can receive data destined to a specific multicast group address; however, Gothic does not control multicast sources. Controlling which hosts can send data to a group is a separate problem. Some solutions have been proposed for multicast sender access control [22]. Sender access control is becoming less of a problem with recent multicast schemes such as source-specific multicast(SSM) [99] that inherently provide sender access control—though they rely on traits of the reverse path forwarding and the security is a side effect.

Gothic is composed of two systems: the *group policy management system* and the *group member authorization system*. Figure 14 shows Gothic and its two subsystems. The group policy management system performs group policy specification functions. The group member authorization system involves access request functions and access control functions. Gothic also interacts with the routing system and any group key management system that may be in place.

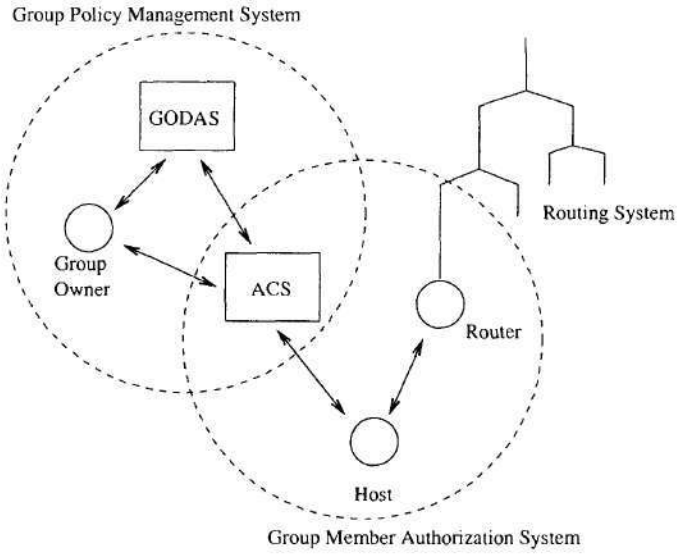


Figure 14: Gothic Architecture

Group Policy Management System

The group policy management system involves a group owner providing the list of authorized members and possibly other security policy for the group to the access control server (ACS). Previous work has presented requirements for specifying group policy [136]. The task of a host specifying the policy to the system is understood. The problem that remains unanswered is how the system verifies that the host is the group owner. We propose two solutions for a *group owner determination and authentication system* (GODAS), described subsequently.

Group Member Authorization System

The group member authorization system provides the core functionality of Gothic by controlling access to the group. Previous proposals for authorization systems that handle multicast were proposed by Ballardie and Crowcroft [22] and Hardjono and Cain [87]. The design goals of our authorization system are to maintain security and to achieve scalability. The main scalability objective is to reduce the computational load on network routers and the second objective is to reduce the message overhead. We provide evaluation results that show our system improves scalability relative to

network and processing overhead while maintaining or increasing the level of security of previous systems.

A Gothic Scenario

Figure 14 provides an illustrative overview of the operation of Gothic.

First, the group owner contacts the ACS; the ACS then performs authentication and authorization functions; and the group owner states the group policy. Second, hosts wishing to join the group request access and the system performs access control functions allowing authorized members to join. Finally, use of the group by sources and receivers may begin. In anycast routing, initiation means that the anycast address may be distributed and requests sent to it. In multicast, initiation means that the source may begin transmitting data even if receivers have yet to join the group. We provide this scenario to show the order of operations: the group policy management system's operations take place before the operations of the group member authorization system. However, the presentation of the paper does not follow that order. We will first discuss the core of the architecture, the group member authorization system, and then explain the supporting component, the group policy management system.

4.2 Group Member Authorization System

This section describes the group member authorization system that allows authorization to be performed before the host is allowed to become a member of the group. The first subsection presents the base protocol. The second subsection discusses the operation of the system including reauthorizations and revocations. The third subsection elaborates on the interesting design features of the system and how they relate to prior work.

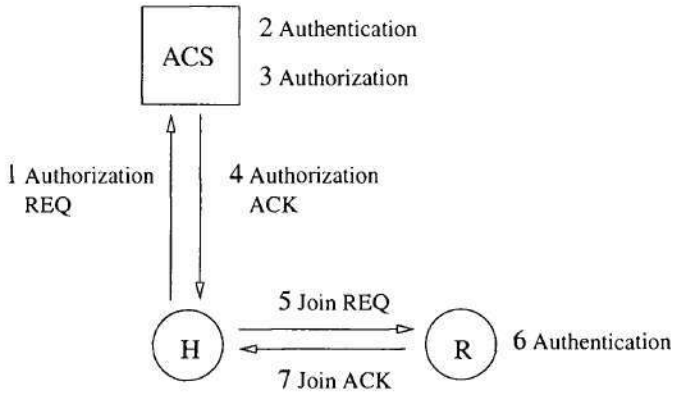


Figure 15: Authorization System

4.2.1 Authorization Protocol

The Gothic authorization system involves a host H, a router R, and the access control server ACS. In reality, the ACS can be a single server or a group of distributed servers. Since the authorization protocol takes place between a host and a single ACS, our discussion here only involves a single access control server. We assume the presence of a public-key infrastructure (PKI) [102]. Hosts and the ACS possess public-key pairs and certificates; we do not require that the routers possess key pairs or certificates. Also, for environments without a public-key infrastructure, we describe how the system can operate without host key pairs and certificates. In general, the host and the access control server each have public keys, K_{+H} and K_{+ACS} respectively and the corresponding private key, K_{-H} and K_{-ACS}. The certificate issued by a trusted authority containing a public-key K_{+X} is denoted by $CERT_{K_{+X}}$ These are used to digitally sign messages and verify those signatures. Digitally signed messages are shown in brackets with the key used to sign it as a subscript $[message]_{K_{-X}}$,

Figure 15 diagrams the operation of the base protocol. The protocol begins with the host wishing to join a group sending an authorization request to the access control server. The *authorizationrequest* (AR) contains the group ID (GID) of the

group the host wishes to join, the host's public key certificate, and is signed with the host's private key. Where key pairs are not available, an alternative authentication method can be used such as a password. Also, the AR can be coupled with the receiver obtaining the group key from the group key management system. In this case, the host is authenticated and authorized once and receives the group key as well as the capability.

1. H $\rightarrow$ ACS:

$$AR = [GID, CERT_{K_{+H}}]_{K_{-H}}$$

The ACS authenticates the host and checks the group policy to determine if the host has access rights to join the requested group. The ACS returns an authorization acknowledgment (AA) specifying a successful or unsuccessful authorization. If successful, the AA is a capability that includes the host's IP address IP_H, the host's distinguished name DN_H, the multicast group address, the expire time, the public-key certificate of the ACS, and the digital signature of the ACS. The use of timestamps to indicate the expire time of the capability requires clocks that are at least loosely synchronized.

2. ACS $\rightarrow$ H:

$$AA = CAP =$$
$$[IP_H, DN_H, GID, T_{exp}]_{K_{-ACS}}, CERT_{K_{+ACS}}$$

The receiver's IP address serves as an identifier and provides propagation control. The receiver sends a join request (JR) containing the capability(CAP) to the router. This join request is formed by including the capability in the IGMP Membership Report message [33] or the MLD Multicast Listener Report message [82, 201].

3. H $\rightarrow$ R:

$$JR = CAP$$

The router checks the validity of the capability. This includes verifying the ACS's signature, checking the expiration time, and verifying that this capability came from the receiver it was assigned to. This can be done by authenticating the host and checking the authenticated identity against the identity in the capability. Or, the router can simply lookup the IP address in its routing table to confirm that the request arrived from the interface leading to that address. This eliminates the need to authenticate the host. Section 4.4 discusses how Gothic can be extended to integrate with the security of the routing system to provide controlled propagation. After verifications, the router sends the host a join acknowledgment (JA) stating a successful or unsuccessful join.

4.R $\rightarrow$ H:

$JA = Status$

To allow for groups with no access control, if a router receives a join request that does not include a capability, then the router queries the ACS to verify that the requested group is unrestricted.

4.2.2 Reauthorizations and Revocations in the Protocol

This section discusses the reauthorizations and revocations that are part of the operation of the system. We explain how we achieve efficient revocations while maintaining the desired level of security. We also describe a method for multicast groups to achieve greater efficiency by leveraging the GKM system.

Our base protocol uses time-limited capabilities to provide revocation. Requiring members to refresh their membership state coincides well with the soft-state of the IGMP group membership reports and of the routing protocols. However, refreshing authorization state is a heavyweight operation compared to a routing or IGMP update. Therefore, for efficiency one might consider extending the lifetime of the capabilities. This reduces the load by reducing the frequency and number

of reauthorizations. However, this weakens the security by increasing the revocation window. That is, if a member is ejected from the group, there will be some vulnerable time where the ex-member still has access because the capability has not expired and he has not been required to reauthorize. Therefore there exists a tradeoff between reauthorization overhead and security. By changing the capability lifetime, the system can be tuned to the desired tradeoff point.

The ideal system would allow a small revocation window and low reauthorization overhead. We propose a method of obtaining this for multicast groups. Our goal is to provide a more lightweight reauthorization phase: instead of reauthorizing with the ACS to obtain a capability, the host uses the group (decryption) key as the authenticator. Since only authorized group members possess the group key, knowledge of it successfully authorizes a host as a member of the group. This requires the router to possess the group key. In many cases, this is straightforward because the router is part of the key distribution path and simply must store the key as well as forwarding it. Since the GKM maintains current authorization state, the authorization system piggyback on that functionality by using the group key as the authenticator. The authorization system can use the group key not only for reauthorizations, but for the initial authorization as well.

4.2.3 Discussion

This section discusses some of the interesting design features of Gothic's authorization system and mentions related designs in other security systems.

The authorization system is designed to gain efficiency by integrating security functions with the current network system that is in place rather than adding bulky components. Among the interesting design features of Gothic's authorization system are:

- There is no need for propagation control components in the system because the

design inherently provides propagation control by using identity-based capabilities. Gong [80] describes an identity-based capability system. Neumann [150] describes a similar concept called the delegated proxy.

- There is no need for additional components to provide revocation, because we use time limited capabilities to provide implicit revocation [1]. Explicit revocation is heavyweight and is usually provided by the use of certificate revocation lists (CRLs) or by supplementing capabilities with access control lists (ACLs) that are checked upon access. The normal problem with implicit revocation is that large time limits weaken the security and small time limits require an increased number of heavyweight reauthorizations.

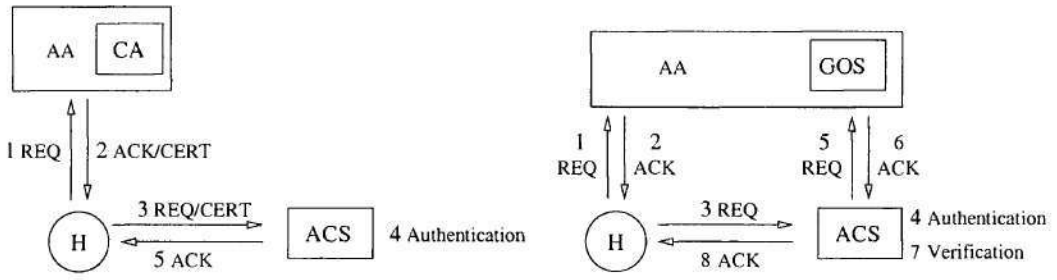
- We propose leveraging the GKM system to reduce the overhead of reauthorizations; thus allowing the strong security of short lived capabilities without the overhead normally involved.

- The design is simplified by not including complex properties of access control models such as lattice security and the *-property [126, 115] that are unnecessary for multicast and anycast groups.

4.3 Group Policy Management System

This section describes the group policy management system. This system involves a group owner providing the list of authorized members and possibly other security policy for the group to the ACS. McDaniel, et al. [136] proposed related work that presented requirements for specifying group policy for the key management and data handling building blocks of the Internet Research Task Force's secure multicast framework [88]. The task of a host specifying the policy to the system is understood. The problem that remains unanswered is how the system verifies that

[1] There has been previous work in efficient revocation schemes including re-acquisitions in DNSSec [76] and the re-confirmation TTL used in the Simple Distributed Security Infrastructure (SDSI) [176].



(a) Group Owner Certificates

(b) Group Ownership Service

Figure 16: Group Owner Determination and Authentication System

the host is the group owner. We propose two solutions that provide *group owner determination and authentication.* The first subsection describes the two solutions. The second and third subsections discuss the use of the group owner determination and authentication system in multicast and anycast environments.

4.3.1 Group Owner Determination and Authentication Systems

The *group owner* is the host that has been allocated control of or use of a particular group address. The purpose of the group owner determination and authentication system is to allow the ACS to determine if the host that attempts to provide the group policy is the group owner. We discuss two different systems that provide this functionality.

The first solution is the use of *group-owner certificates* as shown in Figure 16(a). These are similar to traditional digital certificates in that the certificate verifies the

identity of the entity that possesses the corresponding private key. With group-owner certificates, the identity in the certificate is the group address [2]. The group-owner certificate can be issued by a local Certificate Authority (CA) that is associated with the entity that allocates group addresses in each domain [3]. We specify the association between the certificate authority and the address allocator for different environments in the next two subsections. The group owner presents the group owner certificate to the ACS along with the request to specify the group policy. This allows the ACS to verify that the host is indeed the group owner.

The second solution is the deployment of a *group-ownership service* as shown in Figure 16(b). Rather than the host providing proof-of-ownership to the ACS, the ACS queries the address allocator. It accepts queries specifying a particular group address and responds with the identity of the host that owns the group. The group ownership service is deployed on a system that is associated with the entity that allocates group addresses in each domain. It is deployed at a common address and port number to allow it to be located. Upon receiving a request from a host to specify the group policy, the ACS authenticates the host then queries the group ownership service and verifies that the reply matches the identity of the requesting host. We specify how the group ownership service can be deployed in certain multicast and anycast environments in the next two subsections.

4.3.2 Group Owner Determination and Authentication in Multicast environments

In this section, we discuss how group owner determination and authentication can take place for the different multicast address allocation schemes.

- *Multicast Address Allocation Architecture (MAAA):* MAAA [196] specifies

[2]For example, this can be accomplished with X.509 v3 certificates by specifying the group address in the Subject Alternative Name Extension. [102]

[3]To ease PKI requirements, the local certificate authority can possess a certificate issued by a globally trusted certificate authority.

inter-domain and intra-domain address allocation methods . A well-known instance of this architecture includes MASC [167], AAP [83], and MADCAP [86]. Thus the owner of a multicast group is the host that is allocated that address by MADCAP. The group ownership service can be added to the MADCAP protocol. Or the MADCAP protocol can provide group owner certificates to the host that is allocated the address.

- *Source-Specific Multicast (SSM):* In SSM [99] , a multicast group is specified by a tuple (S, G) where S is the IP address of the source and G is a SSM destination address from the assigned 232/8 range. This provides a straightforward mapping between the group owner and the group address since a host S owns all multicast groups $(S, *)$. When a host wishes to specify the policy for any group (S, G), the access control server simply authenticates the host to verify that it is indeed host S.

- *GLOP:* GLOP [139, 138] provides a method for statically assigning multicast group addresses to Autonomous Systems (AS). The identity of the autonomous system is encoded into the group address. Within each AS, different allocation schemes can be used such as static allocation, MADCAP, or SAP. If the internal allocation scheme is also static allocation, then the AS can provide a group ownership service or provide a group owner certificate authority.

- *Session Announcement Protocol (SAP)/ Session Description Protocol (SDP):* SAP [85] and SDP [84] provide mechanisms to describe a session and to announce that session. The group owner is the host that advertises a session at a particular group address. A malicious user can disobey the protocol and advertise sessions that are not his, thereby making itself the group owner and obtaining the ability to specify access control policy. Therefore, for the highest level of security, one of the previously described address allocation schemes should be used.

4.3.3 Group Owner Determination and Authentication in Anycast environments

In this section, we discuss how group owner determination and authentication can take place with the different anycast schemes.

- *IP Anycast:* IP anycast includes both IPv4 and IPv6 environments. IPv4 anycast uses a separate class of addresses for anycast addresses [155]. For IPv4, group owner certificates can distributed to the host that is assigned the address. IPv6 anycast addresses are indistinguishable from unicast addresses and consist of a set of reserved addresses within each subnet prefix [94, 107]. For IPv6, group owner certificates can be used or a group ownership service can be used since the address is related to the unicast address and the domain.

- *Global IP Anycast (GIA):* In GIA [116], anycast addresses consist of an anycast indicator, the home domains unicast prefix, and a group ID. Domains are allocated anycast addresses according to their allocated unicast address space. Thus, the domain owns its set of anycast addresses and may give control to some host. Each domain can provide a group ownership service or use group owner certificates.

- *Application-Layer Anycast:* In Zegura's et al. application-layer anycast scheme [212], anycast services are referred to by anycast domain names (ADNs) that contain the domain of the authoritative resolver for the ADN. A group ownership service can be deployed at the authoritative resolver. Or, group owner certificates can be used and the authoritative resolver can be the local certificate authority.

4.4 Group Access Control Aware GKM

There has been a significant amount of work in group key management(GKM); see for example [38, 35, 209, 142]. Most of this work has been in creating efficient algorithms and systems for GKM for dynamic groups. These GKM solutions were designed around the assumption of an open Internet multicast group where all hosts have access to the multicast tree. This is due to the fact that IGMP allows any host to join a multicast group and receive the data being sent to that group.

With multicast receiver access control deployed, this assumption no longer holds. Multicast receiver access control provides a means to restrict access to the multicast tree to authorized users. There have been proposals for systems providing multicast receiver access control in the research community as well as in the IETF. Since group access control changes the assumptions of GKM designers, the requirements and approach of GKM should be reconsidered. The goal of *group key encryption* is to prevent unauthorized receivers from obtaining the content. *Group key management* focuses on the dynamic group problem. That is, when a member joins or leaves, the group key must be changed so the new member cannot decrypt past content or so the former member can not decrypt future content.

In traditional GKM, the key is changed upon a join because it is assumed that the new member could easily used IGMP to receive the encrypted content from before it was a member. Thus, giving the new member the old group key will allow it to decrypt the content from before it was a member. Now, with group access control in place, the host can use IGMP to receive the encrypted content before it is a member. If the host does not have the earlier content, then there is no need to rekey the group. There are similar implications for a member leave. In traditional key management, the key is changed upon a leave because it is assumed that the leaving member can use IGMP to easily continue to receive the encrypted content and not changing the key will allow it to decrypt it. With group access control in place, the leaving member can not use IGMP to access the distribution tree to

obtain the encrypted content. If the host can not continue to receive the content, then there is no need to rekey the group.

With these new assumptions, we propose a new GKM technique that leverages the existence of multicast receiver access control. We show that even with the existence of GACA, the need to rekey is not abolished. There are certain issues that must be considered such as the lack of access control beyond the subnet or shared link level. Also, we discuss the risk associated with eavesdroppers. We will show that the system is robust against local eavesdropping attacks, but not against some more involved eavesdropping-based attacks.

A GACA-GKM system requires the group key controller to have knowledge of the multicast topology and the placement of members in the tree. This information is not available in readily available in existing GKM systems. We will explore three methods of providing this functionality. The methods are:

- Traceroute-type Approach
- Topology Inference-based Approach
- Enhanced IGMP-based Approach

4.4.1 GACA-GKM Technique

In the first subsection, we provide some definitions by explaining how Gothic integrates with the multicast routing system. The second subsection discusses the GACA-GKM technique.

4.4.1.1 Group Access Control and the Routing System

For multicast group access control, after the router accepts the host, the router must forward the join request according to the multicast routing protocol. The routing protocol may require reauthorizations or provide its own message authentication methods. There have been a number of studies that propose secure multicast routing

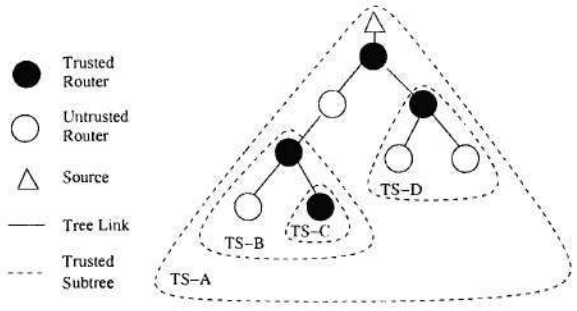


Figure 17: Gothic Trusted Routers form Trusted Subtrees

protocols [188, 205, 144]. Some routing protocols do not assume trust of the entire routing system but only of some subset that are considered trusted routers. In the context of Gothic, a *trusted router* is defined as a router that correctly authorizes all join requests according to the protocol. An *untrusted router* is a router that may accept unauthorized join requests or forward fabricated or unauthorized join requests.

When a host sends a join request to its upstream router, the router forwards the request containing the capability according to the routing protocol. Any trusted router on the path will perform access control by verifying the capability. Thus, an untrusted router may accept and forward requests from unauthorized hosts but trusted routers on the path will provide access control. The *scope of trust* extends from the source to the multicast tree and is bordered by trusted routers. A *trusted subtree* is a subtree of the multicast tree rooted at a trusted router. A trusted subtree can exist within another trusted subtree. A host is a member of the trusted subtree of its first upstream trusted router. Figure 17 shows how trusted subtrees are formed.

