Flexible Owner Retained Access Control for Document Management Systems

by

Alexander Michael Hoole
B.Sc., University of Victoria, 2003

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of

MASTER OF APPLIED SCIENCE

in the Department of Electrical and Computer Engineering

Flexible Owner Retained Access Control for Document Management Systems

by

Alexander Michael Hoole
B.Sc., University of Victoria, 2003

Supervisory Committee

Dr. Issa Traore, (Department of Electrical and Computer Engineering)

Supervisor

Dr. Aaron Gulliver, (Department of Electrical and Computer Engineering)

Departmental Member

Dr. Martial Agueh, (Department of Mathematics and Statistics)

Outside Member

Supervisory Committee

Dr. Issa Traore, (Department of Electrical and Computer Engineering)

Supervisor

Dr. Aaron Gulliver, (Department of Electrical and Computer Engineering)

Departmental Member

Dr. Martial Agueh, (Department of Mathematics and Statistics)

Outside Member

Abstract

The majority of the security policy and enforcement frameworks deployed today require a centralized security model. These models are often tied to a central authentication service or operating system (OS) service. In collaboration environments, such as the Internet, there is no guarantee that users will be using the same OSs, authentication services, or access control policies. In such context, the risk of data interception or information leakage is extremely high during collaborations. Therefore, there is a need to control access to information that remains independent of these, and other, platform specific security features. Owner-retained access control, derived from labeling practices that historically have been used in paper based access control schemes, such as the military use of ORCON label, can provide such a feature. The owner-retained access control model allows for the owner of a document, not necessarily the creator of the document, to specify and maintain the access control restrictions for their data, even after disseminating such data. The access control policy itself is not sufficient to guarantee security; an enforcement framework is also required to ensure that the rules specified in the policy are enforced. The framework will allow us to overcome some of the limitations found in other access control policies. Discretionary access control, as an example, allows for authorized users to copy and distribute data once it has been accessed; thus breaking the principle of attenuation of privilege.

In an attempt to satisfy this objective, we propose in this thesis, a formal security model and flexible policy specification and enforcement framework that allows for owner-retained control for document distribution in scalable collaborative environments. The body of the thesis also includes a description and validation of the security protocols that were developed to provide a framework for enforcing the security policy, the architecture and implementation of the prototype application (ORCS), case studies, and a performance evaluation of the most costly operations.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

ACL	access control list
ACM	access control matrix
AD	active directory
AES	advanced encryption standard
AIS	automated information system
API	application programming interface
BLP	Bell, and LaPadula
BMA	British medical association
CA	certificate authority
CISD	convergent information systems division
CPU	central processing unit
DAC	discretionary access control
DoD	department of defense
DOM	document object model
DRM	digital rights management
FLASK	flux advanced security kernel
FTP	file transfer protocol
GFAC	generalized framework for access control
HRU	Harrison, Ruzzo, and Ullman
HTTP	hypertext transfer protocol
IBAC	identity-based access control
JVM	Java virtual machine
MAC	mandatory access control
NIST	national institute of standards and technology
NOFORN	no foreign nationals

NSA	national security agency
NTFS	new technology filesystem
OASIS	organization for the advancement of structured information standards
ORAC	owner-retained access control
ORC	owner-retained control
ORCON	originator control
ORCS	owner-retained control system
ORGCON	organization controlled
ORGREP	organization control representative
PAC	propagated access control
PDF	portable document format
PMA	protected memory area
RBAC	role based access control
RECREP	recipient representative
RMS	rights management services
RSA	Rivest, Shamir, and Adleman
RSBAC	rule set based access control
SAML	security assertion markup language
SAX	simple API for XML
SCP,	secure copy
SELinux	security enhanced Linux
SSL	secure sockets layer
TAM	typed access matrix
TCB	trusted computing base
TCSEC	trusted computer system evaluation criteria
UCON	usage control
XSD	XML schema definition
XML	extensible markup language

Acknowledgments

During my time as a graduate student, numerous individuals stand out who have offered their generous support and encouragement. I would like to take this opportunity to sincerely thank all of these individuals.

First of all, I would like to thank my supervisor, Dr. Issa Traore, for his guidance, support and encouragement throughout my program and for suggesting the topic of this thesis. By providing direction and recommendations he has been instrumental in helping me complete this work.

Next, my gratitude goes to Richard Grohovac, my employer and friend for the past ten years, who has been extremely understanding and supportive, as I have continued on into advanced studies in academia.

I am also very grateful to the Department of Electrical and Computer Engineering along with the staff and faculty of the University of Victoria who have provided guidance and assistance countless times over the years.

My deepest gratitude goes to those people who are most important to me in my life, my family and friends. The unconditional love and support of my wife Lindsey, parents, siblings, close friends and extended family, have always shown to me, has given me a reason to strive toward the goals I set in life.

Dedication

To my wife Lindsey and our newborn son Liam.
You are my reason for being!

Chapter 1 Introduction

Recently, many new security applications and environments have been created to assist in securing information. Unfortunately, even today, as has been the case for more than ten years, many systems cannot meet the complete requirements for data security of the customer [33, 25]. Scalable collaborative environments, personal handheld devices, and other networked environments demonstrate areas where the security demands of the user may not be sufficiently handled by the application base. When collaboration must occur over mediums involving unauthorized users, such as the Internet, the risk of data interception or information leakage is high. In order for the users of such systems to feel confident when using the features of the system while transferring secure information, a flexible security policy and enforcement framework is required.

Most of the security policy and enforcement frameworks deployed today require a centralized security model. These models are often tied to a central authentication service or operating system service, deploying policies based on mandatory, discretionary, or role based access control. In collaboration environments, such as the Internet, there are no guarantees that users will be running the same operating system, authentication services, or access control policies. A security model, framework, and mechanisms to allow the owner of information to retain control over the information when it departs their environment, is therefore warranted. A means to control access to information that remains independent of operating system security mechanisms and other platform specific security features is therefore needed to show that owner-retained control is a realistic possibility in the digital environment.

Access control is focused on one central concept: the protection of data. Owner-retained access control systems, derived from labeling practices that historically have

been used in paper-based access control schemes (such as the ORCON label used in military documents that is specified in the TCSEC that we explain later [21]), can provide such a feature. Owner-retained access control allows for the owner of a document, not necessarily the creator of the document, to specify and maintain the access control restrictions for their data. The access control policy itself is not sufficient to guarantee security; an enforcement framework is also required to ensure that the rules specified in the policy are enforced.

The framework can overcome some of the limitations found in other access control policies. Discretionary access control (DAC), as an example, allows for authorized users to copy and distribute data once it has been accessed; thus breaking the principle of attenuation of privilege [13, 20]. If DAC were used for online sale of digital media, there would be nothing to prevent a user from sharing their copy with others. Mandatory access control (MAC), as another example, is too rigid and inflexible to allow for the implementation of certain features such as originator control (ORCON) and no foreign nationals (NOFORN) [21, 25].

The objective of this area of research is to move towards the creation of a model, flexible policy specification, and enforcement framework that would allow for owner-retained control for document distribution in many different environments. These environments could include the following: version control systems, operating systems, distributed and web based environments, scalable collaborative environments, and portable handheld devices. The controls could be used to protect virtually any type of digital content, such as full documents, music, media, images, and segments of raw data.

In an attempt to satisfy this objective, this thesis documents and discusses the problems associated with owner-retained control policy and the related problem of shared ownership, the investigations employed during research and the results obtained, and final conclusions.

1.1 Motivation

Access control is the enforcement of rules, specifying what access should be granted to authenticated subjects for requested objects. Subjects are authenticated users and

processes acting on the users' behalf, whereas objects are the electronic resources being protected.

Since the inception of computers, we have continually tried to address the issues surrounding the protection of digital content. Historically, the research was centered on mandatory and discretionary access control and authentication mechanisms. While mandatory and discretionary access control mechanisms do an excellent job of protecting information in certain environments, there still remains a void to be filled in the modern marketplace. In more recent years, additional attempts to secure information and its access have surfaced in the research community. Research areas have included digital rights management, security policies, and the design and validation of security protocols. The majority of early research in the area was primarily driven by the requirements of the military and government; more recent approaches, however, have stemmed from the needs of the business marketplace.

A major problem still existing in data security is the security of information in large distributed heterogeneous environments, rather than just on a single system. When data is sent from a secure machine to a remote machine, via email, file transfer, physical transfer (disk, memory stick, etc.), or by other means, to the target machine, the environment is often not the same and cannot guarantee the level of security that the origin system maintained. This data distribution security problem is most prevalent if the security mechanisms employed on the data are specific to the origin machine and the remote machine does not support those same mechanisms. One must look no further than the access control labels and mechanisms associated with files in most modern operating systems. If the file is emailed to another user, that user takes ownership of the file (rather than just possession, as defined later in the model), and can do whatever he wants with the file. In existing distributed environments, a subject may place a set of permissions on an object within his local environment. So long as the object remains in the environment, the permissions are enforced. It is the case, however, that when the object leaves the environment the access control labels (access attributes, etc.) that were placed on the object are no longer enforced. Thus, there is a need to have a mechanism that will allow

for the owner-retained-controlled access control labels to be enforced, regardless of the environment in which the data resides. This mechanism should provide the following:

- Access must be either granted or refused upon request.
- The environment must be a trusted, secure computing base.
- There should be an allowable difference between the environment policy and the object policy.
- Support for digital signature/certification to provide ownership/control vs. possession/control
- Maintain a reasonable level of performance.

Ideally, the heterogeneous environments would all support the labels of a given object. If that were to become a reality, however, there is still a potential transition period where the environments would not all be supported.

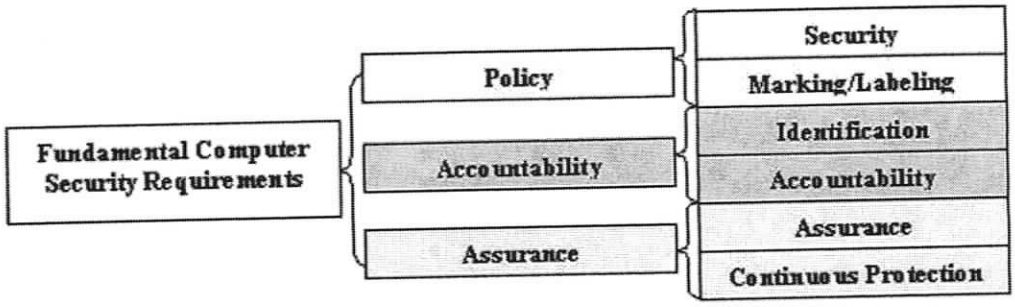
Thus, there still remains a strong need for access control mechanisms that provide mandatory enforcement while preventing data to flow out of the owner's control. Unsatisfied needs still include mandatory enforcement, based on user attributes other than just clearance level, within an automated system of special markings including dissemination controls, release markings, handling caveats, and warnings [33]. This demand falls outside of the capabilities of mandatory and discretionary controls and has been identified by multiple studies, as mentioned by McCollum et al. [33].

1.2 History

Access is a specific type of interaction between an object and a subject that may result in the flow of information between the two. Access controls place constraints on this flow of information that can be verified by evidence, typically stored in an authorization database.

Access control relies on some historical underpinnings that have evolved to what it is today through efforts initiated by military and national security arenas, as well as the academic and commercial research laboratories. It has been an active area of research since the 1960's and 1970's, beginning with mandatory and discretionary access control models [46]. The purpose of access control is to fulfill part of the security policy of a given entity. The trusted computer system evaluation criteria (TCSEC) [21], a standard

produced by the US Department of Defense (DoD) for the evaluation of trusted computer systems, mandates that a secure system be composed of six required parts (as seen in Figure 1).



(1) Security Policy: There must be an explicit and well-defined security policy enforced by the system.

(2) Marking/Labeling: Access control marking/labeling must be associated with objects.

(3) Identification: Individual subjects must be identified.

(4) Accountability: Audit information must be selectively kept and protected so that actions affecting security can be traced to the party responsible for a particular event.

(5) Assurance: The computer system must contain hardware/ software mechanisms that can be independently evaluated to provide sufficient assurance that the system enforces requirements 1->4.

(6) Continuous Protection: The trusted mechanisms that enforce these basic requirements must be continuously protected against tampering and/or unauthorized modifications.

Figure 1: Fundamental Computer Security Requirements

As stated in [36], access control is part of the control objective that refers to a statement of intent with respect to control over some aspect of an organization's resources and/or processes. This statement usually comes in the form of a security policy. Security policies are a statement of intent that defines control over, access to, dissemination of, and modification of information. As defined in TCSEC, a security policy is as follows:

> "The set of laws, rules, and practices that regulate how an organization manages, protects, and distributes sensitive information" [21].