4.4.1.2 Details of the Rekey Conditions

With the introduction of group access control, the goals and requirements of group key management must be reconsidered. The goal of *group key encryption* is to

prevent unauthorized receivers from obtaining the content. *Group key management (GKM)* focuses on the dynamic group problem. That is, when a member joins or leaves, the group key must be changed so the new member cannot decrypt past content or so the former member cannot decrypt future content. There has been a significant amount of work in group key management; see for example [38, 35, 209, 142]. These GKM solutions were designed around the assumption of an open Internet multicast group where all hosts had access to the multicast tree. With group access control in place, this assumption no longer holds. We propose a GACA-GKM technique that leverages the inclusion of a group access control system.

We show how we are able to relax the requirements of GKM. In traditional GKM, the key is changed upon a join because it is assumed that the new member could have received the encrypted content from before it was a member. Thus, giving the new member the old group key will allow it to decrypt the content from before it was a member. Now, with group access control in place, the host can not receive the encrypted content before it is a member. If the host does not have the earlier content, there is no need to rekey the group. There are similar implications for a member leave. In traditional key management, the key is changed upon a leave because it is assumed that the leaving member can continue to receive the encrypted content and not changing the key will allow it to decrypt it. With group access control in place, the leaving member will not be able to access the distribution tree to obtain the encrypted content. So, there is no need to rekey the group.

This simple example shows the significant impact of introducing group access control. However, we are not able to achieve such gains for every member join and leave. For example, if a new member, host A, is on a shared link with current group member, host B, then we must rekey when host A joins since she had access to the distribution tree before she became a member. Similarly, if leaving member, host C, is on a shared link with current member host D, then we must rekey when host C leaves because she will have access to the distribution tree after she is no longer a member. In reality, these cases include not only if the two users are on a shared

link but also if they are in the same trusted subtree. The GACA-GKM technique is stated as follows:

KTS_G is the set of known trusted subtrees for a multicast session G.
TS_h is the trusted subtree of host h.
n_{TS_h} is the current number of members in trusted subtree TS_h.

For a join: *If* (TS_h in KTS_G)
If ($n_{TS} > 0$)
rekey
Else If ($LastRekeyTime < LastMemberLeaveTime$)
rekey
For a leave: *If* ($n_{TS_h} > 1$)
rekey

1. If a host h joins multicast session G from a trusted subtree that has previously been part of the multicast tree for session G, then if the trusted subtree currently has session members or if the group has not been rekeyed since the last session member from this trusted subtree left the group, then rekey.
2. If a host h leaves multicast session G from a trusted subtree that will remain part of the multicast tree for session G, then rekey must occur.
3. Otherwise, there is no need to rekey.

4.4.2 GACA-GKM System: Providing Topology Information

A GACA-GKM implementation would require the group key controller to know certain information such as the trusted subtree of each host, the trusted subtree hierarchy, and if the number of members of a trusted subtree is 0, 1, or greater. We propose three ways to obtain this information.

4.4.2.1 Traceroute-type Approach

The mtrace tool [71] allows multicast receivers to learn the route to a multicast source. It is an extension of the traceroute tool for unicast routes. Tracetree [183] allows a multicast source to receive mtrace data for each of its group members. The tracer protocol proposed by Levine, et al. [127] organizes receivers into a logical trees using mtrace packets. These techniques provide more information than is needed for GACA-GKM. We can reduce this topology information to just the trusted subtree hierarchy. Also, for each leaf trusted subtree, the system does not need the exact count of child receivers. It only needs to know if the count is 0, 1, or greater.

4.4.2.2 Topology Inference-based Approach

Topology inference techniques such as those proposed by Ratnasamy, et al. [171] and Duffield, et al. [62] observe network characteristics and make inferences about the multicast topology. This technique are not as accurate as traceroute-type approaches. In general, these techniques are used when the receivers cannot be explicitly polled to determine the topology.

4.4.2.3 Enhanced IGMP-based Approach

IGMP [33] does not send topology or membership information to the source. It only sends an indication there is some number of interested receivers downstream. EXPRESS multicast included a proposal for an enhanced IGMP that provides group

membership counts [100]. This approach provides only the count. GACA-GKM also needs the trusted subtree information. We propose changes in the IGMP protocol that allows the hierarchy information to be passed up the tree up to the source.

4.5 Evaluation

We now present separate evaluations of Gothic and of GACA-GKM. The first subsection examines the efficiency of Gothic in terms of message overhead and computational overhead. The second subsection presents evaluation results showing that GACA-GKM reduces message overhead by 50

4.5.1 Gothic Evaluation

We evaluate Gothic in the multicast environment because the number of users is larger than the number of anycast servers so this provides the best evaluation of the scalability of the architecture with regards to group size. This also allows the performance to be compared to Hardjono and Cain's and Ballardie and Crowcroft's secure IGMP schemes reviewed in the related work section. To simulate the performance of these schemes, we use data collected by the Mlisten [14] tool over several days for the Mbone multicast of the Space Shuttle Mission STS-80 in November 1996. The session has a duration of 13 days and over 1600 join requests. Figure 20 shows the group membership over the length of the session.

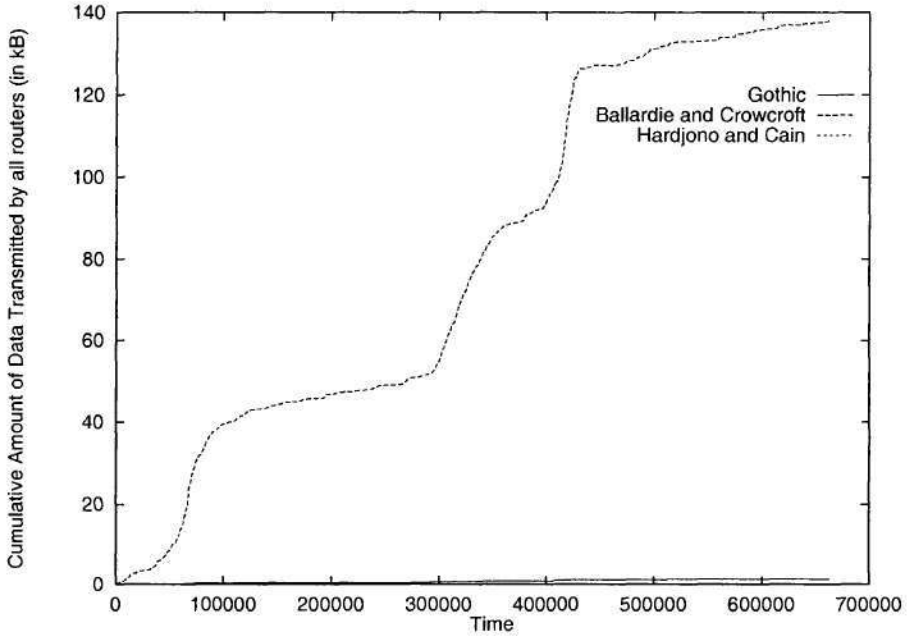
Figure 18(a) shows the total network overhead at all last-hop routers involved in the system. The amount of data transmitted by Gothic and Hardjono and Cain's scheme is 1.2 KB compared to 138.1 KB by Ballardie and Crowcroft's scheme; therefore, they are not clearly visible in the figure. Figure 18(b) shows the total network overhead at all group members. Figure 19(a) shows the cumulative network overhead at the ACS. Figure 19(b) shows the overall network overhead. These figures show that Gothic involves less than half of the total network overhead of the

Operation	Performance
3DES encryption	4.75 MB/sec
MD5 message digest	100.74 MB/sec
HMAC/MD5 message digest	99.86 MB/sec
RSA 1024 signature	10.29 sec
RSA 1024 verification	0.30 sec

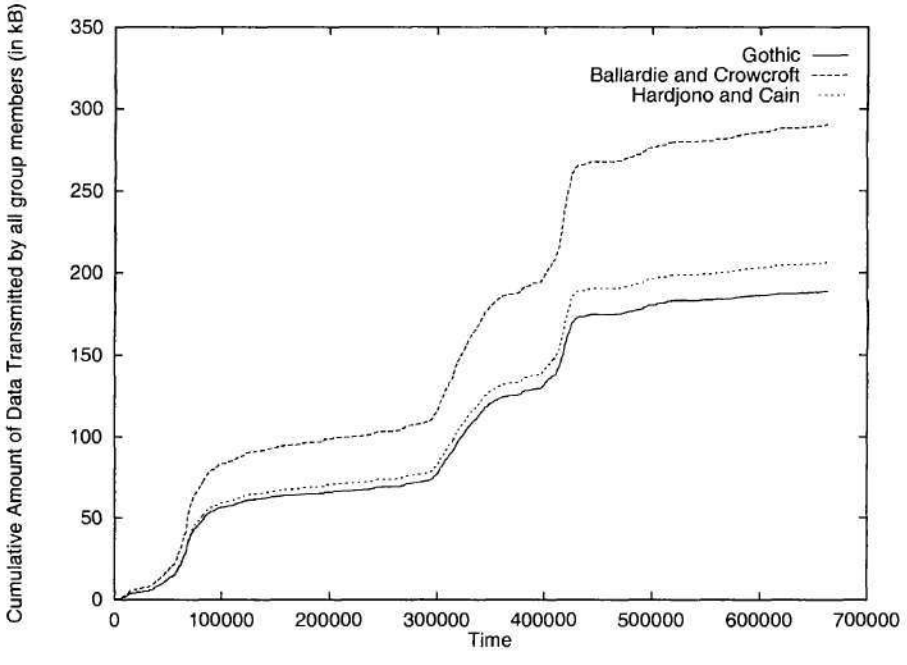
Table 2: Cryptographic computation processing time

other two schemes. The related work section above explains how such performance improvements were achieved.

To analyze the computational overhead, we determine the number of computational operations invoked by each of the schemes. We evaluate these operations at the router and access control server. We then translate the number of operations to the actual computation load by evaluating the processing time involved with these operations. The computation operations include host authentications, digital signatures creation and verifications, authorization lookups, and encryptions. The processing times for the computation operations are based on benchmarks published for the publicly available Crypto++ library [53]. The simulation used 128-bit Triple DES encryption, MD5 message digest, RSA 1024-bit digital signatures, and IPSec AH with HMAC-MD5 authentication. The performance of each of those operations is shown in Table 11. Figure 21 shows the computational overhead of the three schemes at the router in terms of processing time. The computational overhead of Gothic is an order of magnitude less than that of the other schemes. This shows that the Gothic authorization system achieved its goal of reducing the computational overhead at the router. Again, the related work section discusses the operations of these two schemes relative to Gothic and explains how such performance improvements were achieved.

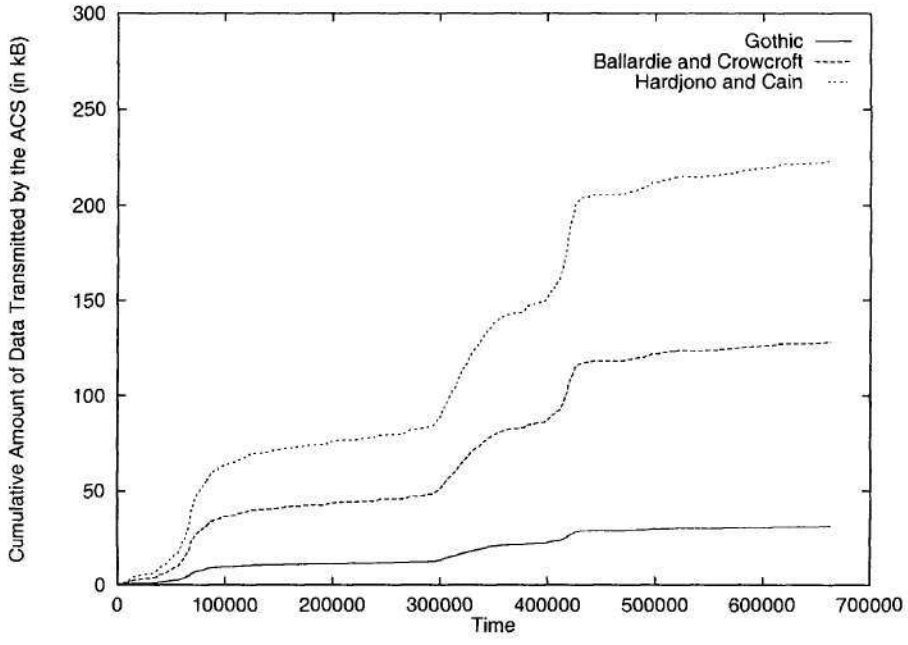


(a) Router

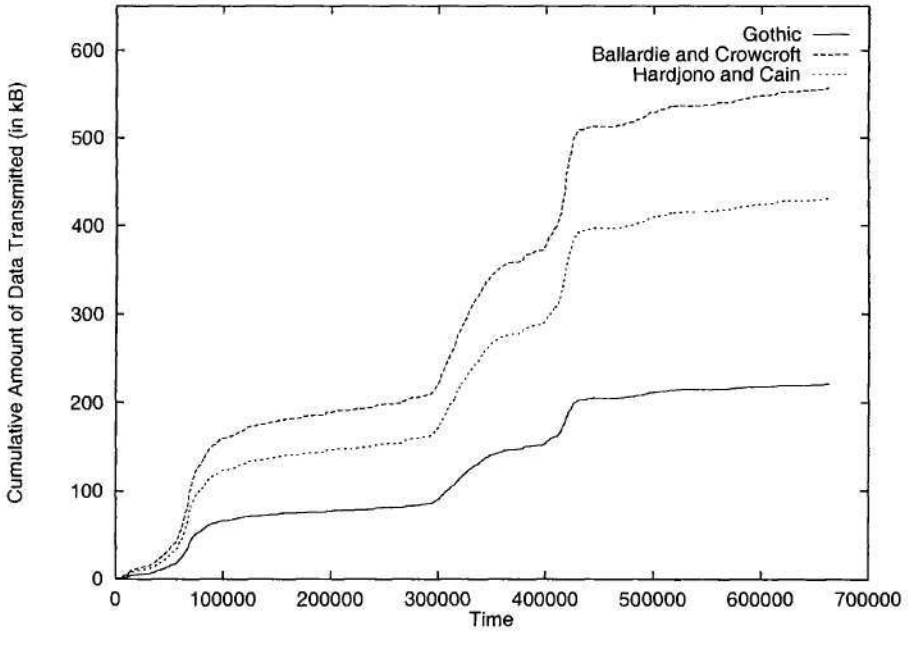


(b) Host

Figure 18: Network overhead at routers and hosts



(a) ACS



(b) Total

Figure 19: Network overhead at ACS and overall

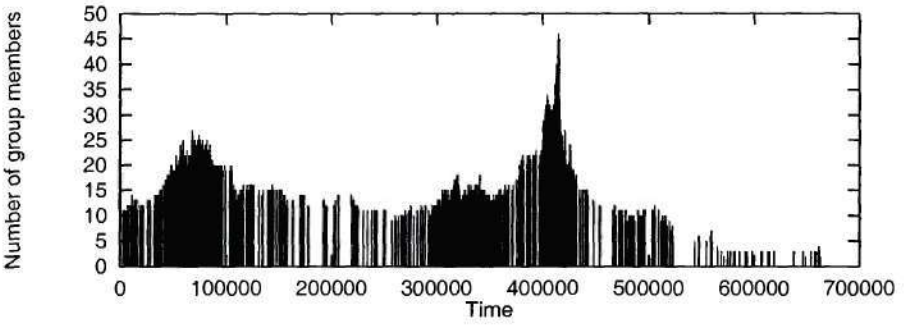


Figure 20: Group membership

4.5.2 GACA-GKM Evaluation

We next provide simulation results to show the efficiency gains of GACA-GKM over traditional approaches. In addition to the NASA session trace described above we also use a trace from a simulated multicast group. This allows us to simulate the performance for a range of trusted subtree sizes. The simulated multicast group model has the following parameters:

1. The pool of potential receivers has $65,536$ receivers. Each receiver joins and leaves the group independently.

2. The length of an individual *active* phase is an exponential distribution with an average of τ. The length of an individual *inactive* phase is an exponential distribution with an average of 10τ. The ratio of active to inactive duration is $1 : 10$, so the average group size is approximately $5,958$ receivers during steady state.

3. The length of the group session is 100τ.

We evaluate the GKM message overhead at the group key controller. We use a logical key hierarchy (LKH) [209, 204] as the underlying rekeying algorithm. Thus, we compare traditional LKH to GACA-LKH, LKH using the GACA-GKM technique. The best performance gains are achieved when the group access control system is

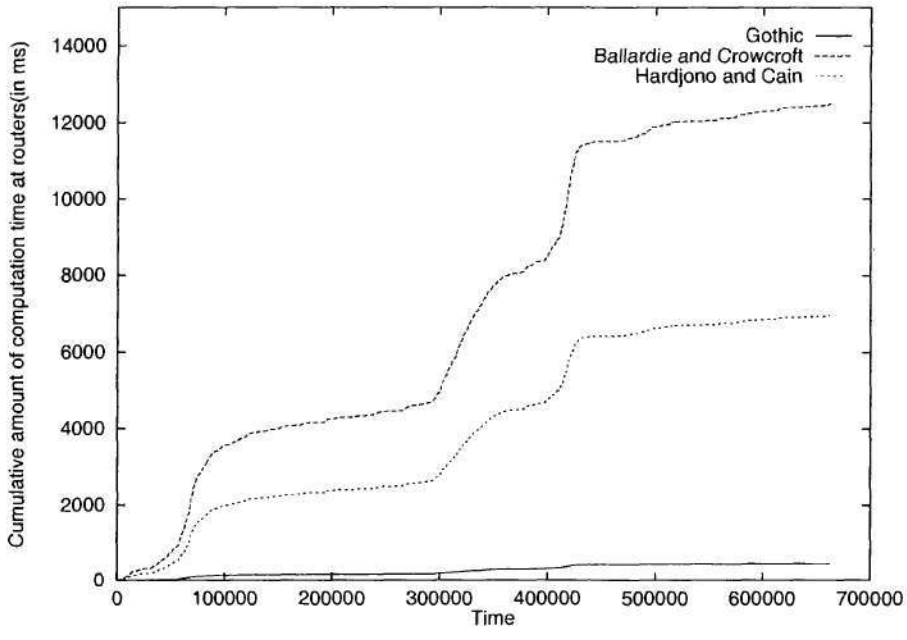
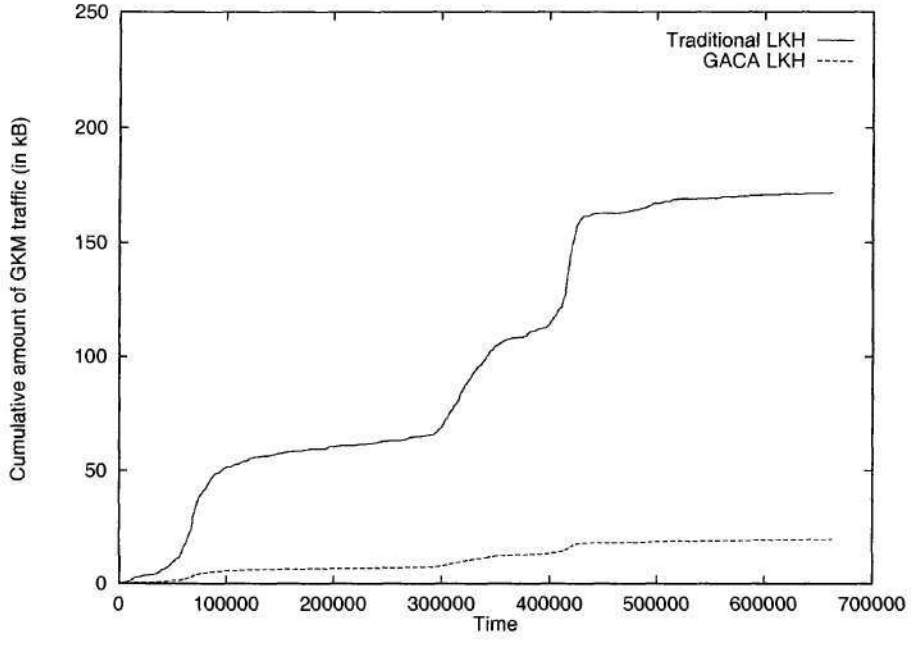
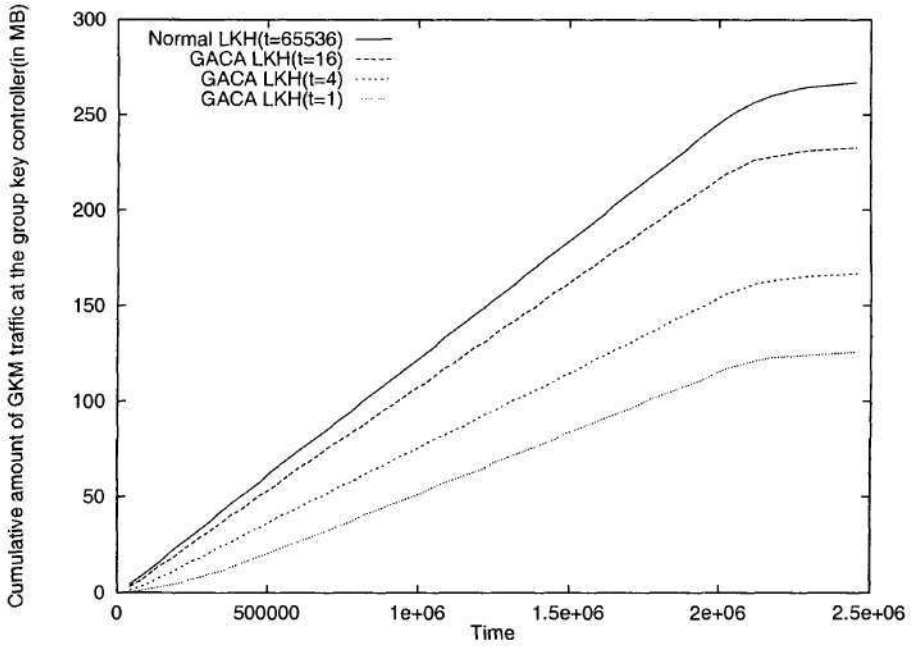


Figure 21: Computational overhead at last hop routers

widespread. This is the case when each router is trusted so each trusted subtree includes one host. The other extreme is if the group access control system is not present. In this case, the scope of trust only includes the source. Thus, the only trusted subtree is the one rooted at the source and it includes all n group members. This is equivalent to traditional GKM. We evaluate the scheme in a range of environments based on the average trusted subtree size which is denoted by t. Figure 22 shows the GACA-LKH ($t = 1$) performance in terms of GKM message overhead at the group key controller as compared to traditional LKH ($t = 65,536$). Figure 22(a) shows the results for the actual mlisten trace data. GKM traffic overhead is reduced from 171 KB with traditonal GKM to 19 KB with GACA-GKM. The simulated trace results in Figure 22(b) show how the size of the trusted subtrees affects the overhead. The graph shows GACA-LKH denoted as $t = 1$ as compared to traditional LKH denoted as $t = 65,536$ and a range of environments in between consisting of various values of t. For traditional LKH where $t = 1$, the overhead is



(a) Actual Trace



(b) Simulated Trace

Figure 22: Group key management traffic overhead

267 MB. By introducing some trusted routers such that $t = 16$, the overhead is reduced to 233 MB. In systems in which all routers are trusted, the overhead is reduced to 128 MB.

4.6 Conclusions

The need for a solution for secure multicast group joins and secure anycast server advertisements is well-known. We have generalized these problems into a single group access control problem and proposed a secure and scalable solution, Gothic. We presented a novel authorization system that improves the scalability and security over previous solutions. We also presented solutions for group owner determination and authentication. We have introduced GACA-GKM and evaluated the performance improvements. We have presented Gothic in the context of many flavors of multicast and anycast including Global IP-Anycast, application-layer anycast, Source-Specific Multicast, and Application-Layer Multicast.

Chapter 5

Rights Management in Peer-to-Peer Systems

Peer-to-peer file sharing systems allow content to be shared between distributed end-systems. Files stored at peers are transferred directly between peers rather than storage at or transmission through a central server. The last few years have seen the popularity of such systems grow tremendously. One such system, Napster [148] was the fastest growing application on the Internet, boasting a total of 50 million users downloading its software. Users are drawn to peer-to-peer systems to locate and retrieve a wide variety of content.

There are two main varieties of peer-to-peer systems. *Centralized* systems such as Napster provide indexing and searching functions at a centrally managed location or a set of replicated locations; while *decentralized* systems such as Gnutella [78] and Freenet [74] depend on the peers themselves to manage content indexing and search functions in a distributed manner. In both types of systems, content is exchanged directly between peers.

The large number of users freely exchanging content has increased the interest of content creators and owners in the protection of content that can be shared on these systems. They, along with legal authorities are attempting to force peer-to-peer system operators and users to control the exchange of content on their systems. In this context we use the term *content protection* to refer to the ability to control or restrict the exchange of content within a peer-to-peer file sharing environment.

We emphasize here the central importance that effective content protection will

play in the future success of many peer-to-peer systems. Napster faced many legal/court imposed obstacles to its operation because of its lack of

content protection, and will re-emerge only after implementing stringent content protection functions [51]. Other existing *decentralized* peer-to-peer systems such as Gnutella [78], Freenet [74], KaZaA[118] and Morpheus[146], continue to operate without content protection, but some are constantly under legal pressure to implement content protection [28]. We believe that it is a matter of time before many peer-to-peer file sharing systems will have a need for content protection features. While on one hand content providers are actively trying to stop these peer-to-peer systems from allowing uncontrolled content distribution, at the same time, content providers are actively seeking content protection technology that will allow them to effectively leverage the popularity of peer-to-peer distribution. The content providers interest and exploration shows in many forms including working groups and requests for proposals [2, 1], creating new companies [163, 147], or acquiring technology companies [6].

Some people are opposed to content protection systems because they feel that such systems unfairly restrict the users ability to access content. We aim to provide a system that provides the level of protection needed by content owners while maintaining the flexibility that end-users desire. This is ultimately beneficial to end-users because content owners will not make content available in the forms that users want until adequate protection measures are in place. By providing a system that meets the needs of both parties, this will ideally increase access to a greater variety of content and flexible business models that users are accustomed to from other distribution models.

Some content protection systems have been implemented or proposed for centralized peer-to-peer systems [122]. Such systems rely on the central authority that maintains the indexing and location functions to provide content protection and, therefore, cannot be applied to decentralized peer-to-peer systems. In this paper we present CITADEL, a novel content protection system designed specifically for

use within a decentralized peer-to-peer system [1]. CITADEL builds a protected file sharing environment over a normal peer-to-peer network using secured content objects and file sharing software enhanced to perform protection operations. A flexible content importation system that is part of CITADEL allows all users to insert new content as well as additional copies of protected content. Our work also includes an implementation that shows that CITADEL is a practical and lightweight approach to creating a protected peer-to-peer file sharing environment.

This chapter is organized as follows. We begin by discussing the benefits of content protection in Section 5.1. We then motivate the approach of an overlay security layer in Section 5.2. Section 5.3 provides some background discussion on peer-to-peer environments and explains the design issues in a content protection architecture for peer-to-peer file sharing systems. Section 5.5 presents an overview of the CITADEL architecture. Section 5.8 presents the detailed operations of the CITADEL components. We conclude in Section 5.9.

5.1 The Benefits of Content Protection

Some people are opposed to content protection systems because they feel that such systems unfairly restrict the users' ability to access content. There are many interesting legal and policy questions concerning file sharing and "fair use". We do not attempt to answer those questions or provide a technical solution to those questions. Instead, we argue that in many environments the ability to provide content protection will provide benefits to many different parties. Therefore, content protection systems should be developed so that they are available for environments that can benefit from them. The idea of providing protection in a content distribution system is not new. Most content distribution systems are built upon the ability to provide access control including cable television, video-on-demand, information

[1] The system can also be viewed as an alternative to current proposals for content protection in a centralized system. We focus, in this paper, on its use in decentralized systems.

websites, and even print mediums.

It is often portrayed that the primary goal of content protection is to control or stop file-sharing systems such as Gnutella and Freenet. We suggest that this is not the single goal of content protection work and that these systems alone cannot obtain this goal. For those interested in achieving this goal, there are two possible ways that protection systems can help:

1. Protection systems for peer-to-peer systems alone cannot stop users from creating new file-sharing networks and freely exchanging content. One view is that along with other protection mechanisms such as secure audio compact discs, operating systems and hardware, peer-to-peer protection may help provide an overall, security-in-depth solution.

2. More importantly and more realistically, *protection systems will allow the creation of legitimate file-sharing systems.* Most users are willing to pay for access to content and the conveniences that accompany such a service [65]. They only use existing file sharing systems because there are no legitimate alternatives. So even if rogue networks do exist, they will represent a minority of the users and such practices have become accepted by content providers in other distribution mediums [157]. Content providers aim to "keep honest people honest" [27] and peer-to-peer content protection provides a solution to achieve that goal.

The introduction of content protection systems for peer-to-peer networks will allow content providers to safely take advantage of the numerous benefits of the peer-to-peer distribution paradigm. This will lead to the availability of more content, richer content, new applications, and traditional content distribution business models in peer-to-peer systems. Thus, content protection will benefit peer-to-peer network operators, content providers, and end-users. We argue that the currently popular file-sharing networks are only the tip of the iceberg in peer-to-peer content

distribution systems and that future systems will build upon the foundation provided by content protection to allow rich, flexible, on-demand content location and access functionality.

Content protection can provide benefits for the different parties involved:

- *Network Operators:* Effective content protection will play a central importance in the future success of many peer-to-peer file-sharing systems. Systems such as Morpheus and Napster were shut down after legal pressures due to their lack of content protection. Other existing systems continue to operate without content protection, but some are constantly under legal pressure to implement content protection [28]. It may be only a matter of time before operators of many peer-to-peer file sharing systems are obligated to provide content protection features.

- *Content Providers:* There are a number of parties ranging from individual musicians and producers to large content providers that desire to utilize peer-to-peer distribution, but require the ability to protect their content. Content providers are actively trying to stop peer-to-peer systems from allowing uncontrolled content distribution. At the same time, however, content providers are actively seeking content protection technology that will allow them to effectively leverage the benefits and popularity of peer-to-peer distribution. The content providers' interest is evident in their participation in working groups [2], requests for proposals [1], creation of new companies [163, 147], and acquisition of technology companies [6].

- *End-users:* We argue that the lack of content protection is currently hindering the introduction of richer content distribution systems. Content protection is ultimately beneficial to end-users because content owners will not make content available in the variety, quality, and formats that users want until adequate protection measures are in place. Content protection will enable providers to offer flexible business models that users are accustomed to from

other types of distribution systems. These systems will possibly offer more reliability and convenience than systems not supported by providers. Protection systems should aim to provide the level of protection needed by content owners while maintaining the flexibility that end-users desire.

5.2 The Case for an Overlay Security Layer

In this section, we describe current peer-to-peer environments, discuss the problems with previous approaches to content protection for peer-to-peer systems and describe how an overlay security layer-based approach overcomes these problems.

5.2.1 Our Approach

We argue that previous peer-to-peer content protection systems are inadequate and we propose a new approach based on the use of an Overlay Security Layer (OSL) that is a secondary overlay layer that is built on top of the existing peer-to-peer network and below the application. Due to this layering, the OSL is able to provide content protection by securing the content and controlling all user access to content. Previous attempts at content protection for peer-to-peer systems have included proprietary systems designed for centralized peer-to-peer systems. These systems are not portable across different peer-to-peer systems; thus, each peer-to-peer application developer must create a custom content protection solution. Furthermore, these systems cannot be utilized for decentralized peer-to-peer systems such as Gnutella, Morpheus, and KaZaa. Also, these systems only provide the ability to enforce simple all-or-none access policies; therefore they provide limited protection and do not provide the flexibility to support traditional business models.

We propose the use of an OSL to provide solutions to these issues and describe CITADEL [108] as an example of an architecture based on this approach.

CITADEL's OSL builds a protected file-sharing environment over a normal peer-to-peer network by being a secondary overlay situated on top of the existing peer-to-peer overlay infrastructure. We explain four benefits of the OSL: portability including use in decentralized peer-to-peer systems, reusability, increased security, and support of new applications and business models.

5.2.2 Environment Description

Peer-to-peer file sharing systems consist of two components, the file location process and the file retrieval process. In most peer-to-peer systems, the file retrieval process is decentralized. That is, files are transferred directly between peers rather than through a client-server model. However, peer-to-peer systems differ in the file location process. As previously defined, there are centralized and decentralized systems. Among the decentralized systems, there are naive broadcast query systems such as Gnutella [78] and distributed hash table (DHT)-based systems such as CAN, Chord, Pastry and Tapestry [170, 194, 179, 214].

Peer-to-peer networks are formed as a logical connection of endhosts over the physical network. Thus, peer-to-peer systems currently add one layer over the normal network. We refer to this as the *distribution layer* and it includes content location, routing, and retrieval mechanisms. We propose a *protection layer* that provides security services independently of the specifics of the distribution layer. We show that the functionality provided by the protection layer enables greater application functionality and consequently further defining the *application layer* for peer-to-peer systems.

5.2.3 Problems with previous approaches

One of the most well-known attempts at a content protection system for peer-to-peer was implemented by Napster [148]. Napster's centralized design requires information about shared files to be sent to the central server where they are indexed. All

queries are sent to the central server which replies with the location of the file. This design provides two points to restrict the ability to distribute certain content: upon indexing and upon querying. To restrict content upon indexing, when a user attempts to share a file that is not approved, the central server does not index the file. To restrict content upon querying, when a user searches for an unapproved file, the central server does not return a valid response.

One issue is that Napster's content protection system relies on the central server to store and enforce the access control policy. Therefore, this approach cannot be applied to decentralized peer-to-peer systems because there is no central server that can be used. The most widely used peer-to-peer systems today are decentralized and therefore lack any means of providing content protection.

A second problem that peer-to-peer distribution has faced is the lack of service models and business models from traditional content distribution methods. Systems have been unable to offer these models because of the absence of the protection functionality required to support them. Previous content protection systems can only enforce simple all-or-none access control policies which is not enough to support popular business models that require a more granular and flexible access control policy.

A third issue is that these content protection systems are not portable across different peer-to-peer systems; thus, each peer-to-peer application developer must create a custom content protection solution. One reason these systems lack portability is that they are situated in the middle of the file location process. As we have shown earlier, the file location process varies across different types of systems. However, the file retrieval process maintains common features across all peer-to-peer systems. So, in order to achieve greater portability content protection systems should be implemented as part of the file retrieval process rather than the file location process.

5.2.4 Overcoming those problems

We propose the CITADEL architecture that provides solutions to these issues. CITADEL is based on an overlay security layer that builds a protected file-sharing environment over a normal peer-to-peer network. This protection layer is a secondary overlay that is layered on top of the existing peer-to-peer overlay infrastructure. There are at least four advantages to using an OSL to provide content protection in peer-to-peer systems:

1. *Decentralized approach:* A primary goal of the OSL is to provide a content protection system that can be used in decentralized peer-to-peer systems. This is achieved by pushing the required functionality from the central server to the peers. The OSL allows decentralized storage of the access control policy by storing the policy at the peers along with the content and allows decentralized enforcement of the policy by providing the protection mechanisms at the peers.

2. *Reusability:* The OSL provides an architecture that can easily be integrated into different peer-to-peer systems and applications. The protection functionality provided by the OSL is built into applications as an underlying library or API making it transparent and easy to incorporate. Thus, peer-to-peer application developers can include protection functionality with minimal effort and without significant application changes. This also allows applications to continue to take advantage of specialized functionality such as lookup and routing functionality without interference from content protection mechanisms.

3. *Enhanced Security:* The overlay security layer is able to create a *protected file-sharing environment.* Thus, only protected objects can be exchanged within this peer-to-peer system. Additionally, all content objects in the system are protected and all access to these objects is controlled. In the past, the goal of content protection in peer-to-peer systems has been to restrict certain content from being exchanged within the system. Thus, these systems only provided

all-or-none access; if content was allowed in the system, then anyone in the system could access it. Due to the efficiency provided by the OSL, the protection goals can be expanded to be able to control access to content on a per user basis. Due to the placement of the content protection mechanisms between the application and the distribution mechanisms, the system is able to provide *protected distribution* by controlling the user's ability to retrieve the content and provide *protected storage* by controlling the user's ability to access the plaintext content within a local copy of the content.

4. *Enables new applications and support business models:* While content control has been criticized in some circles as spelling the end of true peer-to-peer file sharing, we suggest that it may actually be beneficial in that it has the potential to enable many different and desirable service models. As we have mentioned, there are a number of common content distribution business models that peer-to-peer systems have been unable to support because they lack enhanced protection functionality. The OSL provides flexible protection functionality that the application can interact with thereby enabling new application functionality and business models. We describe three such models in Section 6.1.

5.3 Background and Design Issues

Peer-to-peer file sharing systems have two parts, the file location process and the file retrieval process. In most peer-to-peer systems, the file retrieval process is decentralized. That is, files are transferred directly between peers rather than through a client-server model. Peer-to-peer systems differ in the file location process. There are centralized and decentralized file location systems. Among the decentralized systems, there are naive broadcast query systems such as Gnutella [78] and distributed hash table(DHT)-based systems such as CAN, Chord, and Pastry [170, 194, 179].

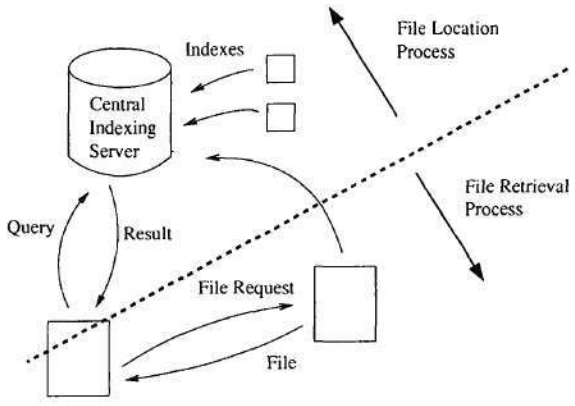
In centralized file location systems such as Napster, information about shared files is sent to the central server where they are indexed. All queries are sent to the central server which replies with the location of the file as shown in Figure 23(a). This design provides two points to restrict the ability to distribute certain content: upon indexing and upon querying. To restrict content upon indexing, when a user attempts to share a file that is not approved, the central server does not index the file. To restrict content upon querying, when a user searches for an unapproved file, the central server does not return a valid response.

The file sharing processes of a broadcast query-based file sharing system and a DHT-based file sharing system are shown in Figures 23(b) and 23(c). These figures show the absence of a central entity in the file location process. Thus, it is clear that the content protection approach used in centralized systems cannot be directly applied to decentralized peer-to-peer systems because there is no central location to filter indexed files or searches.

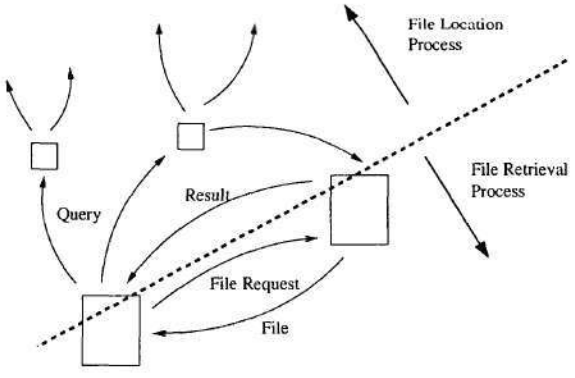
For each type of file sharing system, Figure 23 shows the separation of the file location process and the file retrieval process. It can be seen here that even though these three systems vary significantly in the way that files are located, the file retrieval processes of the systems are nearly identical.

In CITADEL, therefore we create a content protection system that focuses on the file retrieval process. Because of its independence from the file location process, CITADEL can be used in any peer-to-peer file sharing system including centralized, query-based decentralized, DHT-based, or some hybrid systems

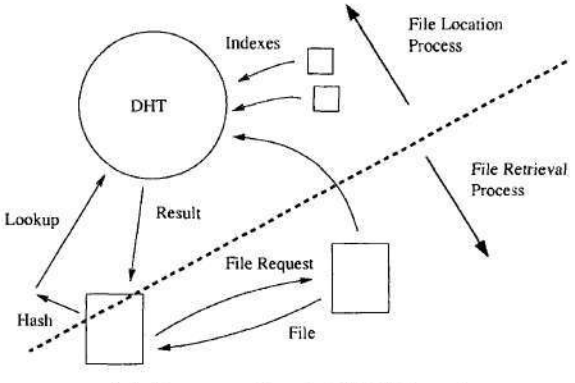
In developing our proposal for CITADEL, we start with an assumption that *content rights lists* are provided by content owners. This assumption is based on the involvement of content owners in recent real-world content protection issues [122]. These lists describe the access rights associated with each content object. The rights described may be dynamic and can change over time.



(a) Centralized



(b) Decentralized, Query-Based



(c) Decentralized, DHT-Based

Figure 23: File location and retrieval in different peer-to-peer systems

5.4 CITADEL Objectives

We outline the design objectives of a content protection system for decentralized peer-to-peer file sharing systems.

- **Content Protection:** In the past, the goal of content protection in peer-to-peer systems has been to restrict certain content from being exchanged within the system. Thus, these systems only provided all-or-none access; if content was allowed in the system, then anyone in the system could access it. In CITADEL, the fundamental content protection goal is to be able to control access to content on a per user basis. Controlling access refers to providing *protected distribution* by controlling the user's ability to retrieve the content and providing *protected storage* by controlling the user's ability to access the plaintext content within a local copy of the content. CITADEL does not aim to provide *output protection*–protection for the analog or digital output after access has been granted to an authorized user. Refer to section **??** for a discussion of related work in these three areas of content protection.

- **Maintain an open peer-to-peer sharing experience:** We define an open peer-to-peer environment as one in which, even in the face of content protection, all peers are equally able to insert content into the system including independent content and copies of protected content (including variations such as different formats or compression rates). This requires the content protection system to be able to appropriately secure all content that is introduced into the system without regard to the peer inserting the content. Without this ability, systems have struggled to find the correct balance of openness and security [5]. We suggest that without the openness, the peer-to-peer system loses many of its attractive features and resembles a client-server model in many respects. When most or all content is introduced by a central source, a peer's role turns from content provider to content cache. At this point, the system is effectively a client-server based distribution system with extensive

caching. In section 5.6.4, we discuss the *content importation system* that will provide such functionality within CITADEL.