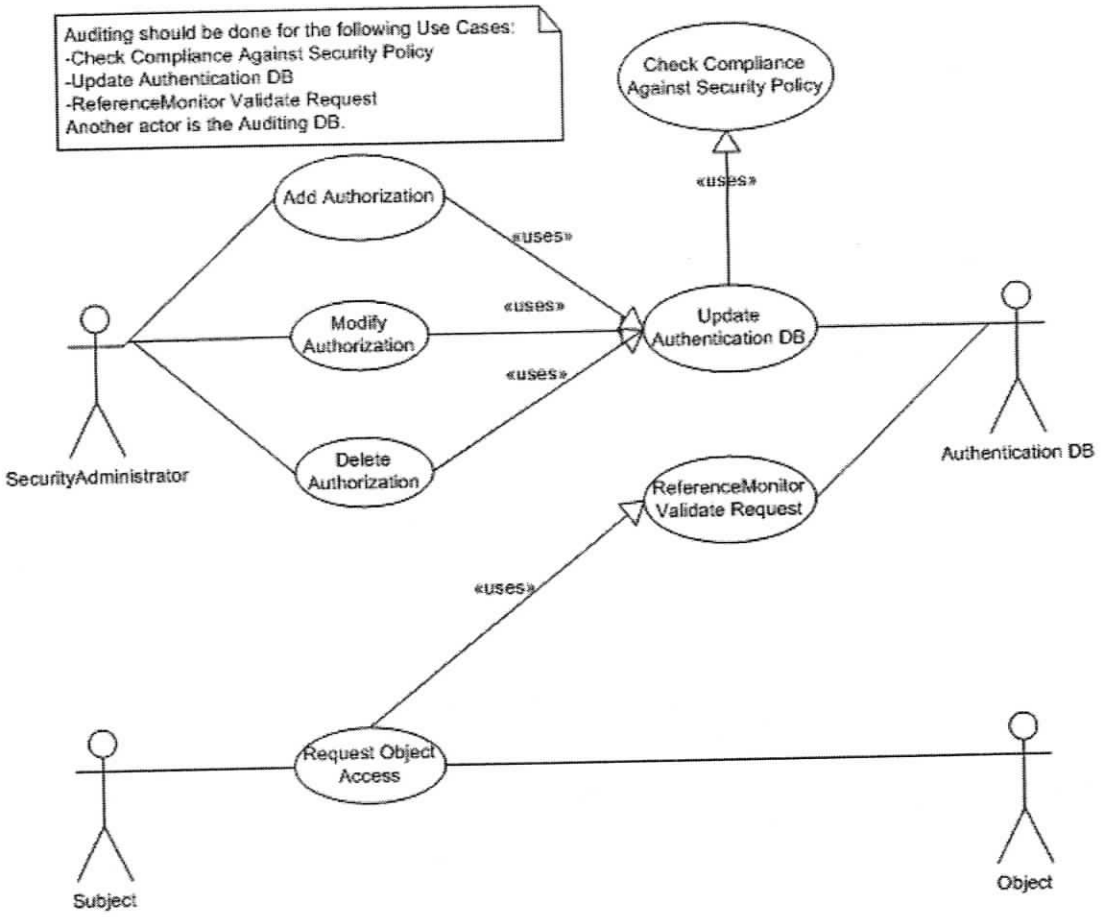


Figure 2: Secure System Depiction

Using access control, we can restrict authenticated user access to resources. Access control, however, cannot secure a system alone; it is the other required components of a secure system working in conjunction with access control that helps create a secure system. The other primary components can be visualized in a high-level diagram (Figure 2) and are further explored in [21, 13, 43].

1.3 The Owner-Retained Access Control Problem

On many popular systems today, a great deal of focus for access control is directed toward the own and copy rights. These two rights are related to the principle of attenuation of privilege [13, 20]. The principle of attenuation of privilege implies that it must be impossible for a subject to give away rights to an object that it itself does not possess. In a heterogeneous distributed environment, the rights that are applied in one

environment often are not enforceable in a second environment. This is referred to as the owner-retained control problem.

Attempts have been made to address the issue of owner-retained control by several researchers in the past. Specifically, the *Propagated Access Control* (PAC) as introduced by Graubart in 1989 [25], *Owner-Retained Access Control* (ORAC) as introduced by McCollum et al. in 1990 [33], the work of Abrams et al. with the *Organization Controlled* (ORGCON) model in their *Generalized Framework for Access Control* (GFAC) in 1991 [1], and Sandhu et al. approached ORCON in 1992 with their work using the *Typed Access Matrix* (TAM) model [41, 42]. Hauser also did related work in 1993 when he proposed integrating an extension to access control to include licensing called *Stateful Access Control* [27]. Recently in 2002, Park and Sandhu have approached the ORCON access control marking again in their work regarding *Usage Control* (UCON) [39].

McCollum et al. raised the concept of owner-retained access control as being needed to fulfill the void left by mandatory and discretionary access control in certain environments [33]. Existing models do not allow for the more complex requirements of certain environments relating to the controlled dissemination of data. There is need for a model that allows for the mandatory enforcement of the access control rules, and a guarantee that data cannot flow to subjects that have not explicitly been identified as being permitted to receive the data. Mandatory access controls are too inflexible to realistically implement owner-retained control in these environments and discretionary controls are generally considered not secure enough. An owner-retained access control model would ideally provide the ease-of-use seen in discretionary models, combined with the security that is realized in mandatory models.

While researchers continue to design new models for addressing owner-retained control issues, several implementation concerns also must be tested for performance implications. Other questions, related to where the technologies are implemented that need to be taken into account:

- At which level of the system architecture should a particular mechanism be implemented?

- Should the technology be portable to all architectures?
- How fast can the enforcement mechanisms perform their task?
- How could the mechanisms be attacked?
- Can the model be easily adapted to future changes?

The problems that this thesis addresses are threefold: (1) definition of theoretical policy and enforcement models to address the issue of owner-retained access control, (2) implementation of a platform independent application prototype to enforce portions of the models, and (3) evaluation of the prototype to determine if such an approach to owner-retained access control is feasible in a production environment.

1.4 Approaches

Traditionally, access control in digital systems is implemented at the operating system level (or kernel level) of an operating system; this approach helps ensure that the mechanisms are more protected from malicious attacks since the code executing at this level are protected by hardware mechanisms. Only the code executing at the kernel layer can typically interface and effect the operation of other code at the kernel level.

Problems arise from this approach since access control in digital systems has typically been approached with narrow focus. Most systems were built to comply with military, intelligence and government standards and specifications for access control. They were not developed with interoperability in mind.

Owner-retained access control is primarily concerned with the ability of the data owner to have his data protection policy enforced independently of where the data travels. Thus, we approach the problem by first identifying and defining the primary elements used in owner-retained access control. These definitions are then used to build a model for a security policy for owner-retained access control. Then we expand this model with an enforcement model that may allow for a portable implementation.

We approach the problem of owner-retained access control from a platform-independent perspective for, among other reasons, the purpose of showing that it is possible to have a security policy attached to the data it is meant to protect rather than implemented in the system which protects it.

1.5 Contributions

This thesis proposes both a theoretical policy model and enforcement model. The models work towards addressing the issue of owner-retained access control. We also provide a prototype application to test the model, and conduct a performance and security analysis of the framework. The contributions of the thesis include the following:

1) Definition of a formal security model for the concept of owner-retained access control in collaborative environments.
2) The implementation of an application named ORCS for the enforcement of owner-retained access control.
3) Verify that owner-retained control can realistically be applied to data using existing technologies in such a manner that CPU performance is not a major issue.

1.6 Outline

The remainder of this thesis is divided into chapters concerned with addressing some of the problems of owner-retained access control and implementing these portions portably and efficiently. The following chapters include a summary of related work (Chapter 2), a detailed description of our formal security policy and enforcement models (Chapter 3), the architecture of the prototype application called ORCS (Chapter 4), a description and validation of the application performance and security protocols used by the framework (Chapter 5), two related case studies (Chapter 6), and our final conclusions (Chapter 7).

Chapter 2 - Related Work

Chapter 2 introduces background material related to the concept of owner-retained access control, access control in general, digital rights management and several mainstream implementations of these concepts.

Chapter 3 - Proposed Solution to the Owner Retained Control Problem

Chapter 3 discusses and defines the theoretical policy and enforcement models for owner-retained control.

Chapter 4 - Implementation

Chapter 4 presents the ORCS prototype as an implementation of the key features specified in the model defined in Chapter 3. The chapter discusses possible applications and architectural approaches, the proposed solution, a discussion of security protocols, the document format used to represent the model rules and definitions, the technologies consumed, and a description of the application.

Chapter 5 - Evaluation Results

Chapter 5 presents a performance evaluation of the most costly operations of the ORCS framework, followed by a discussion and analysis of the security protocols used.

Chapter 6 - Case Studies

Chapter 6 discusses how the ORCS framework could be used theoretically to implement several well-known security policies. The British medical association (BMA) patient protection policy is introduced as a case study, followed by the Chinese Wall Policy used by the financial sector.

Chapter 7 - Concluding Remarks

Chapter 7 provides some concluding remarks and indicates directions to take regarding future work in the area of owner-retained access control models, their enforcement, and related performance.

Chapter 2 Related Work

Access control has had more than 30 years to mature in the digital world. Beginning in the 1970's with the Matrix Model [29], BLP Model [11] and HRU Model [26], through to the recent models of today, researchers have continued to strive to find a way to address issues such as ORCON in access control policies. A summary of these attempts to satisfy the ORCON labeling, and other significant policies is depicted in Figure 3. In this section, we will explore the related work that led to the creation of owner-retained access control models and the needs that still remain.

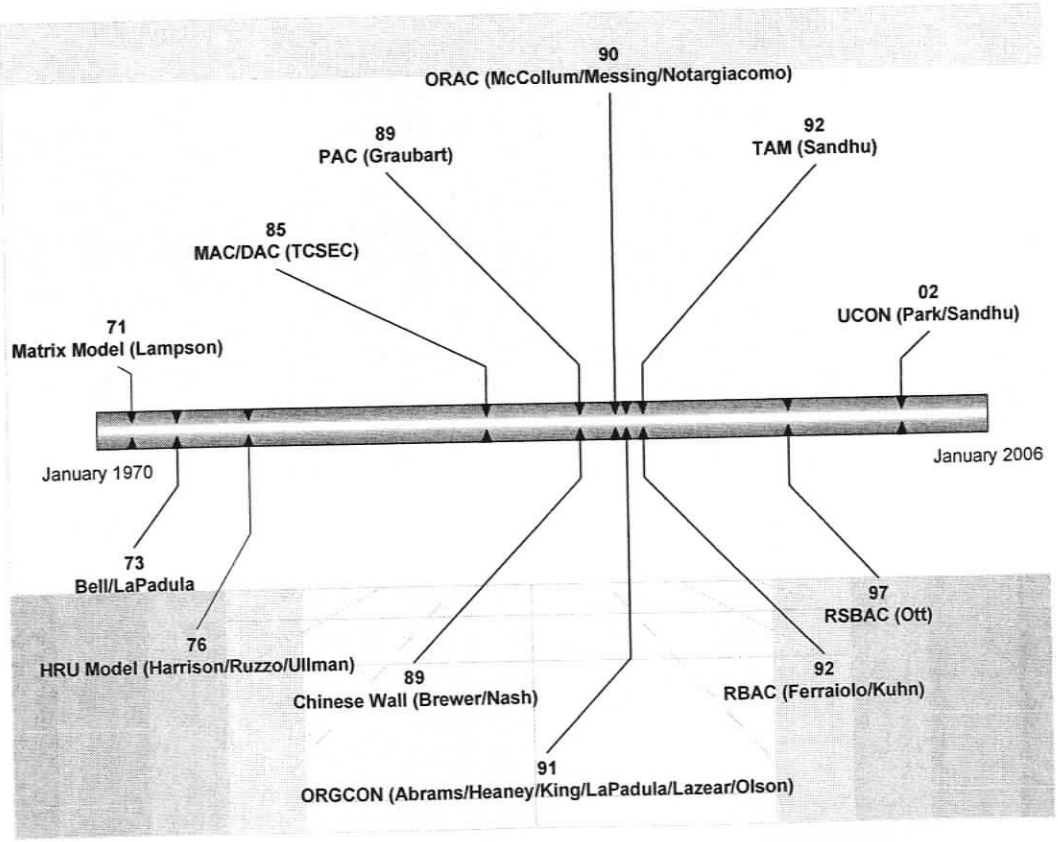


Figure 3: Timeline of Owner-Retained Related Access Control Policies

2.1 Owner-retained Access Control Issues

Owner-retained access control originally stems from document labeling practices in the military [21]. In particular, the marking of ORCON was used to specify that distribution of information, bearing its label, is not to be redistributed to any entity, outside of those specified to have access, without the authorization of the originator of the information. Owner-retained access control should also be viewed in a larger scope; an owner should be able to retain control over their objects regardless of where the object is located. Furthermore, owner-retained control should be considered to be more than just the right to read data, it should specify a set of rules that can be allowed or disallowed on a per subject basis.

As time has progressed, ORCON remains one of the more difficult access control policies to implement. Work has been done related to ORCON by researchers over the past 30 years [25, 33, 1, 41, 39]. More recently, issues relating to owner-retained access control have been approached by digital rights management (DRM) research. In fact, the very recent work of Park and Sandhu has attempted to encapsulate both DRM and access control into a unified framework called usage control (UCON).

The need still remains, however, for a framework and implementation that will allow users to collaborate on information over distributed heterogeneous environments that provides and enforces owner-retained access control on objects for the subjects.

2.2 Dissemination Control

Originally, electronic access controls arose out of a need to have an equivalent means of securing electronic data to that which was already used in the dissemination of printed data. Military, intelligence and government systems have many such people-paper systems in use that typically involve an authentication scheme combined with a mandatory access control mechanism based on labeling [25, 33, 1]. Dissemination control is related to the control over the distribution of information. Dissemination control has arisen again in the research literature as it relates to access control and digital rights management (DRM) [49].

There are many dissemination and handling restrictions and markings used within the DOD and intelligence community for use in the manual handling of classified documents. We will briefly outline several of these markings as they form part of the foundation of our work.

McCollum et al. discusses several markings that are of interest, as they do not depend solely on the subject's clearance. These are ORCON, NOFORN, NO CONTRACT, and REL markings. They are described in [33] as follows:

> **Originator Controlled (ORCON):** "data marked ORCON may be released only to users belonging to a specified list or organizations. The allowed recipients of the data are specified by the originator of the data ... For any additional users or organizations to gain access to the data, the prior consent of the originator is required."
>
> **No Foreign Nationals (NOFORN):** "a marking used to restrict access to U.S. citizens (in conjunction with the restrictions imposed by the MAC sensitivity label) ... To enforce this control, either the nationality of the user associated with a subject must be known to the access mediation function, or some *a priori* provision must be made to map it to controls that depend on the user ID or clearance, based on the system security officer's knowledge of the users' nationalities."
>
> **REL:** "in contrast to the NOFORN marking, which forbids access to foreign nationals, the REL marking specifies nationalities to whom data can be released."
>
> **NO CONTRACT:** "no access to contractors."

2.3 Bell-LaPadula

D. E. Bell and L. J. LaPadula of the MITRE Corporation developed a formal state transition model of computer security policies that describes a set of access control rules relevant to the military sector [11]. This lattice-based access control model deals with how information flows within a computer system and is primarily concerned with confidentiality. The Bell-LaPadula (BLP) model takes the entities of the computer system and divides them into an abstract set of subjects and objects. Within the model, a system state is defined to be secure if the only permitted access modes of subjects to objects are in accordance with a specific security policy [21, 44].

ORCON has stood out as a policy in the military sector that has been difficult to implement using the classic Bell-LaPadula model [41, 12]. The next major step towards

ORCON came about with the TCSEC and the requirements of mandatory and discretionary access control.

2.4 Mandatory and Discretionary Access Control

Traditional access control mechanisms are primarily based on mandatory and discretionary access control policies. In this section we will introduce both mandatory and discretionary access controls and point out the areas that they are deficient when attempting to handle the problem of owner-retained access control.

2.4.1 Mandatory Access Controls

Mandatory access control provides a means for restricting access to objects, based on the sensitivity of the information contained in the objects and the formal authorization of subjects to access such information [25].

Mandatory access control (MAC), also occasionally referred to as rule-based access control or division B in TCSEC, adds sensitivity labels to subjects and data objects that are used to enforce a set of mandatory access control rules. As defined in the TCSEC, MAC is as follows:

> A means of restricting access to objects based on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e., clearance) of subjects to access information of such sensitivity [21]

MAC is based on fiat; consequently identity is irrelevant. In other words, access is determined by the operating system and an individual user cannot alter the access permissions for an object [13]. Mandatory controls are based on the results of a comparison between the user's trust level and the sensitivity designation of the information [36]. The operating system, or trusted computing base (TCB), authorizes access to a given object if the security level of the subject label is greater than or equal to the label of the object.

2.4.2 Discretionary Access Control

Discretionary access control (DAC), also referred to as identity-based access control (IBAC), or division C in the TCSEC, is a model applied when the user has the ability to allow or deny access to a particular object. It is the most widely known form of access control. We call these controls discretionary because a user or process given discretionary access to information is able to pass that information along to another subject. As defined in the TCSEC [36, 21], DAC is as follows:

> A means of restricting access to objects based on the identity of subjects and/or groups to which they belong. The controls are discretionary in the sense that a subject with a certain access permission is capable of passing that permission (perhaps indirectly) on to any other subject (unless restrained by mandatory access control) [21]

The key to DAC is the identity of the user. The access control matrix model of DAC is most commonly implemented by current operating systems by the following mechanisms: capabilities, profiles, access control lists (ACLs), protection bits, and passwords. These mechanisms are thoroughly defined in [36]. Access control matrices are used to determine whether to grant or deny a particular access request.

DAC differs from MAC because it implements the access controls of the user. It is important that we understand that DAC is not a replacement for mandatory controls; rather, DAC allows us to place finer-grained access controls on resources within a mandatory access policy.

2.4.3 Issues with Mandatory and Discretionary Access Controls

Discretionary access control mechanisms are vulnerable to Trojan horse attacks since the mechanism only restricts access to objects based on the subject identity that is attempting access [33, 36, 28]. When the principle of attenuation of privilege is contradicted, information leakage occurs and the integrity of the system is compromised. Other problems were also noted in [36] based on how DAC is implemented:

- Protection bits lack the ability to easily control access to each object to the granularity of a single user.

- Capabilities and profiles may encounter design problems with revocation of access and group access to objects.
- Passwords are subject to Trojan horses and therefore should not be used unless they are combined with another mechanism.
- Deletion of subjects and objects is a potential problem for any DAC mechanism.

MAC and DAC are also inadequate for efficiently addressing issues such as the NOFORN and ORCON labeling found in people-paper systems [25, 33]. In [25], Graubart also discusses the shortcomings of MAC and DAC policies. After his discussion of MAC and DAC, Graubart also considers the troubling case of ORCON. In addition to the weaknesses in DAC mentioned above, MAC is also informally shown to be a less than optimal fit for ORCON. The possible solution to ORCON with MAC is to have a new category for each additional unique sharing relationship. It is obvious that this is far from efficient when the number of originators and recipients rises. In fact, it is easily possible that the number of required categories could exceed the number of possible categories supported by the underlying security system.

2.5 PAC

Based on the previous discussion, as a possible solution to the ORCON policy Graubart introduces the *propagated access control* (PAC) policy [25]. Similar to DAC, PAC can be maintained in a PAC List (PACL). PACLs are similar to ACLs in that they are associated with an object independent of the sensitivity label. Unlike ACLs, PACLs are only concerned with read access. PACLs also differ from DAC ACLs in that only the originator of the PACL has the ability to change the contents of the PACL. Finally, PACLs propagate to new objects by associating the PACL of an object with a subject when the subject reads the object. From that point onward, if the user creates a new object, the PACL is automatically associated with it. Reads from additional objects cause an ANDing of PACLs, resulting in a PACL that contains the intersection of the two PACLs and lists both originators.

Graubart does not claim PAC as a potential replacement for DAC and MAC policies; instead, he states that they be used in concert when ORCON labeling needs to be enforced.

Rather than introducing new terminology, we have chosen to associate an ACL with each object. These ACLs support a variety of access rights rather than just the read access rights found in PACLs. To protect against the dangers of the documented weakness of DAC with regards to copy threats, we do not associate copy permissions with read access. In addition we approach the problem or ORCON as more than just read access. While one could argue that a person cannot gain access to information unless they have read permissions, owner-retained control is focused on retaining FULL control over the document, this includes permissions such as write and append access control attributes.

2.6 ORAC

The term *owner-retained access control* (ORAC) appears to have been first coined by McCollum et al. [33]. It was introduced as a form of access control, which unlike DAC would prevent access to data being propagated to other subjects without the explicit concurrence of the owner.

McCollum et al. refer to several access control markings including REL, NOFORN, NOCONTRACT and ORCON as specified in the TCSEC [21]. ORCON in particular relates to the problem of owner-retained control. ORCON states that data can only be forwarded to a subject based on the explicit permission of the originator of the data, who is also considered the owner.

According to McCollum, Messing and Notargiacomo, ORAC is a policy that provides a stringent label based alternative to discretionary access control. ORAC is useful in environments where the original owners of the data retain tight control over data when it is propagated (through reading, copying, and merging of data).

Shared ownership is another concern when dealing with owner retained control systems. McCollum et al. have approached this issue in [33]. We will briefly cover their notations through the following example in an attempt to point out some of the difficulties that still persist.

1. Subject LIAM creates a new object, OBJ-1, and places control restrictions on it {ORAC(LIAM: ACL-LIAM)}. LIAM places user LINDSEY on ACL-LIAM and gives her read and write access permissions to OBJ-1.
2. Subject LINDSEY copies the object and calls her new object, OBJ-2. OBJ-2 is now owned by LINDSEY but still carries the control ORAC(LIAM: ACL-LIAM) in its label.
3. As OBJ-2's owner, LINDSEY may modify its contents if she wishes. She also may further restrict access by adding her own ORAC to OBJ-2's label. LINDSEY adds her own ORAC control to OBJ's label. The label now contains {ORAC(LIAM: ACL-LIAM), ORAC(LINDSEY: ACL-LINDSEY)}. LINDSEY places LIAM and OWEN on her ACL-LINDSEY. LIAM is given read and write privileges on OBJ-2, while OWEN is restricted to read privileges.
4. Consider the case where LIAM wishes to modify OBJ-2, this is allowed since read and write access controls exist for LIAM in both ACL's associated with the object.
5. Consider the case where OWEN wishes to read OBJ-2. Although OWEN has been given permission by LINDSEY to read OBJ-2, he will not be granted access because he is not included on ACL-LIAM in OBJ-2's first ORAC control. LINDSEY may not modify ACL-LIAM as she was not the originator of that ACL. LINDSEY must ask LIAM to add OWEN to ACL-LIAM in the label of OBJ-2. If LIAM consents and places OWEN on OBJ-2's ACL-LIAM with a read privilege, OWEN would then be able to access OBJ-2.
6. User LINDSEY is able to disallow LIAM from having access to OBJ-2 by removing LIAM from her ACL-LINDSEY, but she cannot remove LIAM's ORAC control from the object. Note that although LIAM no longer has access to OBJ-2, he still maintains control over his data, because he may still add or delete users from his ACL on OBJ-2.

In this example, derived from the work of McCollum et al., we can see that access for a particular user under the ORAC system is determined by the intersection of the ACL's defined in the objects' labels. We can easily see this in step (5) when OWEN is denied access to the object since he is not in ACL-LIAM.

There are potential issues with this approach. First, what happens when two owners disagree to the access level a particular user should be granted? This situation could easily lead to deadlock situations similar to those found in resource contention scenarios in operating systems. Second, why should LIAM retain any control over the new object? In an attempt to answer both of these questions, we could propose a system where the ACL labels are not applied only to the entire document, but also the portions of the document that particular ACL's apply to. For example, in step (2) above, rather than simply append ACL-LIAM to the label of OBJ-2, we could have her ACL apply to only the section of the document that contains her data. With this in mind, what will happen with OWEN in step (4)? It is obvious that LINDSEY wishes that OWEN be able to read the document, LIAM decides not to grant OWEN access to her object data; therefore, we should grant access to only that data which belongs to LINDSEY. We will consider this proposal further when we present the ORCS model.

McCollum et al. also summarize other concerns and problems with their approach [33] such as implementing the features in an environment where the trusted computing base is in question, storage and performance overhead of the additional information, and concerns that ORAC controls could lead to a system in which only the original owners would have access to their data.

If we think of shared ownership as a tree structure, where the root node is the owner of the current object, with each child node an ACL of a shared owner, we can determine which portions of a document should be granted access and to what degree for each user request.

We agree with McCollum et al. regarding the use of owner-retained control in combination with traditional MAC, and dissemination controls. We recommend, however, a different approach to applying these controls.

The ORCON marking is addressed by the ORAC model. Although it could be argued that all of the access control markings listed in the TCSEC could be rules that an owner assigns to a protected object. The qualifying attributes that need to be checked could be integrated into digitally signed certificates. The association of ACL with every object is an interesting approach, as it provides a model that helps ensure data leakage does not occur without the permission of the owner at a fine-grained level. This approach, however, would lead to two potential problems over time. First, as more users share information ACL's would grow out of control and consume too much storage space. Second, it is easy to imagine that as time progresses and subjects work together, differences in opinion are bound to happen. These conflicts between subjects and their views on who should have access to data could lead to deadlocks over the information in contention. That is to say, humans do not always agree on who should get what, and this can lead to no one getting anything.

In this thesis, we approach the first issue by maintaining the ACL at the document-level rather than the object-level. In the future, it would be interesting to look at hybrid approaches. The second issue is only partially addressed by our work. Currently, we allow multiple owners of an object and transfer of ownership, but we do not attack the issue of conflict resolution between owners.

2.7 ORGCON

Abrams et al. introduced organization controlled (ORGCON) as an example policy used in their prototype of generalized framework for access control (GFAC). GFAC was introduced as a framework for studying and constructing access control policies in automated information systems (AISs). ORGCON was chosen for the name of the policy so that they could represent the concept of owner-retained control without infringing on the Government's definition of ORCON [1].

After a discussion of the limitations of existing mechanisms and policies, focus is moved to exploration of the ORCON labeling and the creation of an ORGCON policy. The goals of their prototype were to demonstrate a prototype based on the GFAC concepts and to implement an access control policy, namely ORGCON, in addition to MAC and DAC. Abrams et al. chose to implement their prototype within a system that already provided a B-level MAC policy, namely, System V/MLS. As a result, their implementation was tied to the existing DAC and MAC mechanisms of the underlying system. The discretionary controls provided were for the user/group/other (also known as Owner/Group/World or O/G/W) mechanism. Recall that in standard UNIX environments, permissions for read, write, and/or execute (for files) and search (for directories) may be assigned to each role.

Abrams et al. also discusses the difficulties and tradeoffs that they incurred while developing their prototype within the constraints of the existing MAC and DAC. In particular, they had to overcome the limitations of the existing structures such as the DAC access control list (ACL) and O/G/W bit mask or similar mechanisms. In order to overcome the uncontrolled copy and lack of accountability weaknesses in the DAC mechanisms, they chose to restrict user access to ORGCON objects to a limited set of functions.

In contrast to ORGCON, our approach does not depend on underlying access control mechanisms or policies. In our current implementation the policy and mechanism exists independently of the underlying operating system access controls.

ORGCON identifies two primary roles for access control: ORGREP and RECREP. The view of the ORGCON policy is that individuals do not own data; rather, organizations do. With this view, they then assign individuals or daemons to fulfill the role of organization control representative (ORGREP) and recipient representative (RECREP). The ORGREP is responsible for marking information as ORGCON and assigning a distribution list. A RECREP is responsible for assigning individuals who are authorized to receive the ORGCON information within the receiving organization.

Our approach in ORCS allows for a slightly expanded and more flexible view, where any individual can be the owner of information. As such, they have the ability to modify the access control list themselves. Another feature of ORCS is the ability to transfer ownership of a particular resource to another individual. This can be thought of as equivalent to the ability of an organization assigning an individual to act as the ORGCON role and later assigning a different individual. We, on the other hand, perform this at an individual level; consequently, we have tighter accountability.

2.8 TAM

The *typed access matrix* (TAM) model has risen out of the need to have stronger safety properties. It introduces strong typing into the access matrix model as formalized by Harrison, Russo, and Ullman (HRU). It is formally defined in [41].

Sandhu approaches the ORCON policy in TAM by proposing a solution based on the ability of TAM to have multiple parents jointly create a child subject. It focuses only on preventing a subject from copying a file and then giving another subject access to the copy. The discussion in [41] is brief in its exploration of ORCON and does not cover any implementation details. It seems unclear what would happen to a document should it be removed from the system and transferred to another system that does not comply with the TAM model. We address this issue by providing a platform independent prototype that is used to enforce owner-retained access control policies embedded in the document. We also expand owner-retained control from ORCON to handle more than just read access attributes.

2.9 Role-based Access Control

Role-based access control (RBAC) has been called the dominant access-control model of the 1990's, replacing MAC and DAC, which were dominant in the 1970's and 1980's. RBAC, first introduced by Ferraiolo and Kuhn, is an access control mechanism that is policy neutral, yet policy oriented [46, 23]. Ferraiolo and Kuhn credit the need for a model that can efficiently address the security issues related to processing unclassified sensitive information, since TCSEC and other policies focus on the US Department of Defense (DoD) need to protect access to classified information. Civilian governments and corporations, who have unique security needs and policies that cannot easily be met using traditional DAC and MAC models, can use this new model that is being standardized by the NIST [24].

While DAC and MAC are recognized by official government documents, researchers and practitioners have found that they do not cover many practical requirements [21, 23]. Role-based access control (RBAC) policies are based on the activities that the subjects execute on the system. In order for this policy to be used, roles must be identified on the system, and then roles have to be assigned to subjects. The roles, occasionally referred to as groups, can be defined as a set of responsibilities and actions related to a particular working activity [43]. In RBAC, users cannot pass access permissions onto other subjects at their discretion; a possibility in DAC based systems.