- **Avoid dependency on trusted client software:** Providing content protection in a decentralized peer-to-peer system requires modifications or additions to the file-sharing software. We assume the presence of malicious users that wish to circumvent the content protection system. The system should be robust against attacks by users with full access to the software and the operating system on their computer. Additionally, the system should not rely on the file-sharing software being tamper-proof or trusted software.

- **Maintain privacy:** The content protection system should at least maintain the level of privacy that exists in the normal file-sharing environment. It should be possible to allow a user to obtain access rights and to be authorized without providing identifying information. Additionally, the system should be able to interoperate with the work in anonymous systems such as anonymous communication [189, 172, 173], anonymous authorization [40], anonymous payment [41, 29], and anonymous peer-to-peer file sharing [185, 74].

- **Avoid dependency on centralized security infrastructure:** In a decentralized file-sharing system, the content protection system should not introduce a single central authority. The protection system should allow a decentralized security infrastructure that can support multiple separate hierarchies of trust and control. For example, all content should not be controlled by a single entity. A more realistic approach would be to allow all content providers to establish independent trust systems. Also for a given trust system, the architecture should allow decentralized entities to perform the necessary operations rather than a central server. The security infrastructure should be flexible to allow different types of authorization and payment systems depending on the requirements of each content provider.

- **Provide the flexibility to support common content distribution business models:** There are a number of common content distribution business models that peer-to-peer systems have been unable to support because they lack enhanced protection functionality. While content control has been criticized in some circles as spelling the end of true peer-to-peer file sharing, we suggest that it may actually be beneficial in that it has the potential to enable many different and desirable service models. For example, we suggest three such models: *Pay-per-view and Subscription Model*, *Syndication Model*, and *Reseller Model*. In chapter 6 we explain how CITADEL supports these models.

5.5 Overview of the CITADEL Architecture

In this section, we provide an overview of the CITADEL architecture. We discuss possible approaches to a decentralized content protection system and motivate our approach. We then introduce the components of CITADEL.

5.5.1 Our Approach

As explained above, due to the nature of decentralized peer-to-peer systems, there is no central authority in the file sharing process, so policy enforcement must be done at the peers. This implies that peers must know the access control policy. The question is, how do the peers securely and efficiently access the global content rights list in order to enforce it? The content rights list is a form of an Access Control List(ACL). In the *distributed ACL* approach, the content rights list is distributed to all peers. However, there is significant overhead associated with distributing the entire content rights list to all peers. An alternate approach, the *queried ACL approach*, is to have the peers access the list by querying the content rights list server as necessary for each access to a content object. However, there is significant overhead associated with repeatedly querying the content rights list server [2].

[2]Section 6.2 presents an evaluation of CITADEL relative to these two approaches.

To avoid these types of overhead, our system takes a different approach in which the access control policy for each content object is stored with the content object. Thus, every access control policy that a peer must enforce is available and accessed locally. Compared to the queried ACL approach, this approach behaves like a cache of the relevant access control policy information. This provides greater scalability than the other two approaches. In order to distribute the content rights list with the content, we must protect the integrity of each object's access control policy. To provide this and other protection, we introduce the concept of a *protected file-sharing environment.* The system builds a protected environment over a normal peer-to-peer network. Only protected objects can be exchanged within this peer-to-peer system. Thus, all content objects in the system are protected and all access to these objects is controlled.

Figure 24 shows the protected file-sharing environment. The system uses the peer-to-peer network strictly as a means of file location and distribution. CITADEL exists as the protection layer built upon this distribution layer. The service layer is, in turn, built on top of the protection layer. Thus, service providers and application developers can introduce new services and applications based on a peer-to-peer distribution model by building the services on top of the CITADEL protection layer.

5.5.2 CITADEL Components

In this section, we introduce the components of the CITADEL architecture. The components include the secured content objects, access tokens, the file sharing software, and the content importation system.

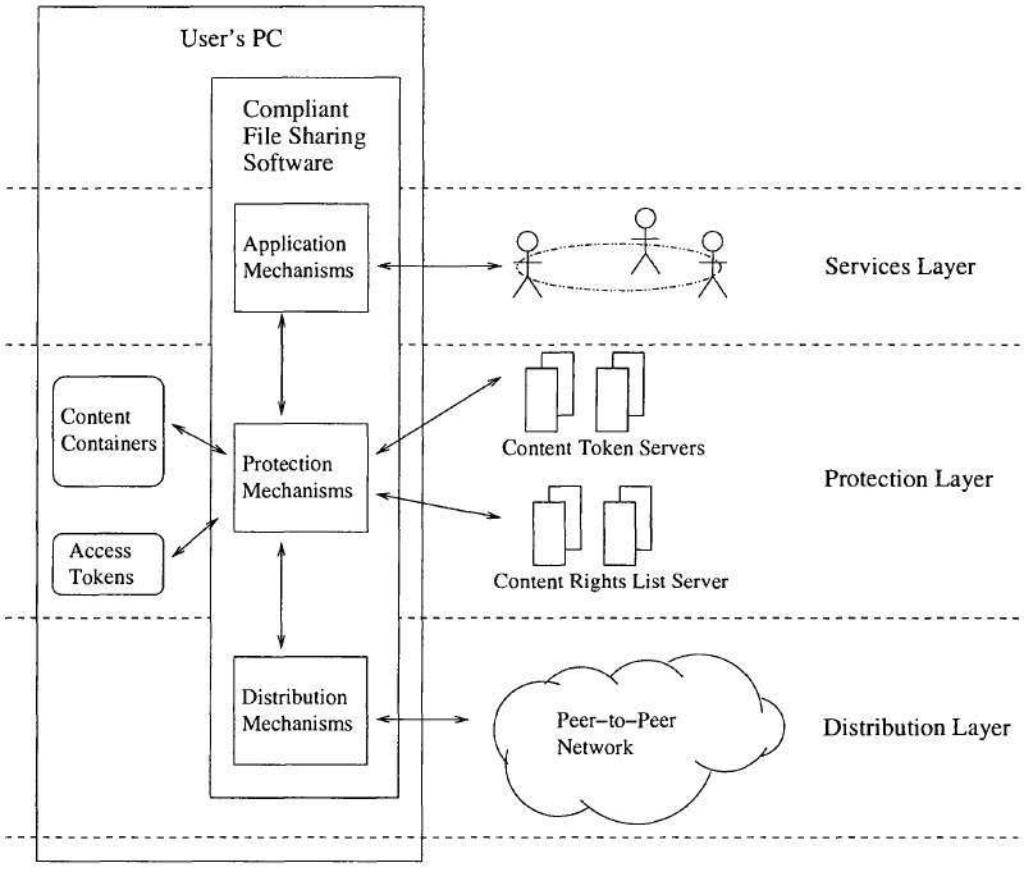


Figure 24: The CITADEL protected file sharing environment.

5.5.2.1 File Sharing Software

We refer to file sharing software that is enhanced with content protection software modules as *compliant file sharing software*(CFSS). Within Figure 24, there is a high-level diagram of the compliant file sharing software. The CFSS provides three types of operations: *distribution operations*, *protection operations*, and *application operations*. Distribution operations involve normal duties of peer-to-peer file sharing software such as interacting with the file location system and the file retrieval system. Protection operations involve interacting with the secured content objects to control access to the content. Protection operations also include periodic interactions with the content rights list server and the content token server as necessary. Application

operations involve supporting the application and the service model interacting with the protection layer to provide access to the content files in the secured content objects. We assume the presence of malicious users that aim to gain unauthorized access to content by circumventing the mechanisms in the CFSS. In Section 5.8, we show that even without any software protection or tamper resistance, the design of the system makes the CFSS robust against any such attacks.

5.5.2.2 Content Containers

Access to content is protected by the use of secured content objects or *content containers*, cryptographically secured objects consisting of a protection label and an encrypted content file. Figure 25 shows the elements of a content container. The protection label contains a content label that provides content identification and ownership information and the content's ACL. In Section 5.6.1.2, we explain that the ACL only requires a small number of entries. The protection label is digitally signed by the appropriate authority to ensure integrity. The content file is encrypted with a random key, K_R which is encrypted with K_{-RTS_ID}, a unique decryption key associated with the token needed to access the content. This encrypted form of K_R is stored in the content container. Additionally, the encrypted K_R and the protection label are encrypted with K_{CFSS}, the decryption key that is built-in the compliant file sharing software. Thus, only the CFSS can access the content container, and even the CFSS can not access the content file in the content container without K_{-RTS_ID}, which is provided to authorized users along with the content access token. The idea of protection labels and attaching protection labels to the objects that they describe has been in the security literature for some time [25]. Slightly more recent work extended this into the concept of a secure package for storing content and its controls [113, 190]. We utilize a similar concept and further define the details of a secure content package; however, our contribution is a complete architecture for content protection in decentralized peer-to-peer systems. Along

the way, we define the details of the content container and its interaction with its reference monitor [125], in our case, the CFSS.

5.5.2.3 Content Importation System

The only objects that can be shared in this protected environment are objects that have been imported into the system in the form of a content container with the appropriate access rights. The CITADEL architecture includes a *content importation system*(CIS) shown in Figure 26, that controls the insertion of objects into the system. It functions as the secure gateway to import any objects into the protected file-sharing environment. A key design goal of the CIS is that it allows content providers to easily protect content and insert it into the distribution network. The CIS enforces that content is identified correctly, labeled with the correct policy, and encapsulated in a content container. The *content importer* is a person or entity that inserts the content object into the protected environment. There are two classes of content that is imported into the environment: *new content* and *existing content.* New content is content that does not exist in the environment. The content importer that inserts new content is called the *content provider.* Existing content includes copies or different versions of content that has already been imported into the environment by the content provider. The CIS supports our goal of maintaining an open peer-to-peer sharing experience by allowing all peers to insert new content or existing content.

5.6 Detailed Operations of CITADEL Components

In this section we discuss the details of the different components of CITADEL. We first discuss how hosts obtain access tokens. We then explain host interaction during normal file sharing. Next, we present the details of the CFSS and the operations that it supports. Then, we discuss the content importation process. We then discuss

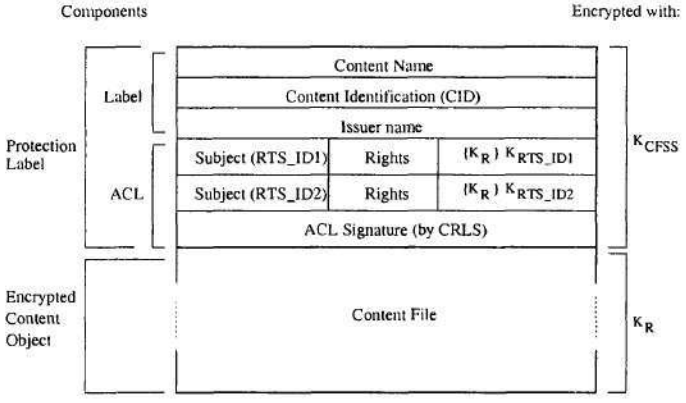


Figure 25: The structure of a content container.

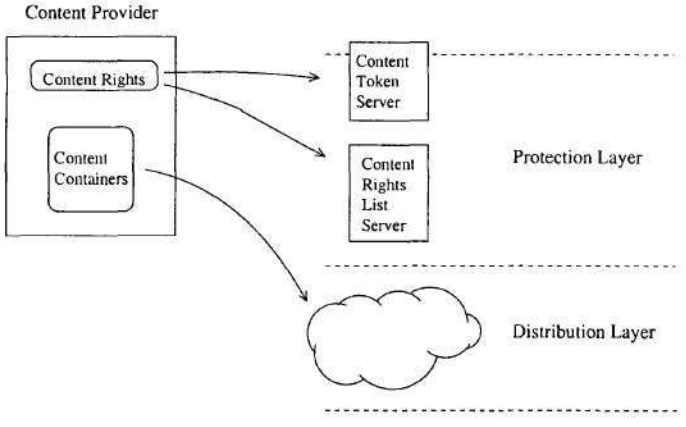


Figure 26: Content importation at a high-level.

common content distribution models that can be supported with CITADEL.

5.6.1 Token Distribution

Token distribution is the process of a user obtaining or purchasing rights to some set of content. This involves the user and the content token server. There are numerous proposals for electronic payment systems that can be leveraged for user authorization for token distribution; see for example [52, 177, 137, 41, 29]. Here, we explain the basic functionality necessary for CITADEL. For example, A user contacts a content seller and obtain a subscription to access all content on the network or perhaps all content in a particular category. We discuss this further in Section 6.1; for now, the point is that this process should be thought of as occuring infrequently and outside of the normal file-sharing experience. It is similar to how a cable subscription is set up for certain channels, but a user watches television regularly without dealing with obtaining new rights. The system can involve different content token services for different content providers. For each content provider, the content token service can be a single server or a group of distributed servers.

5.6.1.1 Detailed Token Distribution Protocol

Since a single instance of the authorization protocol takes place between a host h and a single content token server, CTS. The content token servers possess public-key certificates [102] that are used for authentication and digitally signing messages. This does not require a global public-key infrastructure; the key management requirements are minimal and are equivalent to the common use of public-key certificates for secure sockets layer/transport layer security (SSL/TLS) [58] web server authentication

The CTS has a public key certificate, $CERT_{K_{+CTS}}$ and the corresponding private key, K_{-CTS}. These are used to digitally sign messages and verify signatures. Digitally signed messages are shown in brackets with the key used to sign it as a

subscript $[message]_{K\_X}$ and encrypted messages are shown in curly braces with the key used to encrypt it as a subscript $\{message\}_{K_S}$.

The host establishes a encrypted communication channel with the CTS and sends an authorization request (AR). This secure communication channel can be achieved with SSL, IPSEC [120] or any other secure communications protocol. We show the communication encrypted with a session key, K_{sess}. The AR contains the content rights ID(RTS_ID) and the authorization information(AZ_INFO). RTS_ID is the subject in the ACL; its format depends on the authorization language that is used. We discuss this further subsequently. Also, the specifics of the authorization information depends on the payment system that is used by the content seller and may include account information or an electronic cash payment. In some situations the payment system may be out of band, and obtaining a rights token may only require some sort of authentication, such as a username and password. In some situations, no payment will be required to obtain certain rights tokens so the authorization information will not be needed.

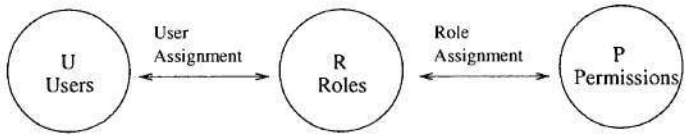
1.H $\rightarrow$ CTS:

$$AR = \{RTS\_ID, AZ\_INFO\}_{K_{sess}}$$

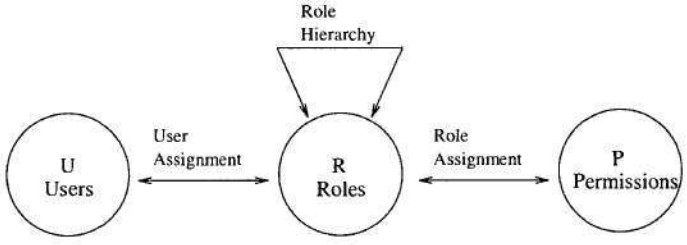
The CTS returns an authorization acknowledgment(AA) specifying a successful or unsuccessful authorization of the user for the requested rights. If successful, the AA contains an access token and the accompanying content decryption key, $K_{\_RTS\_ID}$. The access token includes the RTS_ID, the public-key for RTS_ID, and the expiration time, T_{exp}. $K_{\_RTS\_ID}$ is used for the content decryption key and for authentication. Recall that the content decryption key is required because it is used by the CFSS to access the encrypted content file in the content container.

2.CTS $\rightarrow$ H:

$$AA = \{\{[RTS\_ID, K_{+RTS\_ID}, T_{exp}]_{K\_CTS}\}_{K_{CFSS}}, \{K_{\_RTS\_ID}\}_{K_{CFSS}}\}_{K_{sess}}$$



(a) Flat RBAC



(b) Hierarchical RBAC

Figure 27: Role-based access control models.

5.6.1.2 Access Control Model

To provide an efficient solution that reduces the size of the content ACL and reduces the number of tokens needed by each user, we use role-based access control (RBAC) [181]. The basic notion of RBAC is that permissions are assigned to roles, users are assigned to appropriate roles, and users obtain permissions by being members of roles. The system can use *flat RBAC* or *hierarchical RBAC* [182].

Flat RBAC allows many-to-many relationships between user-role and permission-role assignments. Thus, a user can be assigned to many roles and a single role can have many users. Also, a single role can have multiple permissions and a certain permission can be assigned to multiple roles. Flat RBAC is illustrated in Figure 27(a) froma work by Sandhu, et al. [181]. This allows a user, in CITADEL, to be assigned to multiple roles. A particular role may have permission to access a particular content object or a group of content objects. For example, assume a system that has the following four roles: all content, jazz category content, contemporary jazz category (that is a subset of the jazz category), and jazz song X (that is a member of the contemporary jazz category). The ACL of the content object for jazz song

X specifies entries for each of these four roles. This allows a user to gain access by presenting a token that shows membership in any of the four roles.

Hierarchical RBAC allows role hierarchies, seniority relations between roles where a senior role possesses the permissions of the junior roles. The hierarchy can be expressed mathematically as a partial ordering: a reflexive, transitive and anti-symmetric relation. Hierarchical RBAC is illustrated in Figure 27(b) from [181]. In CITADEL, a hierarchical RBAC model allows a leaf role to have permissions relating to a specific content object and then have senior roles acquire permissions from these roles. This differs from the use of flat RBAC in that instead of role for the jazz category having explicit permissions to every object in the jazz group, the jazz role is a senior role that acquires permissions for jazz subcategory roles that acquire permissions from specific content roles. Thus, the ACL of a content object only needs to specify the single role corresponding to that object. In the CITADEL prototype described in section 6.3, we implemented hierarchical RBAC.

5.6.2 Host Interaction during File Sharing

Figure 28 shows host interaction during file sharing. Note that the file location process is not affected. For each file exchange, the user that receives the file is called the *downloader* and the user that sends the file is called the *uploader*. The downloader first locates the desired content and then sends a request including the access token to a peer that has the content. The uploader verifies that the downloader is authorized before providing access to the file.

5.6.2.1 Detailed Host Interaction Protocol

The process begins, as normal, with the downloader(D) sending a location query(LQ) to the file location system(FLS) as normal. The actual contents of the LQ depends on the file location system that is used. Here, we generalize to say that the LQ contains keywords(KW).

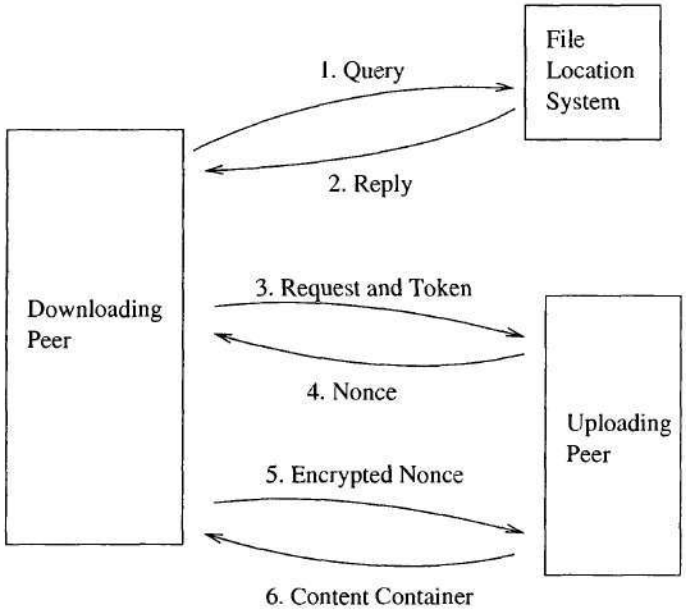


Figure 28: Host interaction during file sharing.

1.D $\rightarrow$ FLS:

$$LQ = [KW]$$

The file location system responds with a location reply(LR) specifies the location of the content.

2.FLS $\rightarrow$ D:

$$LR = [LOC]$$

The downloader sends a content request(CR) to the uploader(U) including the identification of the requested content and the appropriate access token. To authenticate the downloader, the uploader sends a random nonce, N, to the downloader and the downloader responds with a copy of the nonce encrypted with private key corresponding to the access token. This avoids the possibility of replay attacks by eavesdroppers or by the uploader. We must avoid the uploader being able to replay the authorization because the access token used by the downloader may contain

rights that the uploader does not have. This is the reason that tokens that require no additional authentication or tokens that use password-based authentication could not be used.

3.D $\rightarrow$ U:

$CR = [CID, TOKEN]$

4.U $\rightarrow$ D:

N

5.D $\rightarrow$ U:

$\{N\}_{K\_RTS\_ID}$

Upon successfully authenticating the downloader, the uploader verifies the access rights presented by the access token against the access rights specified in content's ACL. The uploader first checks the validity of the token. This includes verifying the CTS's signature and checking the expire time. The uploader then checks the access rights. The ACL of the content contains the name of the content provider or issuer, as shown in Figure 25. This is used to verify that the access token was provided by the correct content provider. After verifications, the uploader sends the downloader a content request acknowledgment(CRA) stating a successful or unsuccessful request. Upon success, the CRA contains the requested content file or instructions to obtain the file.

5.U $\rightarrow$ D:

CRA = Status (file or failure)

5.6.2.2 Discussion

Policy enforcement at the peers can take place at two times: 1) the file location process; or 2) the file retrieval process. Even though CITADEL can support both methods, we suggest enforcement during retrieval for several reasons. First, it allows

independence of the content protection functions from search functions which can differ widely among the different decentralized schemes as described earlier. Second, we feel that controlling the search functions without controlling the retrieval functions makes the system vulnerable to users who may be able to determine file locations by other means (e.g., users advertising file locations on web pages). So controlling file retrieval functions is necessary at any rate. Third, it removes the requirement of having to store access rights along with indexing information. Finally from a commercial viewpoint, allowing users to see what is available on the system without authorization might be an inducement for them to become paying customers. The file location service could also provide information about the access rights necessary to obtain the content. This access rights information could be provided as metadata just as systems currently report information such as the quality of the content of the size of the file.

5.6.3 Compliant File Sharing Software

In this section we describe the operations and security of the CFSS. The CFSS supports three operations: *content exchange*, *content import*, and *content export*. Content exchange is referred to as *content download* at the peer that is receiving the file and *content upload* at the peer that is sending the file. Content upload is the process of a user responding to a file request by transmitting the file to the other peer. Content download is the process of a user obtaining a file that has been requested. Content import is the process of a user inserting content into the protected file sharing environment from a content file stored in a native file format. Content export is the process of a user exporting content from the file sharing system by creating a copy of the content in its unprotected file format. The first subsection describes the details of the operations. The second subsection discusses the techniques used to protect the CFSS from malicious users.

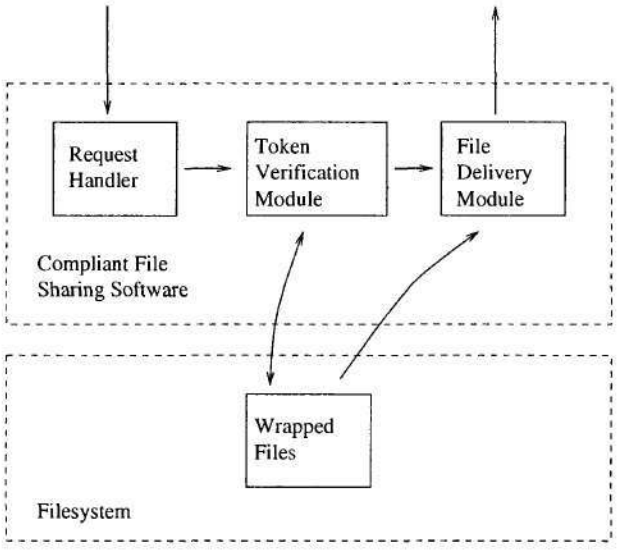


Figure 29: Content upload.

5.6.3.1 Operation Details

The content upload process is shown in Figure 29. During the content upload process, the uploader receives the request and token from the downloader, validates the token, checks the access rights, and then sends the file. The token verification module receives the token, verifies its authenticity, retrieves the rights from the token, and verifies these against the rights specified in the ACL of the requested content. Upon success, the file delivery module sends the file to the downloader.

The content download process includes a number of protection schemes. Even after the file is received, the user must possess the token and the content decryption key to access the content. Since the content container is double encrypted, the content's decryption key is required. This provides an extra layer of security by requiring the token and content key on both sides of the operation: 1) requesting the content and 2) receiving or using the content. Additionally, to provide more immediate propagation of access rights changes and to confine any security breach that may somehow occur, the administrator of CITADEL deployments can optionally have the CFSS re-identify received files and retrieve the ACL. This step can protect

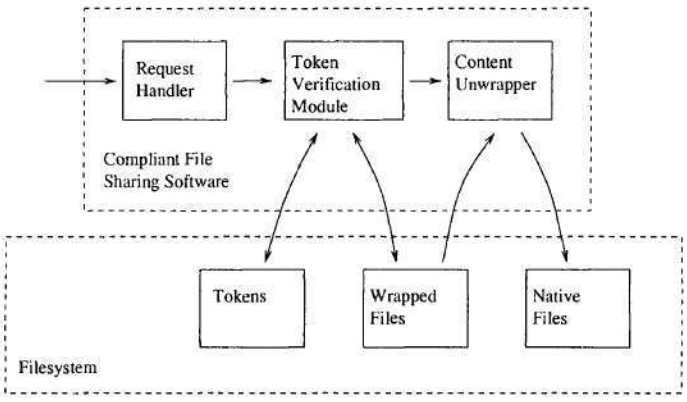


Figure 30: Content export.

against situations in which a user somehow bypasses the security of his CFSS, alters the ACL of the content, and distributes content with illegitimate rights. With this double checking of the ACL, the breach of the security is confined to an individual user.

Content export will not be necessary by many applications because CITADEL aims to provide as many of the functions related to the content as possible within the protected file-sharing environment in order to reduce the need for content export. For example, the CFSS allows the inclusion of a content player to allow content to be viewed or listened to while still in the content container. If the user needs to use the content for purposes other than this, then the file must be exported. The content export process is shown in Figure 30. The CFSS decapsulates the content file from the content container and stores a copy in the native unprotected format. The system can be configured to completely disallow content export by default or require additional access rights in order to export content; we discuss this further in section 6.1.

5.6.3.2 Policy updates and revocations

Here we discuss how the policy or the access rights for content can be updated or revoked by the content owner. By updating the policy at the CRLS, the new

policy is reflected in all future copies of the content imported. Also, if the system is configured with the option to perform ACL checking on file receipt, then all future downloaded copies of the content will get the new policy. Using these two methods, the system can update policy for all content objects imported after the policy update as well as for each existing content object that is shared after the policy update. Another approach is to use expiration times in the protection label that causes the CFSS to retrieve an updated policy for the content object after a certain period of time. These three methods do not provide the ability to explicitly revoke rights for a particular user. This could be achieved by revoking outstanding access tokens; however, due to the heavyweight operations of explicit revocation schemes, we do not use this method. Instead, we provide implicit revocation based on the use of time-limited tokens. The combination of these four methods provides a flexible framework for performing policy updates and revocations without requiring a heavyweight protocol.

5.6.4 Content Importation System

Content importation is the process of a user inserting a copy of content from a native content file into the file sharing system. The content importation system(CIS) is the secure gateway that assures the protection of content that is introduced into the protected domain. It involves modules on the CFSS that identify the content and retrieve the appropriate rights from the content rights service. This system is also the means that allows end users to introduce new content or new copies and formats of old content. We first provide an overview of the CIS. Then, we discuss the content identification process. We then describe the content importation process.

5.6.4.1 Overview of Importing Content

During content importation, the system identifies the content, retrieves the ACL and the necessary encryption keys, and encapsulates the content in a content container.

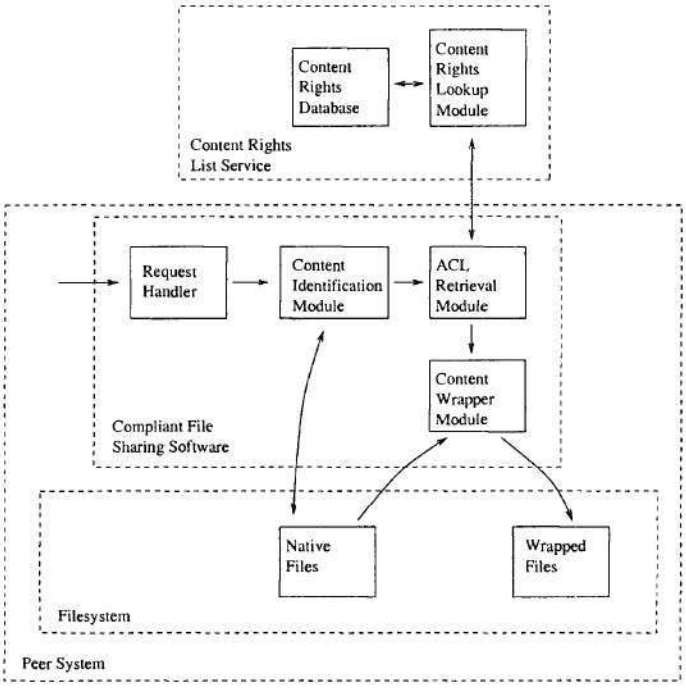


Figure 31: Content importation.

Figure 31 shows this process. The first step, content identification, is performed by a content identification module in the CFSS. The identity information is then passed to the ACL retrieval module which sends it in a query to the content rights service. The service responds with the protection label and the encryption keys for the content. This information is passed to the content wrapper module that creates the content container by encrypting the content with the supplied key and attaching the protection label. The CFSS does not store the encryption key. To access the content, a user must possess the appropriate token and decryption key.

5.6.4.2 Content Identification Process

In this section, we discuss possible methods of content identification and explain why some are ideal candidates for CITADEL and some are not.

In order to protect content with the correct access rights, it is important to accurately identify content as it is imported into the system. Straightforward approaches such as identifying content by the actual file name have been used by content protection systems in the past and proved to be vulnerable. Many applications support the identification of content using metadata, descriptive information about the content such as artist, title, and publisher [145, 3, 206], that is usually stored as a header in the content file [105, 151]. Due to the ability of the user to alter this information, this is not sufficient for our system.

One method to provide metadata in a form that is not susceptible to alteration by users is to embed the information as a watermark. Watermarking is the embedding of some identifying information into the content in such a manner that it can not be removed by the user but it can be extracted or read by the appropriate party. Use of this technology would allow the content identification to be securely embedded by the content provider and read by the CFSS. Content creators could embed watermarks in all content that is created and distributed via any means. One problem with this approach is that there are millions of content files already in distribution that are not watermarked.

A different approach is to not rely on metadata or watermarks to provide the information, but to have the system determine the identity of the content. The system examines the file to extract identifying properties and then determines the actual identity of the content by comparing the properties with a database of all known content files. We discuss this further in the next subsection. This is achieved using a content-based identification algorithm. Content-based identification algorithms analyze the perceptual qualities of the content to derive a fingerprint of the content. A number of such algorithms have been proposed for audio files; see for example [75, 208]. These are different from cryptographic signatures such as MD5 [175] and SHA-1 [7] that examine the bits of the file and can not correlate the same content in two different formats or even different qualities of the same format.

5.6.4.3 Content Importation Process

The content importation process is independent of the identification algorithm that is used. Recall that the two types of content that are imported into the environment are: existing content and new content. We first describe how the system is used to import additional copies of existing content. We then describe the additional steps needed to import new content.

The first step in content importation is the content identification module executing the identification algorithm on the content file and producing a set, s, containing n identification properties, i_1 to i_n. The ACL retrieval modules sends a query containing s to the content rights service. Upon receiving s, the content rights lookup module uses each i_n to search the content rights database for a match to a known content entry. The matching process need not result in a binary decision, but can result in a match and a *certainty value*, c, associated with the match. If c is above the defined *certainty threshold* t then the match is considered valid and the content rights service sends a response containing the rights for the content file.

To import new content, the process involves a few additional steps. The user importing the content specifies the access control policy, p, including rights keys and the content encryption key, K_R. The user's CFSS performs content identification and sends (s, p, K_R) to the content rights service in a content importation request. The content rights lookup module first searches the database for the content using combinations of s. Upon a failed lookup, a new entry is added for (s, p, K_R).

Essential to the effectiveness of this system is the establishment of a complete database and maintaining the database by adding new content before it is made available to the public. Although this may seem like a daunting task, we point to some current efforts that are a creating catalogs of all available music content such as CDDB [55] and Loudeye [54]. Alone, these efforts have made significant progress and it is reasonable to suggest that a content database to support content protection will have considerable support from content providers.

Goal	Single User Attack	Collusion Attack
Protected Distribution	Forge Access Token Replay Access Token	Bypass Authorization Checks Re-distribute Tokens Alter ACL
Protected Storage	Determine K_R Determine K_{CFSS} Alter ACL	Re-distribute Tokens
Output Protection	Obtain K_{RTS_ID} Obtain Plaintext Content	Re-distribute Content

Table 3: Summary of attacks on protection goals

5.7 Analysis

In this section, we discuss how the system maintains the level of assurance relative to its protection goals by being robust against compromise or circumvention. As in any protection system, we assume the presence of malicious users that aim to gain unauthorized access to content.

5.7.1 Threat Analysis

There are different levels of threats to content protection ranging from the casual user to the hobbyist/hacker to the professional pirate [157]. These levels somewhat parallel the following taxonomy of attackers described by the Abraham, et al. [12]: class I (clever outsiders), class II (knowledgeable insiders), and class III (funded organizations). The generally accepted practice in the content distribution industry is that content protection technology is most effective for low to mid-level threats from casual users and hobbyists/hackers while legal protection is most effective for higher level attackers such as professional pirates. The goal of commercial content protection is to "stop unauthorized, casual copying of commercial entertainment content" [157]. This has also been phrased as "keeping honest people honest" [27]. These are somewhat modest goals compared to the protection goals in many military and financial applications.

Table 3 shows the protection goals of the system and possible attacks on these

goals. We look at attacks by an individual user as well as attacks by a collusion. Recall that CITADEL aims to provide protected distribution and protected storage. We briefly mention extensions to CITADEL that can provide output protection. Some of the attacks listed in the table provide no benefit to an attacker and some of the attacks are sufficiently complex due to fundamental properties of cryptography. For example, without an appropriate access token, determining K_R can only be done by brute force attack. For these reasons, we do not detail each of the attacks in this section, but discuss the less obvious ones.

We assume a hobbyist/hacker end-user that has appropriate knowledge and resources. This user has full access to and control of the CFSS and the operating system. Since users have access to the CFSS on their PC, it will be a target for attackers seeking to circumvent the protection of the system. We will show that even without software protection or tamper resistance, attackers can not defeat the CFSS to obtain unauthorized access to content. Let us assume that the attacker reverse engineers the CFSS or otherwise fully compromises the CFSS. We analyze the risk associated with each attack. We show that the damage is limited and does not defeat our protection goals.

We further discuss three of the potential threats:

1. Determine K_{CFSS}: An attacker may obtain the decryption key, K_{CFSS}, that is used to access content containers.

2. Bypass Authorization Checks: An attacker may cause the authorization checks to be bypassed so that the CFSS will allow unauthorized peers to download content.

3. Alter ACL: An attacker may attempt to remove a protection label and replace it with a protection label requiring lesser or no rights.

We now analyze the actual vulnerability associated with each of the above potential attacks.

Determine K_{CFSS} Attack: If an attacker reverse engineers the CFSS to obtain the decryption key, he only obtains the key used to access the protection label. The actual content keys are encrypted with key for the rights role. Therefore, even with the CFSS's decryption key, an attacker can not directly access the content file. The attacker could attempt to alter the ACL in the content object, however, it is digitally signed by the content rights list server, so any alterations will be noticed by the CFSS and the content container will be rejected.

An attacker that legitimately obtains a content token and rights key, K_{-RTS_ID}, can use the compromised K_{CFSS} and K_{-RTS_ID} to obtain K_R and consequently obtain the decrypted content. However, since the user legitimately obtained a content token, the user has the rights to access the content. As stated in section 5, CITADEL does not provide output protection. In general, once access rights are presented, the user is able to access the content in the content container. We discuss output protection further in subsequent sections.

Bypass Authorization Checks Attack: If an attacker is able to alter the operation of the CFSS to cause it to bypass authorization checks, then the CFSS will allow unauthorized peers to download content containers. However, the content containers can not be accessed at the receiving peer without the access token and key because the CFSS requires the token to access the content container and the rights decryption key to access the encrypted content.

Alter ACL Attack: If an attacker removes a protection label from a content container and replaces it with a protection label that requires lesser rights, then this will not work for a number of reasons. Let us assume that a user creates a content container that includes content X and a protection label for content Y. Assume the user attempts to access this content with only the access token and key for Y. The user can use the content token for Y, but the K_R retrieved from the protection label will not match the key used to encrypt content X. Therefore the user can not access content X with only the rights for Y.

5.7.2 Additional Robustness using Software Protection

We have shown that the system's design includes protection mechanisms that are robust even in the face of a completely compromised CFSS. We now point out that there are mechanisms that can be used to make it more difficult for an attacker to even compromise the CFSS. The CFSS can be considered to be a *mobile agent*, a program that is executed on an untrusted computing base. The authors of [96] discuss a number of attacks that malicious users can perform against mobile agents including spying out code and data (i.e. the CFSS's decryption key) and manipulation of code and data. One method that can be used to guard against these attacks is mobile cryptography [180]. This involves executing encrypted functions to guarantee code privacy and code integrity. Another method is time limited black box protection [97] that can protect the code and data of a mobile agent from being read or modified for at least some minimal time interval. Although there is work, such as [16], that discusses the inability of these approaches to provide long-term security against high-level attacks, it is understood that such approaches do protect against casual attackers and also increase the cost of an attack by capable attackers. A deployed implementation should employ these techniques to heighten the security of the application; however, as we have shown, the security of CITADEL does not rely on the protection of the CFSS.

5.7.3 Related Protection Systems

Protected distribution work takes many forms differing greatly depending on the distribution method. In cable and satellite, conditional access is provided by set-top boxes enforcing subscription and pay-per-view models; see for example [149]. In CD and DVD sales, conditional access simply means that the person that pays for the content receives the media containing the content. On websites that sell content, protected distribution is performed in the client-server model of purchasing rights and obtaining content. In multicast or group communications, protected

distribution is provided by using group keys to access encrypted content [35] and by controlling access to the multicast distribution tree [110].

Protected storage work includes Content Protection for Pre-recorded Media (CPPM) [8] for protection pre-recorded DVD-Audio, Content Protection for Recordable Media (CPRM) [9] for protecting content stored on recordable media such as DVD-R or flash memory, Content Scrambling System(CSS) [10] for protecting pre-recorded DVD-Video, and copy-protected CD solutions such as [141] for protecting pre-recorded CDs from replication or extraction to files such as MP3s.

5.7.4 Output Protection

As stated previously, CITADEL has the primary goal of providing protected distribution and storage. It is possible to add components to CITADEL to provide output protection. Some of this requires integration with previous work in output protection for devices. Other mechanisms can be leveraged to help provide output protection and protection against some collusion attacks such as redistribution of content or access tokens.

Fingerprinting provides an effective deterrence against redistribution of content [203, 17]. Such systems can be integrated into the content importation system or the compliant file-sharing software. Other systems such as digital signets [63] and traitor tracing [44] can be used to discourage redistribution of access tokens.

Other output protection work includes Digital Transmission Content Protection(DTCP) [11] for protecting content during transmission between devices using IEEE 1394 or Universal Serial Bus (USB), Macrovision Copy Protection [132], High-bandwidth Digital Content Protection(HDCP) [59] for protecting content during transmission to digital displays, and Microsoft's Secure Audio Path [140] for protecting content on PCs during transmission to audio devices such as sound cards.

5.8 Discussion

Recent work in peer-to-peer systems have proposed different types of file retrieval and file location systems. In this section, we discuss how CITADEL is affected by such systems.

5.8.1 Interaction with different types of file retrieval

Most peer-to-peer systems allow content to be exchanged directly between peers and require that content is stored only at the peer that is providing or sharing that content. However, there are systems in which this is not the case and this can cause problems with some content protection approaches. In some anonymous peer-to-peer systems such as Freenet and APFS, files are delivered through the peer-to-peer network in order to mask the identity of the responder and provide some caching. In some peer-to-peer systems including some that are based on distributed hash tables, peers' files are replicated and cached at other peers. This can be used to provide cooperative mirroring, time shared storage, or increased scalability.

Some content protection approaches assume that the only hosts that will have access to content is the sharing peer and the downloading peer. Therefore, these content protection systems only enforce the policy upon file request, but do not protect the file during download. This leads to vulnerabilities where unauthorized users can gain access to files based on their location in the peer-to-peer network. Furthermore, malicious users can aggressively participate in cooperative mirroring and time shared storage in order to gain access to more content. The design of CITADEL accounts for these systems that require other peers to have access to the content file. In addition to performing authorization upon file request, CITADEL also stores the file in a secure content container that can only be accessed by the authorized users. Thus, even if other peers must cache or forward the file, there is no vulnerability since only authorized users can access the actual content within the content container.

Another variation of peer-to-peer systems that can cause problems for some content protection system designs is parallel downloading. Parallel downloading allow a host to retrieve different parts of a content file from different peers simultaneously. There are two properties of parallel downloading that may cause issues: 1) for a single file retrieval, there are multiple file requests involved and 2) software at the retrieving peer creates a file that is a combination of parts from multiple files. In CITADEL, parallel downloads are supported since authorization will be performed for each file request and the retrieved parts will be combined to form a secure content container.

5.8.2 Interaction with different types of file location

We have described an approach in which content protection is transparent to the file location system and described how this assists portability and ease of implementation. However, we also believe that in some situations gains in efficiency and usability can be achieved by the location system and protection system having knowledge of each other. For example, improved usability and efficiency can be achieved by restricting query results to items that the user is authorized to access.