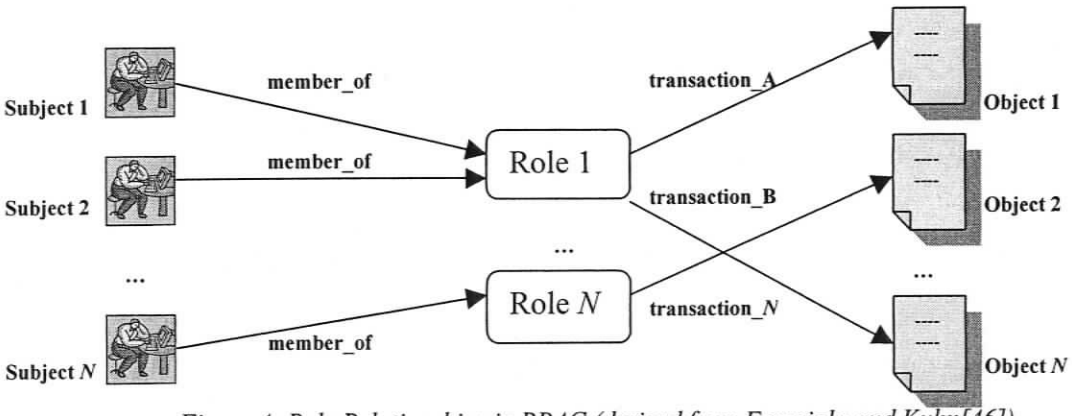


Figure 4: Role Relationships in RBAC (derived from Ferraiolo and Kuhn[46])

In Figure 4 we visualize the basics of role based access control. Subjects are assigned roles independently of the assignment of roles to transactions on objects. Within the

RBAC model, it is possible for a subject to play different roles at different times, or in some implementations, more than one role at the same time. When a subject assumes a role they have permission to perform all accesses for which the role is authorized.

Several advantages of the RBAC model have been identified [43, 23]:

- **Authorization management:** due to the logical independence in specifying subject authorizations in two parts. (1) Assign access rights for objects to roles. (2) Assign subjects to roles.
- **Hierarchical roles:** due to the nature of roles in everyday life, it is natural to generalize and specialize these relationships into a hierarchical structure. In doing so we can build a role inheritance hierarchy, thus simplifying authorization management.
- **Least privilege:** allowing a subject to sign on with the minimum privilege necessary to perform a given task reduces the danger of unintended errors or attacks.
- **Separation of duties:** allowing a task to be separated among different subjects. Based on the principal that no single subject should be given enough privileges to misuse the system on their own.
- **Object classes:** allowing objects to be classified into groups, similar to how we group subjects into roles. Access authorization of roles can then be based on object classes rather than individual objects, thus simplifying authorization management.

While RBAC does not directly deal with the issue of ORCON and owner-retained access control, it is interesting to consider combining the two. In the future, it would be interesting to move the definition of owner-retained control to a higher level. For example, in the UNIX environment the root account has absolute authority over all subjects who have privileges to use the system. This can be a privacy concern. If we were to define a system where subjects could choose to retain control of their objects, as we define later in this text, using roles and groups we would have a better ease of use and be able to reduce the length and complexity of access control matrix entries. It would, however, also require the standardization of access control attributes, generic roles, and a grammar definition that specifies how the rules must be associated.

2.10 UCON

Usage control (UCON) has its origin in the superdistribution paradigm that was introduced by Cox [39]. The work on UCON has interesting properties that are similar to

some of the approaches we have made in our work [39]. Park and Sandhu have done significant work and have introduced UCON as being a mechanism where information is freely distributed; however, access to the information remains restricted. They claim to have been the first to produce a systematic study of how ORCON can be combined with UCON. They use the term "digital container" to refer to an encapsulating wrapper which uses cryptography to protect the information. This is similar to the approach we have developed. They also use the term "virtual machine" to refer to the special software or hardware that is responsible for enforcing the access control restrictions specified in what they call the "control set".

ORCS is an implementation that could be compared to the UCON virtual machine; however, a virtual machine is supposed to represent an abstraction of a machines underlying hardware. We prefer to call ORCS a portable "sandbox" rather than a virtual machine, as it restricts information from flowing outside and also is responsible for enforcing the access control rules specified for a given object. A principle difference between ORCS and UCON is that UCON requires a "control center" that is responsible for controlling and managing the access rights, usage rules, and usage history of the subjects and objects. The ORCS model does not require such a mechanism, but it can integrate a control centre for certain applications. By not requiring a central server mechanism to be part of the framework, we are able to use the model on devices that are not connected to a network. UCON also presents the interesting prospect of being able to represent multiple security policies using the UCON model and warrants future attention. In conclusion, Park and Sandhu have not provided an implementation, but rather recommend that further research on these aspects be performed to find more practical and comprehensive solutions to digital information dissemination controls [39].

2.11 Digital Rights Management (DRM)

Digital rights management (DRM) also has its origin in the superdistribution paradigm that was introduced by Cox [19]. The primary focus of DRM has been on producing financial gain through the dissemination of digital content protected by payment-based controls in an attempt to thwart copyright problems. Once a digital object has been

disseminated the access to, and usage of, the digital content is granted once a fee has been paid to the originator or supplier of the content. DRM has been an active area of research where many issues such as proprietary implementations, and narrowness of scope have arisen [39, 40].

Ericson, Lesk, and many others are starting to raise serious questions concerning side effects of DRM implementations [22, 30]. These issues include who controls access to what information, privacy and the collection of personal information, effects on existing legislation such as copyright and fair-use, as well as what services will potentially cease to exist versus what new ones will surface. While the main driving force behind digital rights management is undoubtedly the commercial sector, the ability to place restrictions on digital content is useful, given the correct circumstances, to most people.

Lyon discusses many of the problems of existing approaches to dealing with DRM in [32]. He suggests that future DRM solutions have a tighter integration with biometric devices and expand the roles of third parties in substantiating the credentials and reputations of individuals. ORCS takes this approach by requiring that all subjects be authenticated by means of a trusted third party, namely the certificate authority. Biometric devices could be integrated into the ORCS mechanisms by replacing the password-based authentication used for the retrieval of a secret key with a biometric implementation.

Digital rights management is tightly coupled with the concept of access control and security policies. In fact, Park and Sandhu have introduced the UCON framework that attempts to merge them all into one [47].

The CISD division of the NIST has identified DRM as a critical issue that must be addressed (http://www.itl.nist.gov/div895/drmmain.html). As noted by Park and Sandhu [39] ORCON could provide a useful form of DRM for not only the dissemination of information for commercial gain, but also for non-commercial use. The access control attributes of the model could easily be expanded to include those offered by DRM solutions.

2.12 Uses of Access Control and DRM

There are many existing examples of access control and several DRM in use within the popular software domain. We will briefly introduce three pieces of mainstream software to relate some of the previous concepts to industrial use.

2.12.1 Access Control in Windows

As remarked by Swift et al., Windows NT 4.0 only supported file and directory based access control through the use of access control lists (ACLs). The permissions in these ACLs were also limited to a very small set of allowable/disallowable actions such as read, write, and execute. When the move was made from Windows NT 4.0 to Windows 2000, several new access control mechanisms were introduced [50]. One of the major additions to Windows 2000 was Active Directory (AD). The active directory allows for hundreds of types of objects, not just files and directories, and each object has the possibility of having multiple properties to be protected separately. The work of Swift et al. resulted in the discovery of several limitations in the Windows NT scheme:

- No support for multiple operations or properties for individual objects
- No support for having multiple object types within a container
- No support for the propagation of changes to ACLs through a tree of objects

After the addition of AD, Microsoft has since moved forward to add new security features to their operating systems to protect documents. Rights Management Services (RMS) for Windows Server 2003 has added new features to protect against proposed vulnerabilities that exist in perimeter-based security methods.

Perimeter-based security methods are mechanisms such as firewalls, access control lists (ACLs), and encryption mechanisms intended to restrict access to secured information. As Microsoft points out, however, these mechanisms do very little to protect the information once a recipient receives the data. There needs to be a way to enforce business rules (or security policies) on people who use and distribute the data outside of the organizational perimeter. This is where the ORCON type controls that we have discussed can come into play, and where Microsoft introduces RMS.

Digital rights management (DRM) in Microsoft Windows currently requires the use of an end-to-end Microsoft solution (Service, Rights Management, and Office). Its ability to restrict use of certain features when a user attempts to access a Word document only works in certain versions of the software, such as Microsoft 2003. Ideally, we would like to see the ability to add digital rights management to any existing environment.

The introduction of both DRM protocols and interchange formats, combined with legislation and certification bodies, in the future could potentially allow corporations to create solutions for commercial gain, while creating a platform independent solution.

2.12.2 Access Control in LINUX

More recently, progress has been achieved in making Linux more secure. Two highly configurable general frameworks for access control have been created. One is based on the GFAC discussed with the ORGCON proposal above, and the other was originally called FLASK. These two frameworks have both been implemented as Linux kernel patches for the Linux operating system.

Security enhanced Linux (SELinux), under the direction of the NSA [http://www.nsa.gov/selinux], is a research kernel and set of utilities aiming to provide an environment that supports the enforcement of many kinds of mandatory access control (MAC) policies. It is derived from the earlier FLASK system. These policies are based on concepts such as role based access control (RBAC) and multi-level security. Beginning with the 2.2 Linux kernel, SELinux has grown through the 2.4 kernel versions and is now available in 2.6. Currently it does not appear that SELinux has an ORCON policy implemented.

ORGCON was conceived from the generalized framework for access control (GFAC) that was first introduced by Abrams et al. in 1990 [3]. It has more recently been implemented in the creation of rule set based access control (RSBAC) for the Linux operating system [38]. While RSBAC implements many models, it does not appear to currently support ORCON markings.

With both Linux and Windows acting as major operating systems in the marketplace it would be useful to have policies that can be implemented consistently across multiple platforms. ORCS is a model and prototype that has assisted us in determining the feasibility of such a mechanism.

2.12.3 Adobe® Portable Document Format

Another more recent addition in the data protection arena is the added security feature of Adobe® portable document format (PDF) [4]. The newer PDF document supports the detection of document alterations, protection from unauthorized access, prevention of authors or senders from refuting documents, and the ability to ensure ongoing document integrity using digital signatures. These features are available with Adobe Acrobat 6.0. Example restrictions include the ability to restrict access based on a user password or master password, as well as restricting the following actions:

- Printing, Changing the Document
- Content Copying or Extraction
- Authoring Comments and Form Fields
- Form Field Fill-in or Signing
- Content Accessibility Enabled
- Document Assembly

Data can still be retrieved from the document, however, as certain protection mechanisms can be bypassed. For example, content copying can still occur through the use of non-Adobe tools. There are several ways to extract information, including copying the document content to a BMP file or using another third-party tool. Another noted exception to the security was found when looking at documents marked with the Printing (Not Allowed) restriction. By saving the document to disk and opening the document using third-party tools, this restriction was also bypassed.

This modern implementation provides many attractive features for security; however, it also exposes the danger of having third-party tools that can extract

information from documents without applying the implied security requirements that the document specifies.

The main problem with this approach to DRM and access control is that it is limited to only PDF documents. A more general model would be attractive to protect all types of information. ORCS derived models could provide such a mechanism.

From our brief discussion of these three products, one unifying issue presents itself: How can we provide owner-retained access control independently from the underlying architecture for all types of data?

Chapter 3 Proposed Solution to the Owner Retained Control Problem

From the original definition of ORCON, we could choose to design the model several different ways, as we have just seen in the related work. In this section, we will approach the design of an owner-retained control model by separating the concepts into two parts.

First we will build a basic model, which considers only one subject, or entity, as being the owner of the object. This section is broken down into two subsections: the policy model which defines what is and is not allowed, and the enforcement model which moves toward how the policy model could be implemented. Second, we will consider an extension to the basic model that can allow for support of shared ownership of objects between two or more subjects, or entities.

3.1 Basic Model

In order for us to analyze policies and compare the features of ORCS against other models and tools, we need to start with the basic model. The following theoretical model follows concepts derived from Bell-LaPadula (BLP) [11], The Chinese Wall Security Policy [15], and work done by McCollum et al. in the area of dissemination control and owner-retained control (often referred to as originator control) [33].

The ORCS framework attempts to take a step towards a more flexible security model, where an owner-retained security policy can be placed on objects in such a way that it must be enforced wherever the data travels. Understanding the domain of the problem involves the definitions of several primary sets. We will adopt the BLP representation of objects and subjects in our model, so that we have a similar language

for comparison, and then add new notations as need arises. In the following sections we will start with a policy model, essentially representing the domain of the ORCS theoretical model, and then expand this model to include an enforcement model in which we explore some mechanisms to enforce the basic model.

3.1.1 Policy Model

Owner-retained control (ORC) provides a means for a subject to retain control over its objects within its environment and when the object leaves its environment. ORC can act as an extension mechanism, augmenting existing access control policies, allowing the enforcement of access control in a distributed heterogeneous environment. The extension allows for, among other things: owner retained control, shared ownership, and transfer of ownership. We will start by defining what is meant by owner-retained control and then define a set of rules such that no subject can, under any circumstances, access objects that are not within the defined set of allowable permissions for that subject on a given object. In the following section, we will first define the sets and terms that are necessary to provide for ORC in a multiple-subject, unary-environment model. Later in the section we will explore model expansions for multiple environments, groups and roles.

The basis of owner-retained control is that other subjects (other than the owner) are only able to obtain the access privileges that have been defined by the owner of the data. The set of all subjects is representative of all users and processes that exist in the environments. Modern symmetric multi-processing systems also make the distinction between a process and a thread. Since processes are treated as entities acting on behalf of a subject, it is warranted to make the further distinction that a thread acts on behalf of a process.

Definition 1:

The set of all *subjects*, including: subjects, processes, and threads.

$$S = \{s_1, s_2, \ldots, s_m\}$$

All information is stored in an object (examples include files, databases, packets, and devices). For the sake of ORC we will define an object as being data, whatever its form.

The set of all objects represents the data that is to be protected in the environments and will be contained in ORC wrappers.

Definition 2:

> The set of all *objects*, including: data, files, programs, subjects, I/O devices, and network connections.
>
> $O = \{o_1, o_2, \ldots, o_n\}$

For the purpose of translating various access attributes into mathematical statements, we will also define what is meant by a *protected memory area* (for example, a clipboard) as it pertains to the operations of *cut*, *copy*, and *paste* for temporary operations performed by subjects against objects.

Definition 3:

> A *protected memory area (PMA)* will be defined as a reserved area of memory that may be used for the transfer of data to/from an object. In the case where the *PMA* is private to the application, we will define the following:
>
> $PMA_{private}$
>
> The alternative is when the *PMA* is public and may be used for the transfer of information between objects:
>
> PMA_{public}

Access attributes, as defined in BLP, are not sufficient for scalable collaborative environments, so we have expanded the set as follows. This is a sample set and could contain more attributes than are defined here.

Definition 4:

> The set of all *access attributes* is representative (security labels) of all possible access attributes that may be assigned to an object on a per subject basis.
>
> $A = \{rd, wr, ap, ex, co, cu, cp, ps, dl, nb, na\}$
>
> These are defined as follows:
>
> | | |
> |---|---|
> | rd: | read (view from anywhere in object) |
> | wr: | write (add anywhere in object allowing both altering and viewing capabilities) |
> | ap: | append (add to end of object without allowing read access) |
> | ex: | execute (execute the object without altering or viewing capabilities) |
> | co: | control (have control over the object) |
> | cu: | cut (delete from object and place on system clipboard) |
> | cp: | copy (copy from object and place on system clipboard) |
> | ps: | paste (insert into object from system clipboard) |

dl:	delete (remove from object)
nb:	not before (deny access to object until specified date-time)
na:	not after (deny access to object after specified date-time)

Each of these access attributes will now be further expanded for clarity.

Definition 4.1:

The *read* (*rd*) access attribute allows only the viewing of the data contained in the object by the subject.

Definition 4.2:

The *write* (*wr*) access attribute allows the modification of data anywhere in the object by the subject, and implicitly implies read and delete attributes.

Definition 4.3:

The *append* (*ap*) access attribute is fundamentally different than the write attribute. Append only allows for the addition/writing of new information at the end of the data contained in the object by the subject. Append does not imply read or delete access attributes.

Definition 4.4:

The *execute* (*ex*) access attribute allows the execution of object content by the subject for those instances where the data of the object is executable.

Definition 4.5:

The *control* (*co*) access attribute allows for the *ownership* of the data contained in the object. This is the most important attribute in the context of owner retained control, as it dictates which subject(s) has/have the ability to control who has access to the object, and which access control attributes the subjects will have.

Definition 4.6:

The *cut* (*cu*) access attribute allows for the deletion of data contained in the object by the subject; however, the data is still available for future operations since the cut operation will place the data in a reserved area of memory known as the *PMA*.

Definition 4.7:

The *copy* (*cp*) access attribute allows for the duplication of data contained in the object by the subject, making it available for future operations since the copy operation will place the data in a reserved area of memory known as the *PMA*.

Definition 4.8:

> The *paste* (*ps*) access attribute allows for the insertion of data by the subject into an object from data stored in a reserved area of memory known as the *PMA*.

Definition 4.9:

> The *delete* (*dl*) access attribute allows for the removal of data from an object by the subject.

Definition 4.10:

> The *not before* (*nb*) access attribute allows for the restriction of access to the object, by the subject, until a specified time has come to pass.

Definition 4.11:

> The *not after* (*na*) access attribute allows for the restriction of access to the object, by the subject, after a specified time has come to pass.

Certain attributes warrant future consideration on how they should be used. For example, the *not before* and *not after* attributes could be applied to the object as a whole or could be applied against the object on a per subject basis. Again, this is only a sample set of attributes.

In all of the following definitions, propositions {theorems, lemmas, corollaries} and axioms, assume the following:

> Let subscripts *a, b,* and *c* be arbitrary elements in some set.
> Let S be a set of subjects, O be a set of objects and A be a set of access attributes.

The set of all permissions represents the combinations of access attributes that subjects are granted/denied on objects.

Definition 5:

> The set of all *permissions* (security labels) is the power set of A
> $P = \{P_1, P_2, \ldots, P_k\}$, where an arbitrary element $P_a \subseteq A$
> The union of all permissions is A
> $\bigcup_{i=1}^{k} P_i = A$

The next step, now that we have the concepts of subjects, objects, and permissions is to identify how to relate sets of access attributes (using permissions) to an object for subjects. The concept of the access control matrix model is considered to be the most precise model used to describe the protection state [36]. The matrix characterizes the rights of each subject with respect to every other entity.

Definition 6:

> Every object (o_b) has an *Access Control Matrix (ACM)* that is an n x 1 matrix, containing permission (P_a) for each subject (s_a) for the said object. We introduce the function $F_1(s_a, o_b) = P_a$. This function determines the set of permissions that the subject s_a has for object o_b.

The definition of an *ACM* in the context of a single object is for the intended purpose of allowing an object to stand alone, rather than being defined as an *nxm* matrix representing subjects and objects. Using our definition the model should be more portable to applications, such as databases, since the *nxm* ACM is easily formed using our definition on a per object basis.

To understand owner-retained control, a distinction must be made between three key terms: to have *possession* of an object, to have *ownership* of an object, and to have *access* to an object. To clarify, we reiterate that the data is what we are trying to restrict access to. In many existing models and implementation, the terms ownership and possession are not clearly separated. We define these terms as follows:

Definition 7:

> A subject (s_a) is considered to have *possession* of an object (o_b) when it is stored on a storage medium that they have ownership of. This does not, however, necessarily indicate that they have *access* to the object (o_b) or more importantly the data contained within.

Ownership is defined in the Orange Book [36] as, "a concept in which one user has total control over access to an object. A subject operating on behalf of that user is normally the creator of the object and is totally responsible for controlling access to it". This, however, is the definition pertaining to discretionary access control (DAC) and does not take into consideration the difference between having possession of an object and having access to the data within it. In much of the literature, ownership is used in place of possession; however, this should not be the case. Simply because a subject has

possession of an object does not mean he owns it. Ownership of an object may or may not imply possession. As a simple example, a person who owns a vehicle has loaned that vehicle to a friend. The friend, while borrowing the car, has possession of the vehicle but not ownership. As a simple demonstration of this fact, the friend does not have the right to sell the vehicle that he does not own. We will define ownership of an object for our purposes as follows.

Definition 8:

> A subject (s_a) is considered to have *ownership* of an object (o_b) when the subject has the control access attribute (co) in their permissions:
> $F_1(s_a, o_b) = P_a$, and $co \in P_a$

To assist in our understanding of who can potentially use an object, we distinguish the concept of access as being separate from ownership.

Definition 9:

> A subject (s_a) is considered to have *access* to an object (o_b) when the subject has an entry within the *ACM* of the object. This does not guarantee ownership, rather it states there will exist an entry P_a in the ACM for subject s_a.
> $F_1(s_a, o_b) = P_a \neq \emptyset$

Now that we have a basic understanding of what is meant to have ownership, possession, and access, we will further our ORCS policy model by adding in the abstract components that will provide for two of the cornerstones of security: confidentiality, and integrity. Confidentiality concentrates on information concealment. As such, an ORC system should keep information confidential when valid subjects are not using the object.

Definition 10:

> The information contained in an object (o_b) has been kept *confidential* when only the owner (see Definition 8) and those subjects who have at least read access (see Definition 9) to the information can obtain it.

When the data is stored, it should be impossible to gain access to the data by a subject who has possession of the object but has not been granted access to it.

Integrity comes in two forms, that of data integrity and that of origin integrity. Data integrity is concerned with ensuring the content of the information maintains its integrity. Origin integrity focuses on the source of the information having integrity (often

called authentication)[13]. While confidentiality focused on protecting access to the object, integrity concerns itself with both the correctness and trustworthiness of the information contained in the object.

Definition 11:

> The information contained in an object (o_b) has maintained its *data integrity* when proper prevention mechanisms are in place to ensure that no unauthorized subject has been able to change the data contained in the object, thus ensuring correctness of the data.

Definition 12:

> The information contained in an object (o_b) has *origin integrity* when proper authentication mechanisms are in place to ensure that we can trust that only authorized subjects, with an access attributes allowing change, can modify the data contained in the object.

For us to continue the expansion of the model, we now need to consider what would be needed to maintain the model and expand its usage to multiple heterogeneous environments. Since previous models, such as BLP, focus on the security of a single environment, we need to consider several additional sets that will allow us to expand our model to include scalable collaborative environments.

The set of all environments is representative of the runtime environments where subjects interact with objects. The environment is responsible for the enforcement of the security policy through implemented mechanisms.

Definition 13:

> The set of all *environments* (operating systems etc.)
> $E = \{e_1, e_2, \ldots, e_j\}$

In a distributed heterogeneous environment we would need to also make a distinction between what security levels exist. Each level will display particular attributes that would certify it as being of a specific security level, thus allowing for an expansion of the model, allowing owners to specify that the object may only be accessed from within an environment that meets the requirements of security level x.

Definition 14:

The set of all environmental *security levels* that exist in all environments may be categorized as

$$SL = \{sl_1, sl_2, \ldots, sl_n\}$$

However, if it were possible to have the same environment available on top of the heterogeneous environments, the security levels may not be necessary.

To further expand our model to a larger working set of multiple heterogeneous environments we will add to our concept of subjects, the additional recognized concepts of groups and roles, thus allowing access control to be restricted by group permissions and subject roles. The set of all groups is representative of the different groupings of users that exist in the system. Only a trusted security administrator, such as a certificate authority, may assign groups and roles to subjects.

Definition 15:

The set of all *groups*

$$G = \{g_1, g_2, \ldots, g_q\}$$

The set of all roles is representative of the different roles a subject may be associated with when attempting to gain access to objects.

Definition 16:

The set of all *roles*

$$R = \{r_1, r_2, \ldots, r_u\}$$

The concepts of groups and roles would enable the model to create ACM entries that are easier to maintain. Groups would allow rules to be defined that gather a set of subjects into one rule. Roles would allow rules to be created that group together a set of actions into one role and then assign roles to subjects.

In order for a security administrator to place subjects into groups and roles, it is necessary for the administrator to have a set of classifications for subjects that would assist in the decision making process. Classifications can be used in enforcement to specify the clearance levels of subjects and the requirement level for objects. The set of all classifications is representative of the clearance level of a subject, and the classification of an object [11].

Definition 17:

The set of all *classifications*
CL = {top secret, secret, classified, … }

Groups, roles, and classifications allow us to extend the model to simplify use and to expand the permissible complexity of security rules. Groups will allow access rules to be applied on a broader basis, simplifying user administration. For example, we can have a single rule stating that *read* access will be granted to the entire group of accountants in a firm, rather than specifying one rule per accountant for a particular object. Roles will allow a subject to perform requests based on roles that the subject is allowed to act as. For example, in intelligence organizations an attribute specifying citizenship might dictate what role a particular user is playing and thus is used in determining access.

Using the above definitions, we will now introduce several axioms before we begin discussion of the enforcement model. In all of the following definitions, propositions {theorems, lemmas, corollaries} and axioms, assume the following:

Let subscripts a, b, and c be arbitrary elements in some set.
Let S be a set of subjects, O be a set of objects and A be a set of access attributes in an arbitrary environment E_b.

Axiom 1:

Once the environment has authenticated a subject (s_a), the only permissions he will be granted for an object (o_b) are those permissions specified in P_a.

Permission may be assigned to group or role; however, authentication is still based on the user. Consequently, subjects may also be groups or roles.

Axiom 2:

If a subject belongs to a group, and that group has been granted access, then the subject has access with the group permissions to the object.

Axiom 3:

If a subject is acting under a particular role, and that role has been granted access, then the subject has access with the role's permissions to the object.

Before moving onto the concept of shared ownership, let us first make a few additional observations. ORCS, as a theoretical model, at the application layer can have a system

that enforces security derived from a policy that is built into the object itself. This model can be expanded to include implementation at the operating system level/file system level, and potentially at the hardware level. Consequently, you can have higher levels of trust/reliability in the operating environment since the enforcement framework resides at a lower, and therefore more secure, level.

At the operating system level you are protected by the protection rings of the processor, thus restricting malicious activities in user space. At the hardware level, you are protecting the mechanism from all malicious individuals except for those who can obtain physical access to the computing resources.

Operating environments that are said to meet a certain criteria with regards to security are often called trusted computing bases. Trusted computing bases have fundamental flaws in distributed collaborative environments with regards to security compliance. Existing systems/environments will have security features in place that will undoubtedly be either sufficient or insufficient in regards to the security policy that the object in question must comply with. It would be useful, therefore, to have "mobile security policies" to allow for rules stating that the environment must support at least the minimum stated mechanisms in order for a subject to access the object. We use the term "mobile security policy" to indicate that the security policy is tied to the object, potentially in transit, rather than tied to the platform that enforces it. A sub-environment can be used temporarily to support the minimum requirements until support is fully integrated into the environment.

The ORC framework is an example of how such a policy could be enforced in any environment; this is similar to the purpose of the Java Virtual Machine. Using the Java programming language we can implement libraries and an application that enforce the ORCS model at the application layer. This approach temporarily provides a sub-layer or sandbox for accessing the objects with the access-control features supported until features can be implemented at lower layers. Security, however, cannot be claimed for a single user-space application unless the underlying layers of the system on which it is executing are also proven to be secure and trustworthy.

There is a need for future research to enable the formal proving of operating systems compliance with security policies. This could entail the creation of testing frameworks that can be dynamically inserted and removed from running systems, compiler and executable file format revisions that would allow for the checking at runtime for integrity of executables, and runtime checks of operating systems to ensure environment integrity and security compliance. These additions/changes may allow for more secure computing bases in which applications may execute.

3.1.2 Enforcement Model

In order to protect the data and the access control matrix, we need mechanisms to ensure confidentiality, and integrity. Encryption techniques are the most common form of protection and thus provide a form of confidentiality that will protect the object from unwanted subjects. These techniques will include both symmetric and asymmetric encryption techniques. First we will define the asymmetric key sets followed by the symmetric set.

The following definitions for asymmetric and symmetric keys, key-ring, and certificate authorities will be used to fulfill the confidentiality and integrity requirements of definitions 10, 11 and 12 of the policy model above.

First, to satisfy Definition 12 above, we define a form of asymmetric encryption typically based on a public key algorithm, such as RSA, that will be used for authentication purposes.