In systems such as Gnutella where query results only come from peers that have access to the content and the access control policy this is straightforward. In distributed hash table-based systems, the solution is more involved. Since many of the lookup operations in these systems may be performed by entities that do not possess the content or the access control policy, they are not able to restrict the query results based on the access control policy. In these systems, it is important to be able to restrict queries based on authorizations because of the resources that can be wasted otherwise. If queries are not restricted at the first hop of the lookup process, then this query can possibly traverse a number of nodes wasting computation and bandwidth. The following questions are potential paths of future work:

1. How can decentralized hash table lookup systems be made aware of the access

control policy?

2. What are the design details of a system that restricts the hosts' ability to lookup based on an access control policy?

3. How would such a system assist in making the lookup system more robust to other types of attacks such as denial-of-service?

5.9 Conclusions

Predicting how peer-to-peer systems will evolve is difficult at best. What has become clear, however, is that content protection will play an increasingly important role in the success of such systems. The challenge has been how to design a system that provides adequate content protection and yet maintains the openness of the peer-to-peer model. In this work we explained the need for content protection in peer-to-peer networks, argued that such functionality should be provided as an overlay security layer, and proposed CITADEL as an example of such an architecture.

Chapter 6

CITADEL Evaluation and Implementation

In this chapter, we further discuss, evaluate and define the CITADEL architecture from a real world viewpoint. Now that the architecture has been motivated and defined, we examine the feasibility of the CITADEL as a real-world, content protection architecture. We address three issues:

1. Can CITADEL be used to provide common content distribution business models?

2. What is the overhead of CITADEL in terms of support infrastructure and at the peers?

3. How feasible is the CITADEL architecture in terms of the ability to implement the components and its ability to be used with common peer-to-peer networks?

This chapter is organized as follows. Section 6.1 describes how content protection can allow common business models to be supported in a peer-to-peer environment. Section 6.2 evaluates the costs of CITADEL in terms of message overhead and computation overhead. Section 6.3 describes our implementation of a working prototype of CITADEL. We conclude in section 6.4.

6.1 Supporting Content Distribution Business Models

There are a number of business and service models that are popular in other content distribution methods that peer-to-peer systems have been unable to offer due to the lack of the protection functionality required to support such models. We identify three common content distribution business models and show how each of these can be supported in a peer-to-peer distribution system by providing the appropriate underlying content protection functionality. We then show how CITADEL supports this additional content protection functionality.

6.1.1 Common Business Models

1. **Pay-per-view and Subscription model**

   Description: The goal of this model is to allow users to subscribe to specific content or set of content. This model follows the subscription by users to particular channels or individual pay-per-view items. Current peer-to-peer content protection systems can only control whether or not a file is allowed in the system [122]; they cannot restrict access to files to certain users. Our system supports this model and allows a user to purchase access to specific content or groups of content. For example, in a music sharing system, music from each record label could require a separate subscription. Another example supported by our model is where users can purchase access to individual songs.

   Underlying access control functionality: To support this model, our system provides *subject-based access rights* such as user-based or group-based access rights rather than only the default world-based access rights provided in current systems [122].

2. **Syndication model**

Description: The goal of this model is to only allow certain users to purchase the right to redistribute content. This model aims to mimic the distribution of syndicated television shows and pay-for-content services such as the services used by retail stores to provide background music. In current peer-to-peer systems, there is no way to specify different types of access rights for a file; thus, all users have all possible access rights for every file. Our system supports this model by allowing an access control policy that specifies multiple types of access rights. Our system supports this model and allows a user to be granted a subset of the possible access rights. For example, a user could have the right to download a file, but not to share it.

Underlying access control functionality: To support this model, our system is able to enforce *multiple types of access rights.* In a peer-to-peer system, the different access rights include share, download, and export. Additionally, each type of access right can have parameters in addition to "allow" and "disallow". For instance, share rights can specify the number of times the file is allowed to be distributed.

3. **Reseller model**

*Description:*The goal of this model is to allow certain users to act as resellers and redistribute content for a fee. This model aims to mimic the distribution of CDs and video via stores. Our system supports this model by building upon the syndication model and adding the ability to have reseller peers perform authorization of downloading peers.

Underlying access control functionality: To support this model, our system provides *delegated authorization.* That is, end users are able to perform authorization for peers attempting to download content.

6.1.2 CITADEL's Support of Business Models

The base CITADEL system, as described in previous sections, provides the subject-based access rights necessary to support the pay-per-view and subscription model.

To support the syndication model, separate download and upload access rights are specified in the access control policy and in the resulting access tokens. The examples in previous sections depicted an environment that controls a peer's ability to download certain content. We now describe how the system can support another approach that involves controlling a peer's ability to upload or share certain content.

To enforce share rights with CITADEL, for each content container to be shared, the user must possess an access token that specifies share rights for that content. If the necessary rights are not provided for certain content, then the CFSS will not allow sharing of this content container by either not responding to file location requests or file retrieval requests for that content.

To support parameterized access rights such as share limits and time-restricted downloads with CITADEL, a new field for each parameter is added to the ACL in the content container or in the access token. For example, to control share limits, a field is added to the ACL in the access token that specifies the number of authorized shares. This field is initialized when the access token is received and decremented at each file download. This portion of the access token is not be signed by the CTS, but is edited and signed by the CFSS.

To provide delegated authorization to support the reseller model, we identify two approaches: 1) Tokens are obtained from the content provider and submitted to the reseller for verification when requesting the content. 2) Tokens are issued by the reseller. CITADEL inherently supports the first approach. To support the second approach, the reseller is issued a certificate by the content provider and the reseller issues access tokens digitally signed with its private key. This allows tokens issued by the reseller to be verified or traced back to the original content provider. If necessary, a payment system can be integrated that allows the reseller to handle

payments.

6.2 Evaluation

We evaluate the performance of CITADEL relative to the performance of the distributed ACL and queried ACL approaches to decentralized content protection that we discussed in Section 5.5. We analyze the costs of operations in each system and then present the results of a simulation based on these costs.

6.2.1 Analysis

We examine the costs associated with file exchange and content importation in CITADEL, the distributed ACL system, and the queried ACL system. Table 4 shows the definitions of the variables used in the analysis.

Tables 5, 6 and 7 show computation and message costs for file exchanges at the downloader, the uploader, and the ACS. For example, the table shows the computation cost at the uploader in CITADEL is $(v + cl)x$. This shows the digital signature verification and content container ACL lookup for each file that is uploaded.

ACS computational costs are significantly greater in the queried system than in the other two systems and slightly more in CITADEL than in the distributed system. This is because in the queried system, the ACS must perform the operations for every file exchange, while it must only perform such operations at most once per user in the other systems. The cost is more in CITADEL than in the distributed system because CITADEL authenticates the host, looks up the rights and digitally signs the token for each user while the distributed system creates a single digitally signed ACL for all users.

Computational costs at the uploader and downloader are interesting because one may hypothesize that due to the computation required by the CFSS of CITADEL the computational load and associated processing time may introduce some service delay. However, as shown in Tables 5, 6, and 7, the computational costs at the

peers in all three systems are similar since all systems require the downloader to sign the request and the uploader to verify the message from the downloader or ACS.

Message costs are significantly greater in the distributed system than in the other two due to the distribution of the ACL. Even with compression, the ACL can be quite a large file due to the number of entries in the list. Therefore, message costs in the distributed system will remain higher since the ACL will be large relative to the size of the token that is sent by CITADEL. Message costs in the queried system are also higher than in CITADEL because message volume is proportionate to the number of files exchanged since the ACS and uploader communicate on every file exchange.

For each content importation, the user that imports the content is referred to as the *importer*. Tables 8 , 9 and 10 show computation and message costs associated with a single content importation at the importer, the ACS, and all other peers. For example, the table shows that the message cost at the importer in the queried and distributed ACL systems is CID and similarly CS in CITADEL. This is due to the message costs of transmitting the content identification to the the ACS for each content object that is imported.

Content importation involves computation cost at the CTS or CRLS for all three schemes since each requires some lookup of the content identification. The distributed system also involves an ACL update sent to all peers; thus, the computational and message costs at the ACS are higher in the distributed system. Also due to the ACL update, all peers in a distributed ACL system must verify the signature on the ACL update; thus, the distributed system is the only one that requires some computation by all peers for a content importation. An implementation could somewhat reduce this cost by batching a number of ACL updates into a single ACL update message depending on the frequency of content importations. CITADEL requires a signature extraction at the importer; likewise, the other schemes require some form of content identification at the importers as well.

System Parameters	
n	= number of peers
x	= number of files exchanged by each peer
Message Transmission Costs	
T	= token
ACL	= access control list
ACL_UP	= access control list update
AR	= authorization request
AA	= authorization acknowledgment
FR	= file request
F	= content file
CS	= content signature tuple (s, r, k)
CID	= CID tuple (CID, r)
SM	= status message
Computational Costs	
ds	= digital signature
v	= signature verification
l	= ACL lookup
cl	= content container ACL lookup
cse	= content signature extraction
csl	= content signature lookup
$cidl$	= CID lookup

Table 4: Definition of Variables Used in Analysis

CITADEL			
Type of Cost	File Exchange Costs		
	downloader	uploader	CTS
computation	(ds)	$(2v + cl)x$	$(v + l + ds)n$
message	$(FR + T)x + AR$	$(F)x$	$(T)n$

Table 5: Cost of file exchange in CITADEL

6.2.2 Simulation Results

To better examine the file exchange costs that we have discussed, we provide simulation results that show the performance of the three systems. We use a simulated peer-to-peer file sharing system based on a model that has the following parameters:

1. The pool of potential peers has n peers. (We show results for varying values of n.)

Distributed System			
Type of Cost	File Exchange Costs		
	downloader	uploader	CTS
computation	$(ds)x$	$(v+l)x$	(ds)
message	$(FR)x$	$(F)x$	$(ACL)n$

Table 6: Cost of file exchange in the Distributed System

Queried System			
Type of Cost	File Exchange Costs		
	downloader	uploader	CTS
computation	$(ds)x$	$(v)x$	$(v+l+ds)(x)n$
message	$(FR)x$	$(F+AR)x$	$(AA)(x)n$

Table 7: Cost of file exchange in the Queried System

2. An active phase refers to time in which the peer is connected to the peer-to-peer system and an inactive phase refers to time in which the peer is not connected to the peer-to-peer system. The length of an individual *active* phase is an exponential distribution with an average of τ. The length of an individual *inactive* phase is an exponential distribution with an average of 23τ. The ratio of active to inactive duration is 1 : 23, so the average group size, g is approximately $n/24$ during steady state. (This is proportionate to 1 hour a day.)

3. The length of the group session is 168τ. (This provides a session length that is proportionate to 1 week. The figures provide information in terms of a 24τ period within the session.)

4. The number of files downloaded by a peer in an active phase is a discrete value derived from an exponential distribution with an average of x. (We show results for varying values of x.)

5. The file size is 3.7 MB.

Many of the parameters of our model are derived from the results of a measurement study of two large peer-to-peer file sharing systems [184]. For example,

CITADEL			
Type of Cost	Content Importation Costs		
	importer	CTS	other peers
computation	*cse*	*csl*	0
message	CS	SM	0

Table 8: Cost of content importation in CITADEL

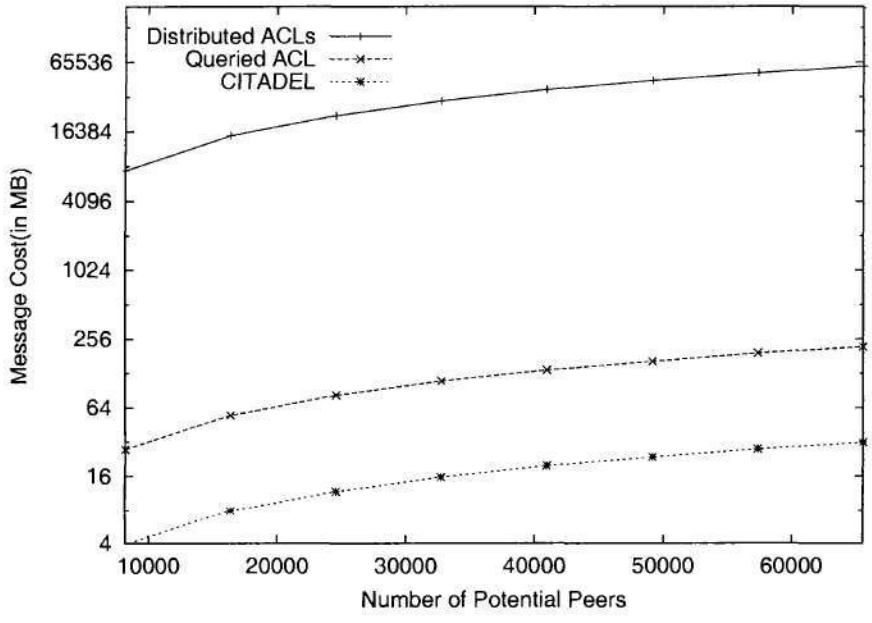
Distributed System			
Type of Cost	Content Importation Costs		
	importer	CTS	other peers
computation	0	$cidl + ds$	v
message	CID	$SM + (ACL\_UP)n$	0

Table 9: Cost of content importation in the Distributed System

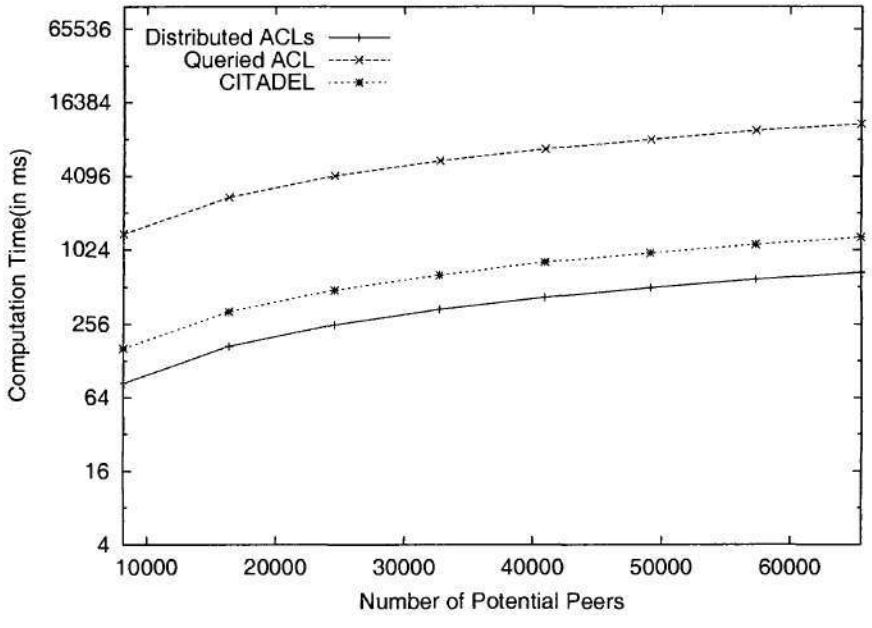
the authors of the study determined that the median session duration is about one hour, thus our active to inactive ratio. The authors determined the average size of a shared file is 3.7 MB. Our inactive phase includes the active and offline states described in [184].

Our simulation results only examine file exchange costs and not content importation costs. There are two reasons for this. First, as explained in the analysis section, the costs of file exchange in the systems vary significantly and we aim to better understand the magnitude of difference. However, the costs of content importation in the systems are fairly close; therefore we feel that the importance of the magnitude of the difference is somewhat lessened. Secondly, we suggest that in most environments file exchanges far outnumber content importations. Thus, the cost of file exchanges is a more important measure.

The simulation calculates the computation load by evaluating the processing time involved with cryptographic operations. The values for processing time for the cryptographic operations are shown in table 11 and are based on benchmarks published for the publicly available Crypto++ library [53]. The simulation used 128-bit Triple DES encryption, MD5 message digest, and RSA 1024-bit digital signatures.

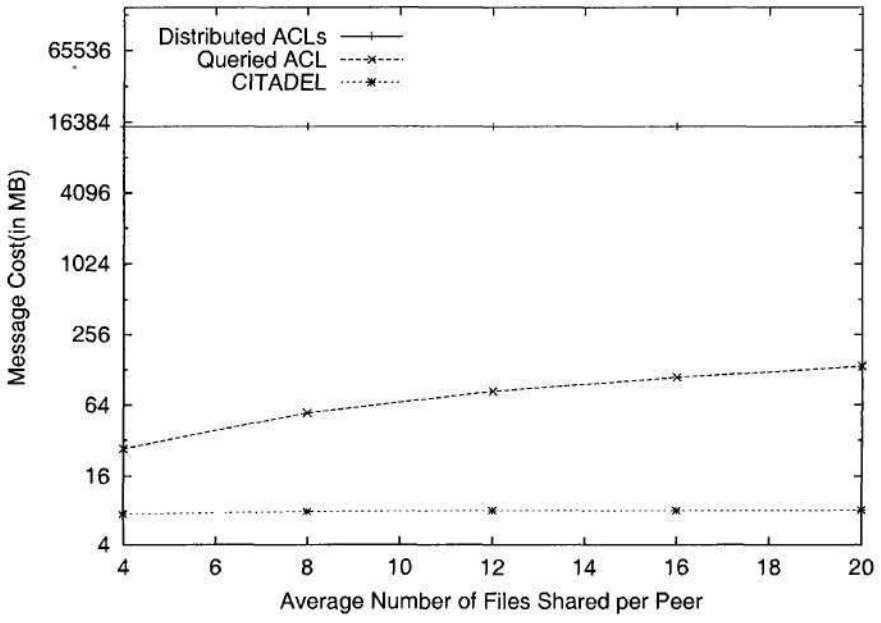


(a) Message Overhead

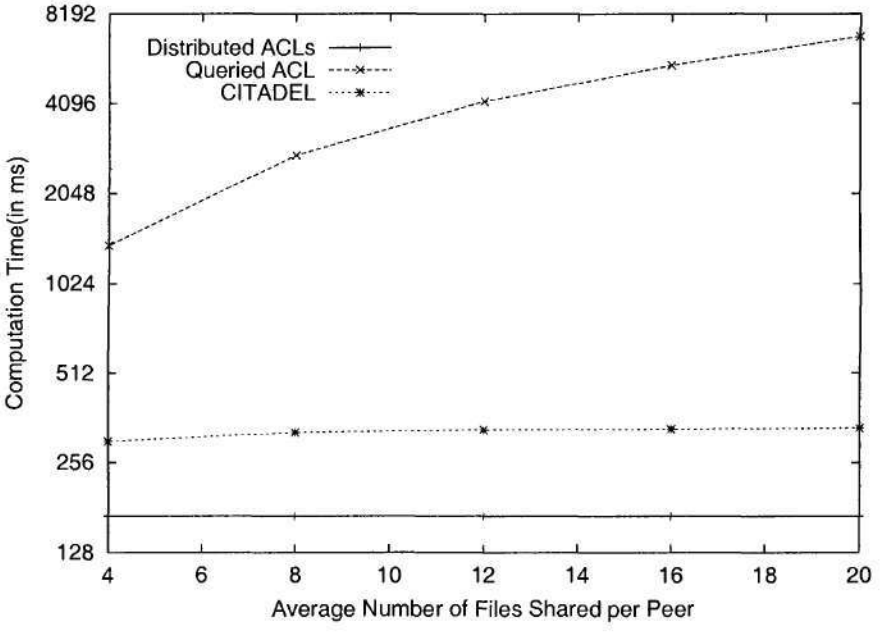


(b) Computation Overhead

Figure 32: Overhead as a function of the number of potential peers



(a) Message Overhead



(b) Computation Overhead

Figure 33: Overhead as a function of the number of files shared by peers

Queried System			
Type of Cost	Content Importation Costs		
	importer	CTS	other peers
computation	0	*cidl*	0
message	CID	SM	0

Table 10: Cost of content importation in the Queried System

Operation	Performance
3DES encryption	4.748 MB/sec
MD5 message digest	100.738 MB/sec
HMAC/MD5 message digest	99.863 MB/sec
RSA 1024 signature	10.29 sec
RSA 1024 verification	0.30 sec

Table 11: Cryptographic computation processing time

Figure 32 shows the message overhead and computation overhead at the ACS as a function of the number of potential peers. Figure 32(a) shows that the message overhead of the distributed ACL system is almost three orders of magnitude greater than the other systems for small group sizes and continues to increase with the number of peers. With $65,536$ potential peers, the message overhead of the distributed ACL system is $60,213.1MB$ compared to $219.3MB$ in the queried ACL system and $31.6MB$ in CITADEL. Figure 32(b) shows that the computational overhead in the queried ACL system increases with the number of peers and becomes an order of magnitude greater than the other systems. For a system with $6,5536$ peers, the computation overhead of the queried ACL system is $10,979.29ms$ compared to $1294.6ms$ in CITADEL and $674.4ms$ in the distributed ACL system. This shows that as the number of peers in the system increases, the queried ACL and distributed ACL approach have difficulties scaling due to computation costs and message costs, respectively.