Definition 18:

> The set of *key-pairs* is representative of all key pairs that exist in the environment. There is a one-to-one mapping, with respect to the environment, of key-pairs to subjects.
>
> $K = \{K_1, K_2, \ldots, K_m\}$
>
> Each key-pair consists of a public key and a secret key, which are related mathematically (such as in RSA). For an arbitrary key-pair K_j we would have:
>
> $K_j = \{pk_j, sk_j\}$

Definition 18.1:

The set of *public keys* is representative of all public keys from the key-pairs in K.

$$PK = \{pk_1, pk_2, \ldots, pk_m\}$$

Definition 18.2:

The set of *secret keys* is representative of all secret keys from the key-pairs in K.

$$SK = \{sk_1, sk_2, \ldots, sk_m\}$$

Each subject needs the ability to store their key-pair and a set of public keys within their environment.

Definition 19:

The *keyring* for a subject s_a is therefore defined as the following:

$$KR_a = K_a + B, \text{ where } B \subseteq PK$$

The set of keys on the keyring (KR_a), for a given subject (s_a), is representative of a collection consisting of one key-pair (K_a) and a collection of public keys from PK. While it is possible for a subject to have more than one key-pair associated with them, we are only concerned with those keys used in the ORCS environment. In this case, each user should be assigned only one key-pair for authentication and integrity purposes.

Next, in order to satisfy Definition 10 above, we implement a form of symmetric encryption to ensure data confidentiality.

Definition 20:

The set of all *symmetric keys* ($SM_{\#}$) is representative of all symmetric keys, using a given algorithm, of size #.

$$SM_{\#} = \{sm_1, sm_2, \ldots, sm_x\}$$

Authentication and the creation of a trusted computing base often requires a level of trust to ensure that subject s_a can believe that subject s_b is who he claims to be. Thus we introduce the concept of a trusted certificate authority that is responsible for registering subjects and guaranteeing the authenticity of all subjects possessing asymmetric keys. The set of all trusted certificate authorities is responsible for evaluating subjects and their credentials in order to certify their identities.

Definition 21:

The set of all *trusted certificate authorities*
$T = \{t_1, t_2, \ldots, t_w\}$

In determining whether to grant access to an object, it is useful to qualify subjects based on their qualifications or credentials. These credentials can be used to fulfill the requirements of classification defined in Definition 17 above. These credentials can be used in the generation of public key certificates. The certificates can then be used by access control mechanisms and for the delegation of subjects into group and role categories. The set of all credentials is representative of the properties associated with a subject and can used in validating their privilege level.

Definition 22:

The set of all *credentials*
$C = \{c_1, c_2, \ldots c_z\}$

Finally, we are able to introduce notation to represent the access control matrix that will be used to store entries that specify which subjects will be granted particular permissions. We choose to represent the ACM (see Definition 6) as an access control list (ACL). Access control matrices are traditionally implemented as either access control lists or as capability lists [13]. The difference between them is their ability to answer the following questions:

- *When given an object, what subjects can access it, and how?*
- *When given a subject, what objects can it access, and how?*

Access control lists provide the easiest way to answer the first question, while capability lists provide the easiest for the second. Since the ORCS model is primarily concerned with the question of which subjects can access the protected object, it is natural to choose access control lists for the enforcement model.

Definition 23:

Every object (o_b) has an *Access Control List (ACL)* that is an n x 1 matrix, containing permission (P_a) for each subject (s_a) for the said object.
We introduce the function $F_1(s_a, o_b) = P_a$. This function determines the set of permissions that the subject s_a has for object o_b.

Postulate 1:

> An ACL entry for object o_b can only be added for subject s_c if, and only if, there is a valid public key pk_c in the owner's keyring.

We consider what is meant for a subject to be trusted with respect to data access contained within our objects.

Definition 24:

> A public key is considered *trusted* if, and only if, the public key (PK_a) is digitally signed by a trusted CA (T_c).

Furthermore, a trusted subject is one whom has been added to the keyring of the owner.

Definition 25:

> A subject (s_b) is considered to be a *trusted subject* if, and only if, its public key (PK_b) was added to the owners (s_a) keyring (KR_a) via a digitally signed certificate from a trusted CA (T_c).
>
> $PK_b \in KR_a$

For every subject s_a there will exist an associated keyring KR_a that contains the key-pair for the subject (K_a) and a collection of public keys from PK. The keyring for s_a is protected, for example, by a password protected symmetric encryption scheme.

There exists only one subject (s_a) that has the necessary secret (password, etc.) to insert a new public key pk_c into the keyring (KR_a) of s_a.

Theorem 1:

> All subjects in the ACL of an object are considered trusted.
>
> Proof:
>
> > For all subjects in an ACL there is a digitally signed public key in the keyring of the owner of the object (by Definition 19 and 25). Consequently, all subjects in the ACL are considered trusted.

We will add the new term candidate to simplify future discussion the following: a subject (s_a) is considered a *candidate* (c_a) for an object (o_b) only when subject has his public key (PK_a) in the keyring of the owner of the object.

Definition 26:

> An object o_b, is said to be *protected* if, and only if, access is restricted to those actions for named subjects specified by subject-permissions pairings named in the associated ACL.

Definition 27:

Access to an object o_b is granted based on a function $F_2(s_a, o_b, P_a, sk_a, KR_a)$.

$$F_2 = \begin{cases} 1, if\left(\left(F_1(s_a, o_b) = P_a \neq \varnothing\right) \wedge \left(sk_a \in KR_a\right)\right) \\ 0, otherwise \end{cases}$$

We have the potential to have a *protected* object (o_b), created by the *owner* (s_a), with an associated ACL consisting of *trusted subjects* who were *candidates* from the owners (s_a) keyring. Each entry in the *nx1* matrix that makes up an ACL will have a permission set $Pa \in P$ assigned to a trusted subject. We now need to explore what is meant to have objects secured.

In order for the owner to retain control, it must be infeasible for any subject outside of the ACL to obtain access to the data. It must also be infeasible for any subject in the ACL of an object to obtain elevated access without the express permission of the owner. Furthermore, it is undesirable for a user (other than the owner) to be able to determine the access rights of other users. This is undesirable because it would provide a potential attacker with information that may assist in a privilege escalation attack.

We must add elements to the framework that will allow us to enforce the owner retained control and prevent other subjects from bypassing such elements. Given an object (o_b) that has an ACL created by an owner (s_a), the data of the object and the information (metadata) describing the data must be secured from untrusted subjects. We will adopt a common notation for representing encryption/decryption of entities as follows:

Encryption: $\{\text{ENTITY}\}_{\text{ENCRYPTION\_KEY}}$
Decryption: $\{\text{ENTITY}\}_{\text{DECRYPTION\_KEY}}$

Definition 28:

An object o_b is said to be *secured* if and only if, the data of the object has been encrypted using a symmetric key SM_c that is in turn encrypted using each associated public key (PK_a), for each subject (s_a), within the ACL of the object.

Secured object $SO_{b,c} = \{o_b\}_{SM_c}$

Secured entry $SE_{c,a} = \{SM_c\}_{PK_a}$

and let SE be the set of secured entries for an object (ob) that has n entries in its ACL:

$SE = \{ SE_{c,1}, SE_{c,2}, \ldots , SE_{c,n} \}$

Information leakage occurs when data owned by one subject is permitted to be viewed by a second subject, and the second subject then forwards this information onto a third subject without the express permission of the owner. Definition 28 helps assure that information leakage does not occur since only those subjects in the ACL, specified by the owner to have access, will be able to gain access. Once access has been obtained, no information can be transferred outside of the permissions set out by the owner. It should be noted, however, that information leakage could still occur in such an environment by a subject who, for example, writes the information down on a piece of paper or photographs the screen. This is outside of the software policy and must be enforced by the guidelines in the physical security policy and supported by legal recourse.

Secured entries and secured objects can contain more than simply the data and the symmetric key. This model is only depicting what needs to be placed in the containers, not what could be. The secured object is essentially being placed in a locked safe, and then the password to the safe is being placed into sealed envelopes that can only be opened by subjects in the ACL of the object. This is achieved in two steps: retrieval of the symmetric key and retrieval of the object data.

Axiom 4:

For each entry in the ACL, the symmetric key (SM_c) for encrypting/decrypting the data of an object (o_b) can only be obtained for the secured entry of $SE_{c,a}$ through decryption using the secret key (SK_a) of a subject (s_a) in the ACL.

$SM_c = \{\{SM_c\}_{PK_a}\}_{SK_a}$

Axiom 5:

There is a unique symmetric key (SM_c) such that the original object (o_b) can only be obtained from the secured object through decryption using this key.

$o_b = \{\{o_b\}_{SM_c}\}_{SM_c}$

Definition 29: (Simple security rule)

> Access is granted if and only if there is an entry in the ACL allowing subject s_a access to object o_b with permission P_n.

For each subject in an ACL, access is only granted with its associated access attributes if the subject has possession of the matching secret key for the public key that was used to secure that particular ACL entry for the object.

Axiom 6:

> For every entry in an ACL for a *secured* object ($SO_{b,c}$) there exists a unique secret key (sk_a) that matches a public key (pk_a), by the properties of public-key cryptography (Section 1.8 and Chapter 8 [34]) This public key was used by the owner to encrypt the symmetric key in the ACL of the object (o_b), allowing for retrieval of the symmetric key.

Theorem 2:

> A secured object remains protected regardless the environment (owner-retained control).
>
> Proof:
>
> > By Axiom 5, the only way the original data can be obtained is through decryption using a unique symmetric key. By Axiom 4, the only way this key can be obtained is through decryption of a secured entry in the ACL using corresponding (unique) secret key of a trusted subject (from Theorem 1). Then by Definition 26, we know that the object remains protected at all times since the permission-subject pairings specified in the ACL are for only trusted subjects.

Corollary 1:

> An object may only be accessed by those recipients chosen by the owner.
>
> Proof:
>
> > By Theorem 2, a secured object always remains protected. By Definition 27, access is only granted to those subjects specified in the ACL. Then by Postulate 1, we know that the owner may add only trusted subjects to an ACL for a given object. Thus only the owner can specify which recipients can access the object.

We have defined what having a secure, protected object that is controlled by the owner who has specified which trusted subjects have been granted what permissions means. Final remarks for the basic model are needed; listed below are some thoughts to keep in mind when considering implementation of this model:

- ownership does not have to belong to the creator of the object
- ownership can be transferred, in contrast with originator controlled policies
- only the creator or an existing owner can specify a new subject as the owner of an object
- a new ACL entry for object o_j to be accessed by subject s_i can only be created, if and only if, the subject performing the action is the object owner or the object creator
- objects can come in many forms, therefore, the owner-retained control policy can apply at many level, such as the following:
  - per file
  - per data (this can be data that has been copied from one document into another)
  - per packet
- objects can contain other objects and can be nested, as such each object should have its own ACL.

3.2 Shared Ownership

Considering owner retained control in the basic sense and in the distributed sense are useful. However, in this expansion of the model we will consider the problem of shared ownership. The prospect of having multiple subjects share the ownership and responsibility of a single object presents new problems that the basic model does not consider.

For a basic understanding, consider the following: the creator of a document is considered to be the owner-creator (OC), and this person wishes to promote other contributors to the ownership level. In this situation, we may desire that only the OC be able to add and modify permissions for other owners; however, the sub-owners should be able to manage the permissions for other users. There are multiple ways this problem could be addressed, but we leave that discussion for future work.

As defined above, ownership is defined as a subject who has been granted *control* (*co*) as one of their permissions. It is not hard to imagine the situation arising where more than one subject has this attribute in their ACL entry.

Axiom 7:

Every object must have at least one owner.

$$\forall x : \exists y : \left(\mathrm{F_1}(\mathrm{s_y}, \mathrm{o_x}) = \mathrm{P_y} \wedge \mathrm{co} \in \mathrm{P_y}\right)$$

To simplify this determination we will introduce another function $F_3(s_a, o_b)$ that will determine if a subject is the owner of an object.

$$F_3 = \begin{cases} 1, if\left(\left(F_1(s_a, o_b) = P_a \neq \varnothing\right) \wedge \left(co \in P_a\right)\right) \\ 0, otherwise \end{cases}$$

Definition 30:

An object is said to have *shared ownership* when more than one subject in the ACL of the object contains the access control attribute (*co*). Given an object (o_b) and an ACL (ACL_b) containing entries for n subjects, we have the following:

$$\left(\sum_{i=0}^{n}\left(F_3(s_i, o_b)\right)\right) > 1, \text{ where } s_i \in ACL_b$$

Consider the case where we have an object (o_b) and two subjects (s_a, s_c)

$F_1(s_a, o_b) = P_a$ and $co \in P_a$

$F_1(s_c, o_b) = P_c$ and $co \in P_c$

In this example we have two subjects (s_a, s_c) who both have ownership of the object.

Not only should there be support for shared ownership, but also the ability to have transferal of ownership. The true owner of any object has the option to sell, donate, or share the ownership of the said object at his discretion. This is not the same as discretionary access control as the definition of owner is not the same. We will discuss these differences later.

Definition 31:

An object has had a *transfer of ownership* when the current owner of an object modifies an ACL entry for the object to contain the control attribute and also removes the control attribute from his own entry. Once committed the new subject is the owner of the object.

Consider the case where subject s_a transfers ownership to s_c:

$(F_1(s_a, o_b) = P_a$ and $co \in P_a)$ and $(F_1(s_c, o_b) = P_c$ and $co \notin P_c)$

-s_a adds s_c as an owner and commits the change

-s_a removes s_a as an owner and commits the change

$(F_1(s_a, o_b) = P_a$ and $co \notin P_a)$ and $(F_1(s_c, o_b) = P_c$ and $co \in P_c)$

Resulting in s_a transferring ownership to s_c.

Following from the above definition, subject s_c cannot change or modify the access control list of object *ob* until he has become owner. To enable this transfer of ownership, we must have the notion of time.

Axiom 8:

A subject s_c who is not an owner of an object o_b at time t can only add an entry to the access control list ACL_b at time $t+n$ if and only if, between t and $t+n$ the owner from time t has added s_c as an owner of o_b

We now have policy and enforcement models in which we can base an implementation for an environment that enforces owner-retained control for objects where ACL can be populated with trusted subjects. The environment should support concepts of shared ownership and transfer of ownership and should abide by the models specified above.

Chapter 4 Implementation

4.1 Possible Applications and Architectures

The following section explores the application domain and possible architectural designs that the ORCS prototype could coincide with. We first discuss some of the work sectors that could benefit from owner-retained access control along with different applications that could also benefit from such mechanisms. Before moving onto the proposed solution, we briefly discuss several architectural design approaches to dealing with owner-retained access control.

4.1.1 Applications

Owner-retained access control systems can be of use in many different environments and organizations. Entities who could have definite use for more diverse access control systems, implementing owner-retained access control, include the following: academic, government, military, intelligence, non-profit, law enforcement, corporate, legal, financial, medical, and personal. Within the physical and software environments that are used by these entities, many different systems can make use of owner-retained control mechanisms. These systems include, but are not limited to, the following: collaboration environments, operating systems, databases, version control systems, web-based systems, network systems and distributed environments.

Databases, as an example, could benefit from owner-retained control mechanisms. For instance, ORCS can be seen as a potential extension to Hippocratic and other protected database schemes. The concept of ORCS may be applied to outgoing datasets,

requested from the database, destined for trusted applications that enforce the ORCS policy.

Much of the security of documents in circulation today is dependent on the security systems built into the operating systems and associated file systems that the files reside on. These architectures are dependent upon the access control and encryption policies that are tied to the file systems and enforced by the operating system. An example of this is the enforcement of security through file permissions and optional encryption of the Microsoft NTFS system. Several of these approaches are discussed in section (2.14). It is, however, a problem when we export data from operating system environments. Once data leaves the protection of its filesystem, operating system, or application it is an open book to anyone who wishes to intercept it. We must be responsible in our attempts to protect information in this intermediate stage between data source and endpoint. Before we discuss the implementation, let us first discuss other potential architectural decisions that could be explored in the future.

The mechanisms outlined in the enforcement model could be implemented at different layers of a computing device. A particular mechanism could be implemented in any of the following ways:

- as a user-space (application layer) application
- as a user-space (application layer) library
- as a feature of the kernel
- as an extension to the network layers
- as a hardware mechanism
- as a server or service
- as a hybrid of any of the above

When designing a new piece of software there are many factors to take into consideration. In the case of ORCS, the prospect of having a security tool that enforces the restrictions, placed on a document by the owner, no matter where the data travels is a very attractive prospect. To create an application that allows this type of enforcement, you have two primary choices using today's technology. First, you could implement the application through a platform independent interpreted language. Second, you could create a web-based framework. But the second option requires a network connection in

order to use the system. In reality, data is often accessed without a network connection; consequently, we should strive to create an application framework that can run independently. This brings us back to the first option. Of the platform independent languages that are available, Java provides many advantages for security, such as possessing virtual machines for essentially all architectures in addition to a mature security API. Virtual machines (VM) also provide a security advantage because they provide an abstract layer on top of lower level mechanisms, where VM mechanisms can enforce restrictive rules to increase reliability and improve security. Examples of this are found in the Java VM, such as its ability to enforce security restrictions on Java applets running from within Web browsers and the ability to remove the possibility of buffer overruns by performing bounds checking on arrays.

Now that we have discussed the variety of entities and environments that can use owner-retained control systems, we will now discuss some architectural considerations before introducing the ORCS application prototype.

4.1.2 ORCS Architectural Considerations

The ORCS prototype addresses the owner-retained access control problem at the application layer (also known as user space). By choosing to architect the application at this layer, we are ensuring that the data protected by the application is secured through the lower layers of the computing base (see Figure 5). Linux, as an example, has an architectural layout that consists of three primary layers: user space, kernel space, and hardware. At the operating system layer (kernel space), we could revise the current implementations to allow for the containment of files inside of XML wrappers. These wrappers will allow for the specification of more restrictive access control measures implemented in the protected mode of the kernel.

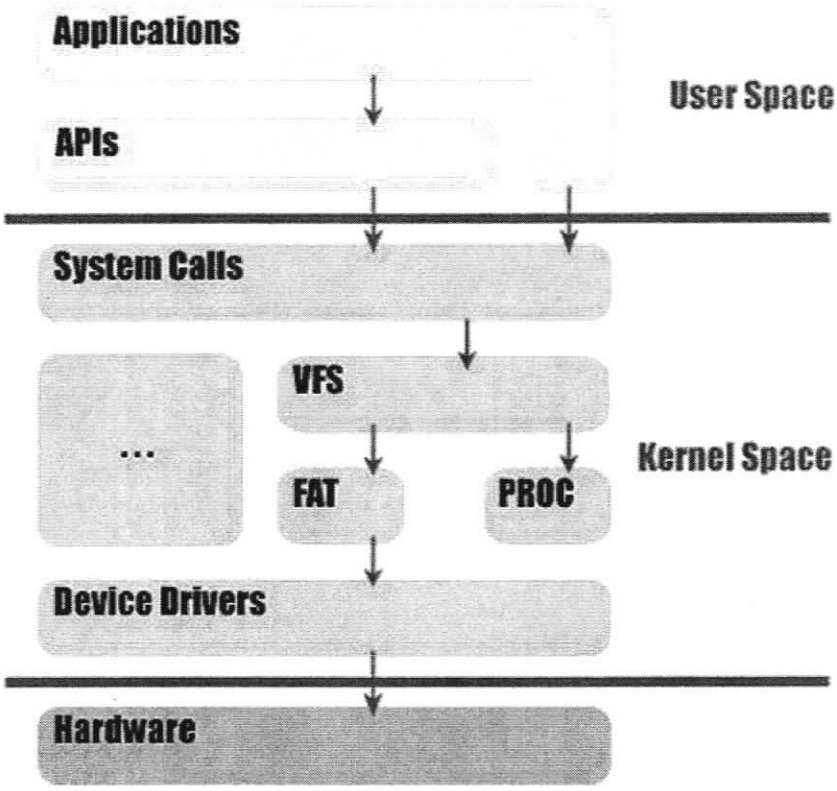


Figure 5: Linux Handling of I/O Requests

With such an approach, applications in user space can be customized to allow for enforcement of owner-retained access controls and other DRM at a finer grain of control. At the network layer, protocols could be established to allow for remote collaboration and document transfers. It should be noted that no changes to the underlying network protocols should be necessary. Such an approach would remove the feature of portability. Thus the ORCS prototype remains, for the time being, a proof of concept that resides entirely in user space.

4.2 Proposed Solution

Now that we have discussed the application and architectural considerations, we will discuss our approach to solving the problem of owner-retained access control, describe and validate our choice of technologies, and explain the format of the protected information. Following this section we will discuss the implementation.

4.2.1 Approach

Our approach involves taking the control of security away from the platform-dependent mentality of storing security policies in systems that are tied to the file and operating systems attached to a particular platform. Rather, we place the security information in the file itself and enforce the policy through the use of our application software. Once the pertinent information needed for owner-retained control is embedded within the file itself, we are able to retain and enforce the same restrictions/control of the document whether the document remains on our local machine or elsewhere.

Integral to the success of a mechanism for enforcing owner-retained access control for a document management system is an effective and efficient document format and an application base that has intimate understanding of this format. In truth, multiple applications or software frameworks could be created to enforce the format of the document, such as the different approaches listed in the previous section. There would need to be a protocol to audit and certify each application base in such an environment to ensure compliance with policies.

The implementation of the ORCS prototype creates an application that is meant to enforce portions of the policy and enforcement model we presented earlier. In this application, subjects are simply people or system users, objects are represented as the character data we wish to protect, and a protected memory area is represented as the system clipboard.

The access control attributes (Definition 4) must be enforced by the application when a user attempts to access an object. For example, when a user attempts to use the cut, copy and paste operation the system should restrict access to these features unless the particular subject has been granted the appropriate access attributes.

Asymmetric key-pairs (Definition 18) can be generated using the ORCS prototype; this is done to simplify testing. In a production system, however, the key-pairs and certificates would need to be produced by the security administrator (such as the certificate authority). In addition, all certificates are self-signed to ease testing. Each ORCS environment belongs to a single owner, and as such, possesses its own password-protected keyring (Definition 19). Symmetric keys are generated in a black box and are used to protect the inner document data.

Each object, or character data, that is to be protected is wrapped inside of a wrapper that is protected using an ACL (Definition 23) that restricts access to only the trusted subjects (Definition 25). Subjects are only added to the keyring of an owner's ORCS environment via a digitally signed certificate from a trusted certificate authority; however, the prototype assumes that a self-signed certificate is actually from a trusted CA.

Enforcement of the ACL is the job of the ORCS prototype, and thus, the object should remain protected (Definition 26). The object is secured (Definition 28) by implementing the algorithm using the advanced encryption standard (AES) as the symmetric key algorithm, and RSA key-pairs for the asymmetric algorithm.

When an object is saved, the ORCS prototype encrypts the inner data using the symmetric key and proceeds to encrypt the AES key with each RSA public key specified by the ACL (as per Definition 28). Consequently, the resulting document is protected (Definition 26) until a subject who is specified in the ACL attempts to access the document and provides his secret key (as per Corollary 1), resulting in a secured object that can only be accessed by those subjects specified by the owner of the object in the associated ACL.

4.2.2 Choice of Technologies

Our primary goal in building such a system is to combine a specific document distribution format with a suitable security protocol. For the implementation of the document format we researched some of the available technologies and found that technologies based on and tied to eXtensible Markup Language (XML) were appropriate. XML provides an efficient means of marking up a document with the future intent of parsing and interpreting the contents. As we will discuss later, the document will contain multiple sections with varying levels of information and encryption. The other major advantages of the XML technologies are the versatilities found in the grammar definitions and the relative ease associated with performing modifications to the grammar. Furthermore, it appears that XML represents a standard that is here to stay. The grammar for ORCS will ideally cover all necessary permissions and allows for

assignment of access rights on a per-user, and in the future, group basis. Using XML schema definitions (XSDs) we are able to draw a clear line between access control policy and access control mechanisms. XSD documents allow us to specify the format to which a document must conform; this is where we define the policies. Thus, the policy of a system can be defined in XSD and the protected data can then be stored in XML documents that conform to the given policy.

Security assertions markup language (SAML) was chosen for the basis of our access control mechanism since it represents a potential industry standard means for representing complex security rules in XML (other grammars also exist, such as XACML). Document object model (DOM) was selected for the parsing and manipulation of the document as a whole. When a document is opened, we never want the data stored on disk in any form other than encrypted. DOM assists with this by allowing the document to be temporarily placed in a structured format in memory during runtime rather than storing to disk, a process not accomplished by SAX. DOM, therefore, gives us the ability to parse, manipulate and restore the data in one seamless cycle.

With the popularity and widespread use of the platform independent Java programming language, we have chosen to implement the application prototype for ORCS in Java. This choice has several strategic advantages. The primary advantage, at this phase of development, is platform independence. As we are attempting to create a document control system that will eventually allow collaborators to use and enforce document control independent of their location and operating environment, Java provides the portability we are seeking. With Java virtual machines (JVMs) available for virtually every operating system, we have an efficient medium for widespread distribution; however, in the future other technologies may also provide this same advantage. We will now briefly discuss protection protocols and document format before we validate our selection of technologies for the application prototype.

4.2.3 Protection Protocols

Before discussing how the data is stored, we first need to discuss how the data is used. For owner-retained access control and DRM there is more than one possible application, as mentioned above. In some instances it is desirable to maintain control over the data while it is in live use. Other instances demand that the data be in static form and simply distributed. We will now briefly discuss the two possible configurations and validate our current approach.

In live environments, such as real-time collaboration environments, it is possible that the owner may desire to add or remove access control attributes for a particular subject or group. For example, if a particular company had a contractor collaborating on a project with other employees, but the contractor later left the company, the owner would want to revoke the permissions of the contractor to access resources immediately. This would require that the owner-retained access control implementation be designed using a distributed network architecture, consisting of both client and server components. With this architecture, more advanced security protocols would be needed, such as user registration, information protection update, and document distribution protocols.

In static environments, such as the case of digital music and portable music devices, the network is not always present; consequently, advanced network protocols cannot be used to enforce security policies. This architecture would require that documents be stored in a protected format, containing their security policies, which are then applied when access is attempted. Thus resulting in greatly simplified protocols. If an owner wished to revoke privileges for an object of this type, he would have to place that restriction on the data before the document was delivered to the subject. In DRM technologies this is often done using access-control attributes which decrement after each use, and the owner has paid a license fee that allotted the subject a certain number of accesses.

Both of these approaches have advantages and disadvantages, and they both have well defined markets. Even with live environments, we have to keep in mind the original intent of ORCON: to allow the originator to control the redistribution and access to data. In the case of documents, images, videos, and music it could be argued that revoking a

privilege during use would be desirable; however, once accessed the subject possesses the knowledge, and the knowledge cannot be removed once obtained.

Thus for the ORCS prototype we discuss here, the static approach for owner-retained access control has been chosen as we wish to focus on restricting access to data to those subjects the owner chooses. We leave the live environment approach and the associated security protocols as future work.

4.2.4 Format of Protected Information

Figure 6 depicts our secure document format. The document header maintains all of the mandatory information that is needed to enforce the constraints of the system. Whether the file is on the owner or any other machine, the following fields need to be maintained: owner ID, owner name, owner address, list of users (each containing a user ID, user name and user address), an access control list (ACL), three time stamps (creation date, modification date, accessed date), a synchronization bit (for synchronizing states with the owner when both are online [for future use]), and finally, the field for the secret symmetric key for encrypting/decrypting the document body. Most of these fields are further protected through the use of either symmetric or asymmetric encryption. The headers associated with the ACL are implemented using the security assertion markup language (SAML) XML grammar from OASIS combined with our custom markup.