Figure 33 shows the message and computation overhead at the ACS as a function of the average number of files shared per peer. For this particular simulation, the number of potential peers was 16384. Figure 33(b) shows that the computation overhead in the queried ACL system increases with the number of files shared in the

system. The computation overhead of the other two systems scales better with an increase in the number of files shared. With peers sharing an average of 4 files each, the computation overhead is 1371.9ms in the queried system, 301.8ms in CITADEL, and 168.59ms in the distributed ACL system,. However, as the average number of shared files increases to 20, the computation overhead of the queried system increases fivefold to 6877.0ms while the others remain approximately the same. Figure 33(a) shows that the message overhead of CITADEL maintains around 7.8MB and the queried ACL system does increase fivefold from 27.3MB to 137.4MB, but they both remain a couple of orders of magnitude less than the 15056.9MB of message overhead in the distributed ACL system.

6.3 Practical Implementation

We have implemented a CITADEL prototype using open source components including the Gnutella network as the distribution layer [78] and the LimeWire graphical user interface-based Gnutella client [129] as the filesharing software. One of the key goals of the implementation efforts was to show that the CITADEL architecture is a realistic and lightweight approach and that it can be implemented as part of any popular file-sharing network. LimeWire is written in Java and runs on multiple platforms including flavors of Windows and UNIX. Our prototype was developed on Solaris machines and also runs on Linux.

Figure 34 depicts the CITADEL prototype. The implementation consisted of creating a CFSS by providing download authorization and upload authorization modules and creating a content insertion and exportation system including the content wrapper and unwrapper. Rather than implement custom authorization protocols and content container formats, we aimed to use standard security protocols to perform these operations. Overall, our implementation involved creating authorization modules that were added to the LimeWire software by modifying two LimeWire modules to call our libraries and providing a program that allows content insertion

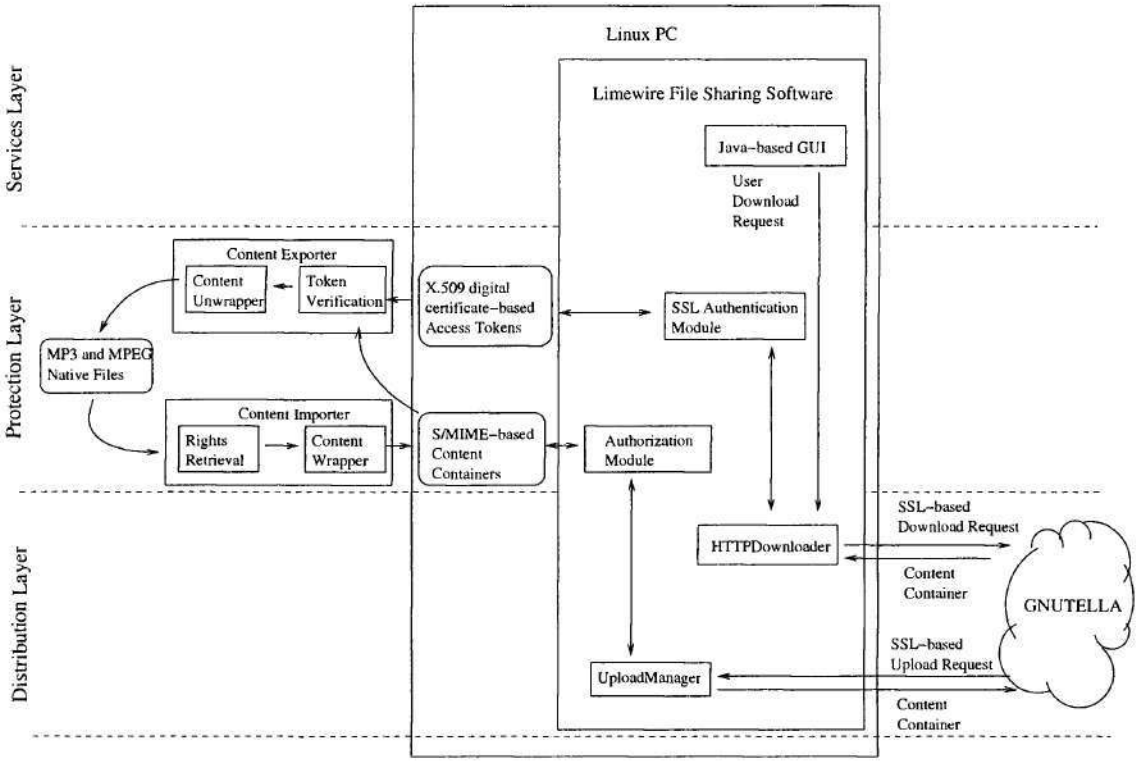


Figure 34: CITADEL implementation

and exportation.

6.3.1 Authorization

Our design does authorization based on standard X.509 version 3 public-key certificates [102] just like those used for web server authentication. The identity in the public-key certificate directly specifies the role to which the holder is assigned. This allows standard X.509 version 3 certificates to be used as the access token without any new extensions. Alternatively, there have been proposals for extensions to X.509 certificates to allow the identity to specify the user and have new attributes that specify the roles assigned to the user [153].

Our system currently provides hierarchical RBAC by creating leaf roles for each content object and forming a role hierarchy. Membership in any senior role can be shown by authenticating with the public-key certificate for that role. Instead of

specifying the role hierarchy in the ACL of the content object, we create a certificate issuance hierarchy that reflects the role hierarchy. A role token also includes the trusted certification path from the content's role certificate to the token's role certificate. This allows the uploader to perform authorization by authenticating the downloader then verifying the certificate chain.

We use the private key corresponding to the content role's public-key certificate as the content decryption key, K_{-RTS_ID}. To perform authentication, we use the strong client authentication that is part of the SSL/TLS protocol. This allows the client to be cryptographically authenticated by the server based on the client's public-key certificate. In the Gnutella protocol, a file download request is sent as an HTTP GET request. Instead, in our system we use HTTPS [174], HTTP over a SSL connection. This allows the uploader to utilize the SSL protocol to authenticate the downloader upon receiving the GET request. The SSL protocol at the uploader verifies the validity of the certificate and that the client possesses the corresponding private key. The application then performs an authorization lookup based on the authenticated identity. In our implementation, the authorization table is stored in memory and is initialized at start-up by retrieving the rights from each shared file. Since in the real world, users will share a large number of files, this authorization table should cache only a certain amount and then lookup other files as necessary.

6.3.2 Content Containers

Content containers are created using S/MIME (Secure/Multipurpose Internet Mail Extension) [169, 168]. Although S/MIME was designed for securing Internet mail, we feel that it provides a simple and elegant example of content containers. MIME [73] supports virtually any content type including MPEG, JPEG, GIF, and generic binary application data. We create the encrypted content file using the S/MIME envelope functionality. This envelope function creates a MIME entity by first encrypting the content data with a triple DES symmetric session key, K_R and encrypting the

session key with a public key. In our system, this public key is K_{+RTS_ID}, the public key of the role identity that is authorized to access this content. The MIME entity consists of the encrypted content and a header containing the sender's public-key certificate, an identifier of the algorithm used to encrypt the session key, and the encrypted session key. Now that we have the encrypted content file and K_R, we add the content rights list to complete the protection label of the content container. In this design, we only need to specify the role identity and the rights. We achieve this by attaching the public-key certificate of the role identity to the protection label. To perform this, we create a new multipart MIME entity that includes a part containing the public key certificate of the role identity and a part that contains the S/MIME envelope. So after one S/MIME and one MIME operation, we have a content container. For the content wrapper and unwrapper, we wrote shell script programs that perform the content rights lookup, token verification, MIME operations, and uses OpenSSL libraries [152] for S/MIME operations. The CFSS creates the authorization lookup table using OpenSSL commands to retrieve the public-key certificate from the protection label of each content container.

Also, since in our implementation, all authorized users for a certain content object had the same set of rights on the object, we were able to only specify the identities in the protection label and not the access rights. Our implementation can be extended to support different access rights by using S/MIME security labels [95] to create the protection label. Our implementation can also support flat RBAC and its need for multiple content rights list entries in the protection label. This can be achieved because S/MIME envelopes allow a single entity to be encrypted with different decryption keys for a set of users. S/MIME does this by encrypting the entity with a session key then generating a different encrypted version of the session key for each user similarly to how we proposed.

6.4 Conclusions

One of our central premises is that content protection should be viewed as an opportunity to enable new service functionality and not a nuisance that designers have to deal with. This layered approach to content protection and the focus on providing a flexible framework allows CITADEL to perform the role that we envision for it. This point is illustrated in our work through the proposal and description of services and business models that can be deployed on top of the CITADEL architecture. Our work also considers the performance of CITADEL relative to other approaches that can be used in decentralized systems. Our work also includes the description of a prototype that shows a practical implementation and validates that CITADEL is a realistic and lightweight approach to create a protected peer-to-peer file sharing environment.

Chapter 7

Conclusions and Future Work

This thesis describes research in securing large-scale content distribution systems. It addresses the security issues that are introduced by emerging content distribution paradigms. It addresses the need for authentication and authorization, data protection, and system security in such systems. Here we summarize the specific contributions made in this research.

In multicast and content distribution network environments, there was a need for a scalable system to provide theft deterrence. Chapter 3 describes the development of WHIM, a system that uses a hierarchy of intermediaries to fingerprint multicast multimedia content. We show the benefits of this approach as compared to previous approaches that were implemented at the network-layer or within the application. This work also proposed distributed and real-time watermarking of multimedia content. We described a prototype of the system that validated the feasibility of these techniques.

Chapter 4 describes the GOTHIC group access control architecture that provides a solution to the secure IGMP problem in multicast and the secure anycast server advertisement problem. This work generalized the two problems into a problem of group access control and proposed a generic scalable architecture for Internet group access control. We specify how the system can be used in a range of environments including different flavors of multicast and anycast. This work also considers issues in group policy. We identify the problem of group owner determination and authentication and propose two solutions. Within this work we also propose the possibility of group access control-aware group key management and provide results that show

the performance improvements that it can achieve.

In the area of content protection in peer-to-peer systems, we propose the CITADEL architecture that is presented in Chapter 5. We discuss the benefits of content protection in peer-to-peer systems and define the objectives of such systems. We show that the objectives of end-users and content providers conflict and propose a system that is able to achieve both sets of objectives. We describe the details of the content protection system including the content containers, the content importation system, and the access control model that is used to maintain manageability. In chapter 6, we explain that content protection can enable the support of common content distribution service models. We provide an analysis and simulation results to show the costs of providing content protection. We also describe an implementation of the CITADEL architecture that shows how the system can be implemented as a lightweight addition to current peer-to-peer systems.

7.1 Future Work

We provide a discussion of possible directions for future work based on this research.

- In Chapter 4 we described the performance improvements that could be gained by a group access control aware group key management system. We proposed three approaches to providing topology information. A potential area of future work is the further definition and evaluation of these approaches. The evaluation must consider not only the costs of providing the topology information, but also the varying performance improvements based on the trusted subtree topology and the placement of receivers. One issue is the placement of trusted routers for performance optimization. Further optimization can be achieved based on the grouping of receivers within the group key management system based on the receivers location in the trusted subtree topology.

- The CITADEL architecture as described in Chapter 5 is transparent to the

file location system and we described how this assists portability and ease of implementation. However, we also believe that in some situations gains in efficiency and usability can be achieved by the location system and protection system having knowledge of each other. For example, improved usability and efficiency can be achieved by restricting query results to items that the user is authorized to access.

In systems such as Gnutella where query results only come from peers that have access to the content and the access control policy this is straightforward. In distributed hash table-based systems, the solution is more involved. Since many of the lookup operations in these systems may be performed by entities that do not possess the content or the access control policy, they are not able to restrict the query results based on the access control policy. In these systems, it is important to be able to restrict queries based on authorizations because of the resources that can be wasted otherwise. If queries are not restricted at the first hop of the lookup process, then this query can possibly traverse a number of nodes wasting computation and bandwidth. Further research could design a system that restricts a host's ability to perform lookups based on an access control policy. This would involve detailing how decentralized hash design a system that restricts a host's ability to perform lookups based on an access control policy. This would involve detailing how decentralized hash table lookup systems can be made aware of the access control policy. Additionally, one could consider how such a system could assist in making the lookup system more robust to other types of attacks such as denial-of-service.

- Currently, if a single content distribution system needs some combination of security services such as encryption, fingerprinting, and content protection, it must utilize three different security systems. This requires the creation and maintenance of multiple security policies. Our work has taken steps toward this. For example, In GOTHIC, we showed how group key management and

group access control can benefit from each other. WHIM showed that data encryption, distribution, and watermarking can be effectively joined. A desired approach would be to allow a single framework that can accept a security policy for a particular system and provide the necessary content security services.

Bibliography

[1] Call for proposals for content protection and copy management technologies. http://www.dvb.org/dvb_technology/pdf/cfp_cp_cm.pdf.

[2] Content Protection Technical Working Group (CPTWG). http://www.cptwg.org.

[3] Resource Description Framework (RDF). http://www.w3.org/RDF/.

[4] Video codec for audiovisual services at p*64kb/s. Recommendation H.261, ITU-T, 1993.

[5] Report: Napster users lose that sharing feeling. http://www.cnn.com/2001/TECH/internet/06/28- /napster.usage/, June 2001.

[6] Report: Bertelsmann wants all of Napster. http://www.usatoday.com/life/cyber/invest/2002/04/05/napster.htm, April 2002.

[7] FIPS PUB 180-1. *Secure Hash Standard.* NIST, U.S. Department of Commerce, April 1995.

[8] 4C Entity, LLC. Content protection for pre-recordable media (CPPM). http://www.4centity.com.

[9] 4C Entity, LLC. Content protection for recordable media (CPRM). http://www.4centity.com.

[10] 4C Entity, LLC. Content scrambling system (CSS). http://www.dvdcca.org.

[11] 4C Entity, LLC. Digital Transmission Content Protection(DTCP) white paper. http://www.dtcp.com/data/wp_spec.pdf, July 1998.

[12] D.G. Abraham, G.M. Dolan, G.P. Double, and J.V. Stevens. Transaction security systems. *IBM Systems Journal*, 30:206–229, 1991.

[13] Akamai. http://www.akamai.com.

[14] K. Almeroth and M. H. Ammar. Multicast group behavior in the internet's multicast backbone (MBone). *IEEE Communications Magazine*, 35(6), June 1997.

[15] R. Anderson. The eternity service, 1996.

[16] R. Anderson and M. Kuhn. Tamper resistance - a cautionary note. In *Second Usenix Workshop on Electronic Commerce*, pages 1–11, November 1996.

[17] R.J. Anderson and C. Manifavas. Chameleon – A new kind of stream cipher. In *IWFSE: International Workshop on Fast Software Encryption, LNCS*, 1997.

[18] RTPGW: An application level RTP gateway. http://daedalus.cs.berkeley.edu/software/rtpgw/.

[19] E.M. Bakker and R.B. Tan J van Leeuwen. Prefix routing schemes in dynamic networks. *Computer Networks and ISDN Systems*, 26:403–421, 1993.

[20] D. Balenson, D. McGrew, and A. Sherman. Key management for large dynamic groups: One-way function trees and amortized initialization. Internet Draft, IETF, March 1999. Work in progress.

[21] A. Ballardie. Scalable multicast key distribution. RFC 1949, IETF, 1996.

[22] A. Ballardie and J. Crowcroft. Multicast-specific security threats and countermeasures. In *Proceedings of ISOC Symposium on Network and Distributed System Security*, pages 2–16, San Diego, California, February 1995.

[23] T. Ballardie, P. Francis, and J. Crowcroft. Core based trees (CBT). In Deepinder P. Sidhu, editor, *SIGCOMM*, pages 85–95, San Francisco, California, September 1993. ACM. also in *Computer Communication Review* 23 (4), Oct. 1992.

[24] G. Barish and K. Obraczka. World wide web caching: Trends and techniques. *IEEE Communications Magazine*, May 2000.

[25] D. E. Bell and L. J. LaPadula. Secure computer system: Unified exposition and MULTICS interpretation. Technical Report MTR-2997, MITRE Corp., Bedford, Mass, March 1973.

[26] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *IEEE Symposium on Security and Privacy*, number 96-17, 28, 1996.

[27] J.A. Bloom, I.J. Cox, T. Kalker, J.P.M.G. Linnartz, M.L. Miller, and C.B.S. Traw. Copy protection for DVD video. In *Proceedings of the IEEE*, volume 87, July 1999.

[28] S. Bonisteel. RIAA sues Napster clones: Kazaa, Morpheus and Grokster. http://www.newsbytes.com/news/01/170798.html, October 2001.

[29] S. Brands. Off-line electronic cash based on secret-key certificates. In *Proceedings of the Second International Symposium of Latin American Theoretical Informatics (LATIN '95)*, Valparaiso, Chili, 1995.

[30] J. Brassil, S.H. Low, N. F. Maxemchuk, and L. O'Gorman. Electronic marking and identification techniques to discourage document copying. In *IEEE Infocom*, pages 1278–1287, Toronto, Canada, June 1994.

[31] B. Briscoe. Marks: Zero side-effect multicast key management using arbitrarily revealed key sequences. In *Proc First International Workshop on Networked Group Communication (NGC'99)*, Pisa,Italy, November 1999.

[32] I. Brown, C. Perkins, and J. Crowcroft. Watercasting: distributed watermarking of multicast media. In *Networked Group Communication '99*, pages 286–300, Pisa, Italy, November 1999.

[33] B. Cain, S. Deering, B. Fenner, I. Kouvelas, and A. Thyagarajan. Internet group management protocol, version 3. Internet Draft, IETF, March 2001. Work in progress.

[34] K.L. Calvert, S. Bhattacharjee, E.W. Zegura, and J. Sterbenz. Directions in active networks. *IEEE Communications Magazine*, 36(10):72–78, October 1998.

[35] R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and efficient constructions. In *IEEE Infocom*, New York, March 1999.

[36] C. Castelluccia and G. Montenegro. Securing group management in ipv6 with cryptographically generated addresses. Internet Draft, Internet Engineering Task Force, July 2002. Work in progress.

[37] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Security for structured peer-to-peer overlay networks. In *Symposium on Operating Systems Design and Implementation*, December 2002.

[38] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, and D. Saha. Key management for secure internet multicast using boolean function minimization techniques. In *IEEE Infocom*, New York, March 1999.

[39] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell. A hierarchical internet object cache. In *USENIX 1996 Annual Technical Conference*, San Diego, California, January 1996.

[40] D. Chaum. Blind signatures for untraceable payments. In *Proceedings of CRYPTO '82*, pages 199–203, 1982.

[41] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. *Lecture Notes in Computer Science*, pages 319–327, 1988.

[42] Y. Chawathe. *Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service.* PhD thesis, 2000.

[43] D.R. Cheriton and S.E. Deering. Host groups: A multicast extension for datagram internetworks. In *Data Communications Symposium*, pages 172–179, September 1985.

[44] B. Chor, A. Fiat, and M. Naor. Tracing traitors. *Lecture Notes in Computer Science*, 839:257–270, 1994.

[45] C. M. Christensen. *The Innovator's Dilemma: When New Technologies Cause Great Firms to Fail.* Harvard Business School Press, 1997.

[46] H. Chu, L. Qiao, and K. Nahrstedt. A secure multicast protocol with copyright protection. In *Proceedings of IS&T/SPIE's Symposium on Electronic Imaging: Science and Technology*, January 1999.

[47] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *ACM Sigmetrics*, June 2000.

[48] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling conferencing applications on the internet using an overlay multicast architecture. In *ACM SIGCOMM*, August 2001.

[49] Y.H. Chu, S.G. Rao, and H. Zhang. A case for end system multicast. In *ACM Sigmetrics*, pages 1–12, Santa Clara, California, 2000.

[50] B. Coan, V. Kaul, S. Narain, and W. Stephens. Hasm: Hierarchical application-level secure multicast. Internet Draft, Internet Engineering Task Force, November 2001. Work in progress.

[51] P. Cohen. Napster remains closed following court order. http://maccentral.macworld.com/news/0107- /12.napster.shtml, July 2001.

[52] B. Cox, D. Tygar, and M. Sirbu. Netbill security and transaction protocol. In *First USENIX Workshop of Electronic Commerce Proceedings*, 1995.

[53] W. Dai. Crypto++. http://www.eskimo.com/~weidai/benchmarks.html, June 2000.

[54] Loudeye Music Database. http://www.loudeye.com.

[55] Gracenote CD database (CDDB). http://www.cddb.com.

[56] S. Deering. *Multicast routing in a datagram internetwork.* PhD thesis, Stanford University, Palo Alto, California, December 1991.

[57] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C.G. Liu, and L. Wei. The PIM architecture for wide-area multicast routing. *IEEE/ACM Transactions on Networking*, 4(2):153–162, 1996.

[58] T. Dierks and C. Allen. The TLS protocol version 1.0. RFC 2246, IETF, January 1999.

[59] Digital Content Protection, LLC. High-bandwidth digital content protection (HDCP). http://www.digital-CP.com.

[60] C. Diot, B. N. Levine, B. Lyles, H. Kassan, and D. Balsiefien. Deployment issues for the ip multicast service and architecture. *IEEE Network, special issue on Multicasting*, 2000.

[61] J. Dittmann, M. Stabenau, and R. Steinmetz. Robust MPEG video watermarking technologies. In *Multimedia and Security Workshop at ACM Multimedia*, 1998.

[62] N. Duffield, J. Horowitz, and F. L. Presti. Adaptive multicast topology inference. In *IEEE Infocom*, Anchorage, Alaska, April 2001.

[63] C. Dwork, J. Lotspiech, and M. Naor. Digital signets: self-enforcing protection of digital information (preliminary version). pages 489–498, 1996.

[64] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. Protocol independent multicast-sparse mode (PIM-SM): protocol specification. RFC 2362, IETF, June 1998.

[65] J. Evers. Napster to launch fee-based service. http://www.cnn.com/2001/TECH/computing/01/29/fee.based.napster.idg/, January 2001.

[66] eXtensible Media Commerce Language (XMCL). http://www.xmcl.org.

[67] eXtensible rights Markup Language(XrML). http://www.xrml.org.

[68] J. Fan, P.Q. Judge, and M.H. Ammar. HySOR: group ley management with collusion-scalability tradeoffs using a hybrid structuring of receivers. In *IEEE ICCCN*, 2002.

[69] L. Fan, P. Cao, J. Almeida, and A. Broder. Summary cache: A scalable wide-area web cache sharing protocol. In *ACM SIGCOMM*, volume 28, pages 254–265, September 1998.

[70] J. Feigenbaum, M.J. Freedman, T. Sander, and A. Shostack. Privacy engineering for digital rights management systems. In *Digital Rights Management Workshop*, pages 76–105, 2001.

[71] W. Fenner and S. Casner. A traceroute facility for ip multicast. Internet Draft, Internet Engineering Task Force, July 2000. Work in progress.

[72] P. Francis. Yoid: Your own internet distribution. Unrefereed report, http://www.aciri.org/yoid/, April 2000.

[73] N. Freed and N. Borenstein. Multipurpose internet mail extensions (MIME) part two: Media types. RFC 2046, IETF, November 1996.

[74] Freenet. http://freenet.sourceforge.net.

[75] TRM™: Advanced Audio Fingerprinting from Relatable. http://relatable.com/tech/trm.html.

[76] J. Galvin. Public key distribution with secure DNS. In *Sixth USENIX Security Symposium*, July 1996.

[77] R. Gennaro and P. Rohatgi. How to sign digital streams. *Lecture Notes in Computer Science*, 1294, 1997.

[78] Gnutella. http://gnutella.wego.com.

[79] P. Golle and N. Modadugu. Authenticating streamed data in the presence of random packet loss. In *Network and Distributed System Security Symposium*, 2001.

[80] L. Gong. A secure identity-based capability system. In *IEEE Symposium on Security and Privacy*, pages 56–65, 1989.

[81] L. Gong and N. Shacham. Elements of trusted multicasting. In *Proceedings of the 2nd ACM Conference on Computer and Communications Security*, pages 176–183, Fairfax, Virginia, 1994.

[82] B. Haberman and D. Thaler. Host-based anycast using MLD. Internet Draft, IETF, February 2001. Work in progress.

[83] M. Handley and S. Hanna. Multicast address allocation protocol (AAP). Internet Draft, IETF, June 2000. Work in progress.

[84] M. Handley and V. Jacobson. SDP: session description protocol. RFC 2327, IETF, April 1998.

[85] M. Handley, C. Perkins, and E. Whelan. Session announcement protocol. RFC 2974, IETF, October 2000.

[86] S. Hanna, B. Patel, and M. Shah. Multicast address dynamic client allocation protocol (MADCAP). RFC 2730, IETF, December 1999.

[87] T. Hardjono and B. Cain. Key establishment for IGMP authentication in IP multicast. In *IEEE European Conference on Universal Multiservice Networks(ECUMN)*, CREF, Colmar, France, 2000.

[88] T. Hardjono, R. Canetti, M. Baugher, and P. Dinsmore. Secure IP multicast: Problem areas, framework, and building blocks. Internet Draft, IETF, September 2000. Work in progress.

[89] H. Harney and C. Muckenhirn. Group key management protocol (GKMP) specification. RFC 2093, IETF, July 1997.

[90] F. Hartung and B. Girod. Digital watermarking of raw and compressed video. In *European EOS/SPIE Symposium on Advanced Imaging and Network Technologies*, October 1996.

[91] F. Hartung and B. Girod. Fast public key watermarking of compressed video. In *IEEE International Conference on Image Processing*, Santa Barbara, California, October 1997.

[92] R. J. Hayton, J. M. Bacon, and K. Moody. Access control in an open distributed environment. In *IEEE Symposium on Security and Privacy*, pages 3–14, May 1998.

[93] H. He, T. Hardjono, and B. Cain. Simple multicast receiver access control. Internet Draft, Internet Engineering Task Force, November 2001. Work in progress.

[94] R. Hinden and S. Deering. IP version 6 addressing architecture. RFC 2373, IETF, July 1998.

[95] P. Hoffman. Enhanced security services for S/MIME. RFC 2634, IETF, November 1999.

[96] F. Hohl. A model of attacks of malicious hosts against mobile agents. In *4th Workshop on Mobile Object Systems: Secure Internet Mobile Computations*, 1998.

[97] F. Hohl. Time limited blackbox security: Protecting mobile agents from malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*. Springer-Verlag, 1998.

[98] H. Holbrook and B. Cain. Source-specific multicast for IP. Internet Draft, IETF, March 2000. Work in progress.

[99] H. Holbrook and B. Cain. Source-specific multicast for IP. Internet Draft, IETF, March 2001. Work in progress.

[100] H. Holbrook and D. R. Cheriton. Ip multicast channels: EXPRESS support for large-scale single-source applications. In *SIGCOMM*, Cambridge, Massachusetts, August/September 1999.

[101] M. J. Holliman, N. D. Memon, B.L. Yeo, and M. M. Yeung. Adaptive public watermarking of dct-based compressed image. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 284–295, 1998.

[102] R. Housley, W. Ford, W. Polk, and D. Solo. Internet X.509 public key infrastructure certificate and CRL profile. Request for Comments (Proposed Standard) 2459, Internet Engineering Task Force, January 1999.

[103] Inktomi. http://www.inktomi.com.

[104] Internet 2 Distributed Storage Infrastructure. http://dsi.internet2.edu.

[105] ISO/IEC JTC1/SC29/WG11. Overview of MPEG-7 Standard. *ISO*, March 2001.

[106] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and Jr. J. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Symposium on Operating Systems Design and Implementation*, San Diego, California, 2000.

[107] D. Johnson and S. Deering. Reserved IPv6 subnet anycast addresses. RFC 2526, IETF, March 1999.

[108] P. Q. Judge and M. H. Ammar. CITADEL: A content protection architecture for decentralized peer-to-peer file-sharing systems. Submitted for publication, http://www.cc.gatech.edu/~judge/papers/citadel.ps.

[109] P. Q. Judge and M. H. Ammar. WHIM: watermarking multicast video with a hierarchy of intermediaries. In *Proc. International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, Chapel Hill, North Carolina, June 2000.

[110] P. Q. Judge and M. H. Ammar. Gothic: Group access control architecture for secure multicast and anycast. In *IEEE Infocom*, July 2002.

[111] P. Q. Judge and M. H. Ammar. The role of watermarking in securing peer-to-peer systems. In *Workshop on Multimedia and Security at ACM Multimedia*, December 2002.

[112] P. Q. Judge and M. H. Ammar. Security issues and solutions in multicast content distribution: A survey. *IEEE Network, special issue on Multicasting: An Enabling Technology*, 2003.

[113] M. Kaplan. IBM Cryptolopes^{TM}, superdistribution and digital rights management. http://www.research.ibm.com/people/k/kaplan, 1996.

[114] D. Karger, E. Lehman, T. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing*, pages 654–663, El Paso, Texas, 4–6 May 1997.

[115] P. Karger and A. J. Herbert. An augmented capability architecture to support lattice security and traceability of access. In *IEEE Symposium on Security and Privacy*, pages 2–12, 1984.

[116] D. Katabi and J. Wroclawski. A Framework for Global IP-Anycast (GIA). Internet Draft, IETF, June 1999. Work in progress.

[117] C. Kaufman, R.Perlman, and M. Speciner. *Network Security: Private Communication in A Public World.* Prentice-Hall, 1995.

[118] KaZaA. http://www.kazaa.com.

[119] S. Kent. Encryption-based protection for interactive user/computer communication. In *Fifth Data Communications Symposium*, September 1977.

[120] S. Kent and R. Atkinson. Security architecture for the internet protocol. RFC 2401, IETF, November 1998.

[121] A. Keromytis, V. Misra, and D. Rubenstein. Sos: Secure overlay services. In *ACM SIGCOMM*, August 2002.

[122] B. King. Napster to start filtering songs. http://www.wired.com/news/politics/0,1283,42140,00.html, March 2001.

[123] E. Koch and J. Zhao. Towards robust and hidden image copyright labeling. In *IEEE Workshop on Nonlinear Signal and Image Processing*, 1995.

[124] L. Kohnfelder. Towards a practical public-key cryptosystem. Master's thesis, M.I.T., May 1978.

[125] B. Lampson. Protection. In *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, pages 437–443, Princeton University, 1971.

[126] C. E. Landwehr. Formal models for computer security. *ACM Computing Surveys*, 13(3):247–278, 1981.

[127] B. N. Levine, S. Paul, and J. J. Garcia-Luna-Aceves. Organizing multicast receivers deterministically by packet-loss correlation. In *ACM Multimedia*, pages 201–210, 1998.

[128] X. Li, S. Paul, and M. Ammar. Layered video multicast with retransmissions (LVMR): evaluation of hierarchical rate control. In *IEEE Infocom*, page 1062, San Francisco, California, March/April 1998.

[129] LimeWire. http://www.limewire.org.

[130] J. C. Lin and S. Paul. RMTP: a reliable multicast transport protocol. In *IEEE Infocom*, San Fransisco, California, March 1996.

[131] A. D. Rubin M. Waldman and L. Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[132] Macrovision Corp. Macrovision copy protection. http://www.macrovision.com/solutions/video/copyprotect/.

[133] Open Mash. http://www.openmash.org.

[134] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *ACM Multimedia*, 1995.

[135] C. McCollum, J. Messing, and L. Notargiacomo. Beyond the pale of MAC and DAC - defining new forms of access control. In *IEEE Computer Society Symposium on Security and Privacy*, pages 190–200, May 1990.

[136] P. McDaniel, H. Harney, P. Dinsmore, and A. Prakash. Multicast security policy. Internet Draft, IETF, November 2000. Work in progress.

[137] G. Medvinsky and B. C. Neuman. Netcash: A design for practical electronic currency on the internet. In *Proceedings of the First ACM Conference on Computer and Communications Security*, volume 1993, pages 102–106, 1993.

[138] D. Meyer. Extended allocations in 233/8. Internet Draft, IETF, April 2001. Work in progress.

[139] D. Meyer and P. Lothberg. GLOP addressing in 233/8. RFC 2770, IETF, February 2000.

[140] Microsoft Corp. Understanding secure audio path. http://www.microsoft.com/windows/windowsmedia/wm7/WMRMsap_bro.pdf.

[141] Midbar Tech. Cactus data shield. http://www.midbartech.com.

[142] S. Mittra. Iolus: A framework for scalable secure multicasting. *ACM Computer Communication Review*, 27(4):277–288, October 1997. ACM SIGCOMM'97, Sept. 1997.

[143] J. Moy. Multicast extensions to OSPF. RFC 1075, Internet Engineering Task Force, 1991.

[144] S. Murphy, M. Badger, and B. Wellington. OSPF with digital signatures. RFC 2154, IETF, June 1997.

[145] MusicBrainz. http://musicbrainz.org.

[146] MusicCity. http://www.musiccity.com.

[147] MusicNet. http://www.musicnet.com.

[148] Napster. http://www.napster.com.

[149] NDS Limited. NDS VideoGuard. http://www.nds.com/solutions/videoguard.html.

[150] B. C. Neumann. Proxy-based authorisation and accounting for distributed systems. In *13th International Conference on Distributed Computing Systems*, pages 283–291, Pittsburgh, Penn, may 1993.

[151] M. Nilsson. ID3 tag version 2.3.0. informal standard, February 1999.

[152] OpenSSL. http://www.openssl.org.

[153] J.S. Park and R. Sandhu. Smart certificates: Extending x.509 for secure attribute service on the web. *National Information Systems Security Conference*, 1999.

[154] J.S. Park, R. Sandhu, and J. Schifalacqua. Security architecture for controlled digital information dissemination. In *Annual Computer Security Applications Conference (ACSAC)*, December 2000.

[155] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. RFC 1546, IETF, November 1993.

[156] S. Paul, K. K. Sabnani, J.C.H. Lin, and S. Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, April 1997.

[157] B. Pearson. Digital transmission content protection. http://www.dtcp.com/data/dtcp_tut.pdf, June 1999.

[158] A. Perrig, R. Canetti, D. Song, and D. Tygar. Efficient and secure source authentication for multicast. In *Network and Distributed System Security Symposium*, February 2001.

[159] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Attacks on copyright marking systems. In D. Aucsmith, editor, *Information Hiding, Second International Workshop*, pages 219–239, Portland, Oregon, U.S.A.,, April 15-17 1998.

[160] F. A. P. Petitcolas and R. J.Anderson. Evaluation of copyright marking systems. In D. Aucsmith, editor, *IEEE Multimedia Systems (ICMCS'99)*, volume 1, pages 574–579, Florence, Italy, June 7-11 1999.

[161] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In *ACM Conference on Computer and Communications Security*, pages 151–160, 1997.

[162] C. Pfleeger. *Security in Computing.* Prentice-Hall International, Inc., Englewood Cliffs, NJ, 1997.

[163] PressPlay. http://www.pressplay.com.

[164] T. Pusateri. Distance vector multicast routing protocol. Internet Draft, Internet Engineering Task Force, March 1999. Work in progress.

[165] L. Qiao and K. Nahrstedt. Watermarking method for mpeg encoded video: Towards resolving rightful ownership. In *IEEE Multimedia Computing and Systems*, June 1998.

[166] C. Metz R. Boivie, N. Feldman. Small group multicast: A new solution for multicasting on the internet. *Internet Computing*, 4(3):75–79, May/June 2000.

[167] P. Radoslavov, D. Estrin, R. Govindan, M. Handley, S. Kumar, and D. Thaler. The multicast address-set claim (MASC) protocol. RFC 2909, IETF, September 2000.

[168] B. Ramsdell. S/MIME version 3 certificate handling. RFC 2632, IETF, June 1999.

[169] B. Ramsdell. S/MIME version 3 message specification. RFC 2633, IETF, June 1999.

[170] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In *SIGCOMM*, 2001.

[171] S. Ratnasamy and S. McCanne. Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements. In *IEEE Infocom*, New York, March 1999.

[172] M.G. Reed, P.F. Syverson, and D.M. Goldschlag. Proxies for anonymouxs routing. In *Proceedings of the 12th Annual Computer Security Applications Conference*, pages 95–104, San Diego, CA, December 1996. IEEE CS Press.

[173] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[174] E. Rescorla. HTTP over TLS. RFC 2818, IETF, May 2000.

[175] R. Rivest. The MD5 message-digest algorithm. RFC 1321, IETF, April 1991.

[176] R. Rivest and B. Lampson. SDSI — a simple distributed security infrastucture. Technical report, M.I.T., April 1996.

[177] R. Rivest and A. Shamir. Payword and micromint: Two simple micropayment schemes. In *Security Protocols Workshop*, pages 69–87, 1996.

[178] P. Rohatgi. A compact and fast hybrid signature scheme for multicast packet authentication. In *ACM Conference on Computer and Communications Security*, November 1999.

[179] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Middleware*, 2001.

[180] T. Sander and C. F. Tschudin. Protecting mobile agents against malicious hosts. In Giovanni Vigna, editor, *Mobile Agents and Security*. Springer-Verlag, 1998.

[181] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *IEEE Computer*, 20(2):38–47, 1996.

[182] R. Sandhu, D. Ferraiolo, and R. Kuhn. The NIST model for role-based access control: Towards a unified standard. In *ACM Workshop on Role-Based Access Control*, Berlin, Germany, July 2000.

[183] K. Sarac and K. Almeroth. Scalable techniques for discovering multicast tree topology. In *NOSSDAV*, Port Jefferson, New York, June 2001.

[184] S. Saroiu, P. Krishna Gummadi, and S.D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Multimedia Computing and Networking(MMCN)*, 2002.

[185] V. Scarlata, B.N. Levine, and C. Shields. Responder anonymity and anonymous peer-to-peer file sharing. In *ICNP*, November 2001.

[186] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: a transport protocol for real-time applications. Request for Comments (Proposed Standard) 1889, Internet Engineering Task Force, January 1996.

[187] S. Sen, J. Rexford, and D. Towsley. Proxy prefix caching for multimedia streams. In *IEEE Infocom*, New York, March 1999.

[188] C. Shields and J. J. Garcia-Luna-Aceves. KHIP - a scalable protocol for secure multicast routing. In *SIGCOMM*, pages 53–64, 1999.

[189] C. Shields and B.N. Levine. A protocol for anonymous communication over the internet. In *ACM Conference on Computer and Communication Security*, November 2000.

[190] O. Sibert, D. Bernstein, and D. Van Wie. The digibox: a self-protecting container for electronic commerce. In *USENIX Electronic Commerce*, 1995.

[191] T. Sikora. Mpeg digital video–coding standards. In *IEEE Signal Processing Magazine*, volume 14, pages 82–100, September 1997.

[192] E. Sit and R. Morris. Security considerations for peer-to-peer distributed hash tables. In *International Workshop on Peer-to-Peer Systems*, March 2002.

[193] J. Snoeyink, S. Suri, and G. Varghese. A lower bound for multicast key distribution. In *IEEE Infocom*, Anchorage, April 2001.

[194] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *SIGCOMM*, 2001.

[195] D. L. Tennenhouse, J. M. W. Smith, D. Sincoskie, D. J. Wetherall, and J. G. M. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):–, January 1997.

[196] D. Thaler, M. Handley, and D. Estrin. The Internet Multicast Address Allocation Architecture. RFC 2908, IETF, September 2000.

[197] J. Touch and S. Hotz. The X-bone. In *Third Global Internet Mini-Conference in conjunction with Globecom*, Sydney, Australia, Nov 1998.

[198] T.Wu and S. Wu. Selective encryption and watermarking of mpeg video. Technical report, North Carolina State University.

[199] A. Van Moffaert and O. Paridaens. Security issues in protocol independent multicast - sparse mode (pim-sm). Internet Draft, IETF, February 2002. Work in progress.

[200] D. Verma. *Content Distribution Networks: An Engineering Approach.* Wiley, January 2002.

[201] R. Vida et al. Multicast listener discovery version 2 (MLDv2) for IPv6. Internet Draft, IETF, February 2001. Work in progress.

[202] Ogg Vorbis. http://www.vorbis.com.

[203] N. R. Wagner. Fingerprinting. In *Symposium on Security and Privacy*, pages 18–22, April 1983.

[204] D. Wallner, E. Harder, and R. Agee. Key management for multicast: Issues and architectures. RFC 2627, IETF, June 1999.

[205] L. Wei. Authenticating PIM version 2 messages. Internet Draft, IETF, July 2000. Work in progress.

[206] S. Weibel, J. Kunze, C. Lagoze, and M. Wolf. Dublin core metadata for resource discovery. RFC 2413, IETF, September 1998.

[207] D. Wessels and K. Claffy. Internet cache protocol (ICP), version 2. Request for Comments (Informational) 2186, Internet Engineering Task Force, September 1997.

[208] E. Wold, T. Blum, D. Keislar, and J. Wheaton. Content-based classification, search, and retrieval of audio. In *IEEE Multimedia*, 1996.

[209] C.K. Wong, M. Gouda, and S.S. Lam. Secure group communications using key graphs. *ACM Computer Communication Review*, 28(4):68–79, September 1998.

[210] C.K. Wong and S.S. Lam. Digital signatures for flows and multicasts. *IEEE/ACM Transactions on Networking*, 7, 1999.

[211] T.Y.C. Woo and S. Lam. Designing a distributed authorization service. In *IEEE Infocom*, San Francisco, CA, March 1998.

[212] E.W. Zegura, M.H. Ammar, Z. Fei, and S. Bhattacharjee. Application-layer anycasting: a server selection architecture and use in a replicated web service. *IEEE/ACM Transactions on Networking*, 8(4):455–466, August 2000.

[213] E.W. Zegura, Kenneth L. Calvert, and Samrat Bhattacharjee. How to model an internetwork. In *IEEE Infocom*, San Fransisco, California, March 1996.

[214] B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, April 2001.

Vita

Paul Judge was born in Baton Rouge, Louisiana on March 5, 1977. He began matriculation at Morehouse College in 1995 and received the B.S. in Computer Science three years later in May 1998. Along the way, he held interns at NASA's Stennis Space Center and was also a software developer at IBM from 1997 to 1998.

He continued in pursuit of a longtime goal of an advanced degree and performing research in computer security. He began at the College of Computing in August 1998 and will receive his Ph.D. in December 2002. Along the way, he has also done work with the research and development group at CipherTrust, a company that creates Internet messaging security products.