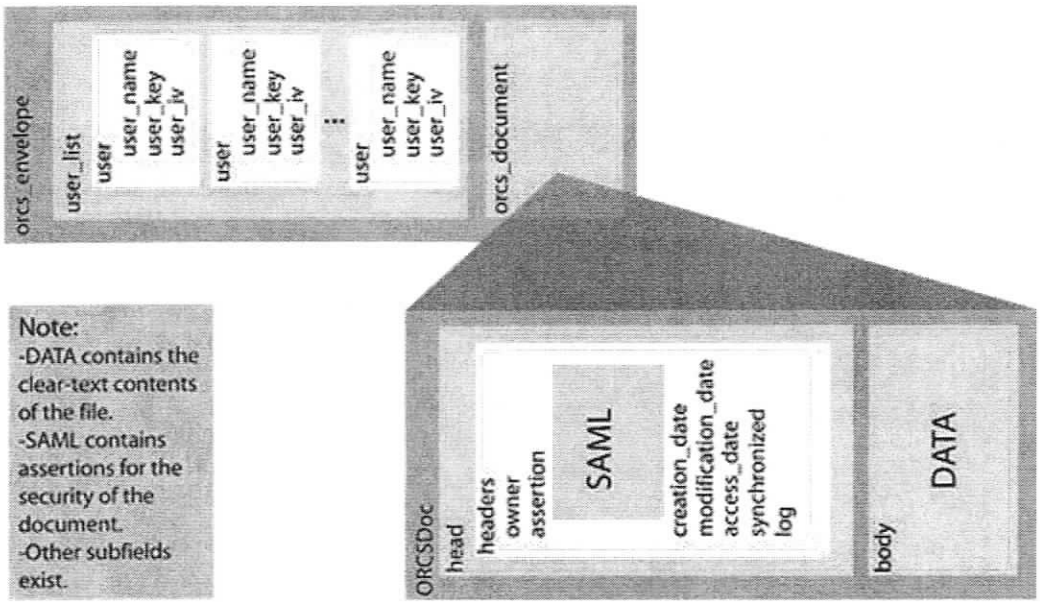


Figure 6: Format of Protected Information

The data that we are trying to retain control of is contained within the body of the document. Using the symmetric key from the header this data may be encrypted or deciphered by an authenticated party at any time. Recall that a user must have authenticated privileges in order to obtain the symmetric key. We will further discuss the encryption algorithms later in this thesis.

SAML, a specification managed by OASIS, defines an XML vocabulary, protocol, transport bindings and usage profile for exchanging assertions about authentication, attributes and authorization decision. Our focus with SAML is on authorization of users through a well defined, suitable grammar based on the rules specified in the SAML specification. The ACL supports standard access specifications for granting and revoking privileged operations. These operations include but are not limited to the following: read (open and browse), write (modify, save and save-as), no access (access revoked), and update user permissions (change user access rights). Considerations are also given to the more advanced situation in which a document has multiple owners. In summary, the grammar ideally covers all necessary permissions and allows for assignment of access rights on a user, and in the future, group basis.

4.3 Implementation

During an example situation, a user/principal can only access the body of the document via a trusted software application (ORCS) running on his machine. It is essential that the trusted application be installed and executed on a secure/trusted computing base [31]. If this is not the case, there are known ways to attack a system that is not secured. We will discuss some of these attacks later in the thesis.

4.3.1 Software Interface

In most large organizations the hardware resources of the digital environment are commonly configured in a network environment. From the standpoint of the system we wish to explain the interface in a progressive manner. First, we will consider a standalone user working with the system (the document owner). Then we will consider sending the document to a second principal, residing at a different host, who plans to view the document using a trusted application. The second case can easily be extended to include multiple subjects at multiple locations.

4.3.1.1 Document for Owner

When the owner first opens the ORCS application, they are welcomed by a splash-screen before continuing on to the user interface. If this is the first time using the tool, the user will then be placed in a wizard that will step him through the process of creating a keystore and a set of RSA keys for his own personal use (see Figure 7 and Figure 8).

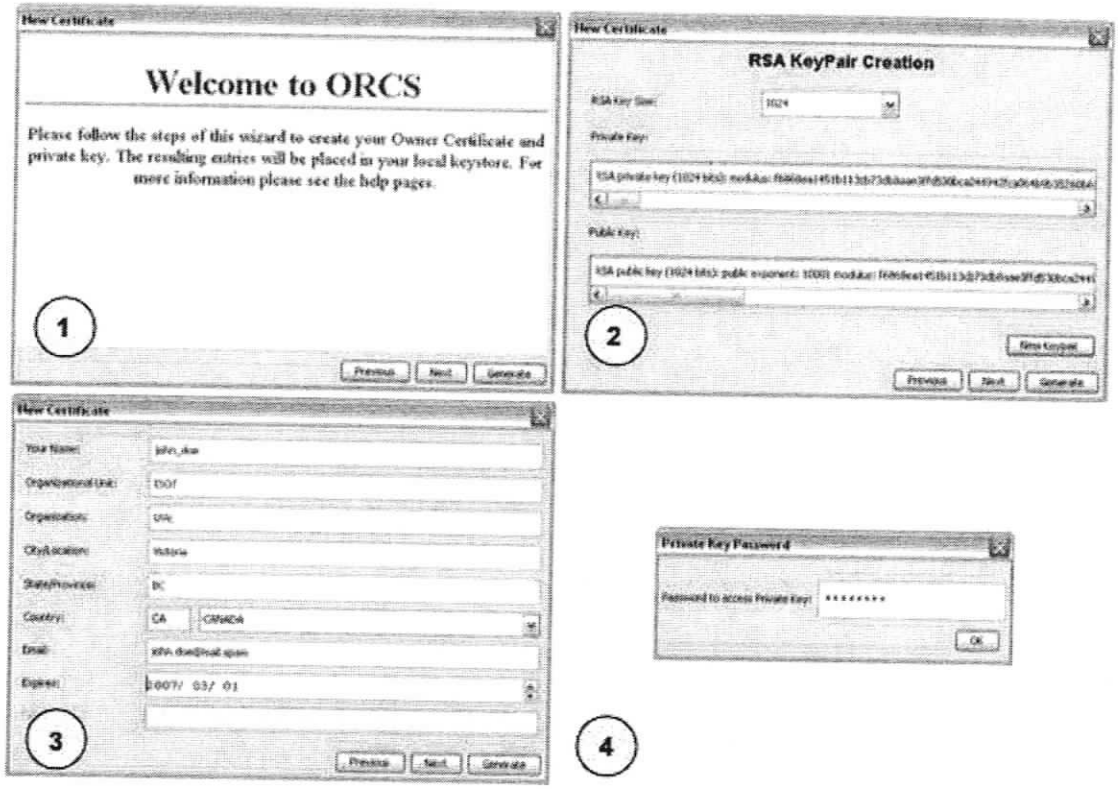


Figure 7: Creation of RSA key-pair and ORCS keystore.
The subject is first welcomed to the environment (1), then he is prompted to generate an RSA key-pair, followed by a form to enter his personal information for the digital certificate, and finally he is prompted for a password to protect his keystore. For the prototype we use self-signed certificates.

Users, also known as subjects, entering the system are required to authenticate themselves when they attempt to open any file. They can fall into one of three categories: authenticated as owners, authenticated as valid users, or failed user authentications. In all cases they must provide user information when opening a file. At this time, we have implemented only basic authentication based on a secret (password). Future authentication could be implemented with technology such as biometric devices. SAML can also provide XML features for a clean definition regarding user access rights and authentication.

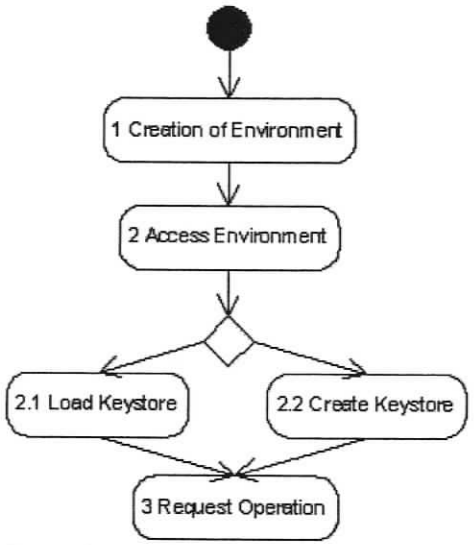


Figure 8: Loading the ORCS environment.
When the ORCS environment is started, the environment first checks for the existence of a keystore. If the keystore does exist, it is loaded and users can then request operations. If the keystore does not exist, the user is placed in a wizard that will assist him in creating one.

For our first scenario the owner, john_doe, will continue working on a document that he started previously. From the file menu the user selects to *open a file*.

The *road.orc* file is provided as a starting file, containing all the headers and body in ciphertext. Once the user selects this file (Figure 9), he will need to be authenticated prior to being able to potentially view, modify or save the document.

The user is immediately prompted for his authentication information when he requests to open the specified resource. Upon successful authentication, the user is then presented with the document after it has been decrypted. With the document now opened, the owner may proceed in manipulating the content as he see fit.

In addition to the traditional properties dialog, the owner may also review the user profiles and ACL that have been specified in the document header. There is an intentional resemblance to the UNIX/Linux naming (Figure 10) convention for the user IDs and a noticeable resemblance to the file permission scheme that we have seen for years in UNIX and UNIX like environments. We have chosen this route for the simple reason that there will be a lesser learning curve for users if they can use familiar configurations. We are simply changing the working environment.

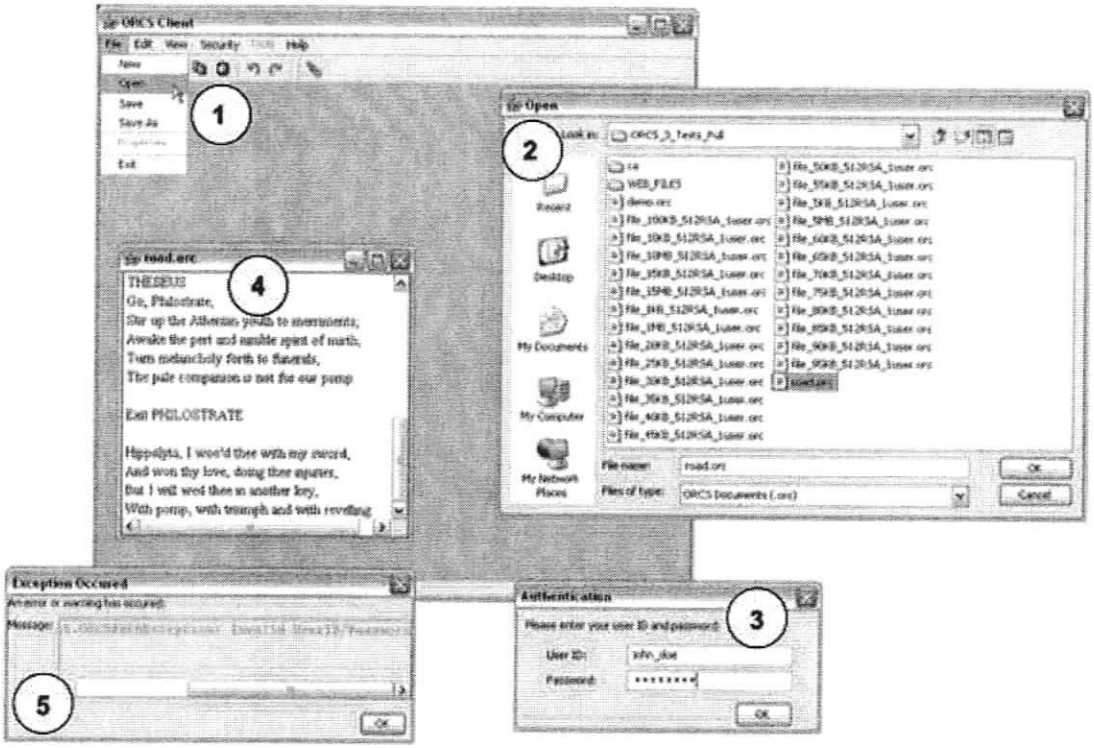


Figure 9: Owner opening ORCS document.
The owner, john_doe, elects to open a file (1), is immediately prompted to select his file (2), then is prompted to authenticate against the file (3), resulting in either success (4) or failure (5).

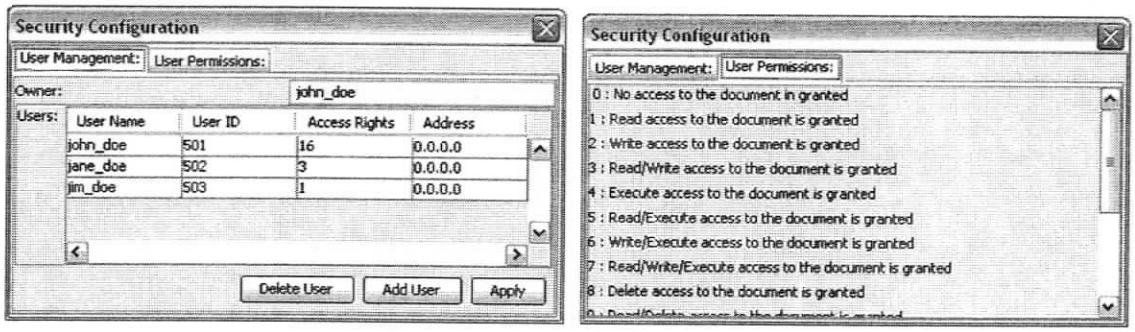


Figure 10: User profiles and ACL of ORCS document.
The owner has currently specified that two additional subjects may share the document. Jane has been granted both read and write permissions, while Jim has only read access.

After the owner has modified the document to his liking, he can then choose to save the document using the "Save" or "Save As" options in the File menu, resulting in a protected document (see Appendix A). Since he is the owner, he should be allowed to save the document as any filename he chooses. This should not be the case later on for other users who are modifying a document that is owned by someone else. When the

owner saves the file it is parsed back into a file to be stored encrypted on his storage device. During a save operation, an entry is placed in a read-only log contained in the document to help track changes to the document and ensure its integrity.

4.3.1.2 Document for Sharing

We are now ready to consider the scenario of handing the file off to another user. The file can be transferred to another party by any means seen fit. This can include, but not be limited to, network file transfer, email attachment, CD-ROM, floppy disk, or a shared network drive.

The trusted application software, ORCS, is responsible for decryption of the protected information and the enforcement of the security restrictions specified in the header of the document. Any form of access to the document is made through the trusted application, which is responsible for enforcing the owner's security policy at the subject machine. If an individual were to access the document through any other application software, such as a text editor, he would see the content of the XML document with the critical information viewable only as base64 encoded cipher-strings that would mean very little until it is decrypted. In order to restrict the actions of the user, the trusted application intercepts all the communications between the user and the underlying operating system. Actions that are to be captured if the subject only has read access include clipboard actions (cut, copy, paste), and document modification commands (see Figure 11). The restrictions are essentially implemented through the use of an event listener that captures user requests, allowing the requested actions only if the user permissions are consistent with the action requested. Event features can either be enabled/disabled when a subject is authenticated, or the calls can be redirected to a verification module that checks the access rights and processes requests accordingly. It is important to stress that the document, when stored on a physical medium, always remains encrypted. This approach is somewhat similar to the restricted set of functions discussed by Abrams et al. for the implementation of the ORGCON policy [1].

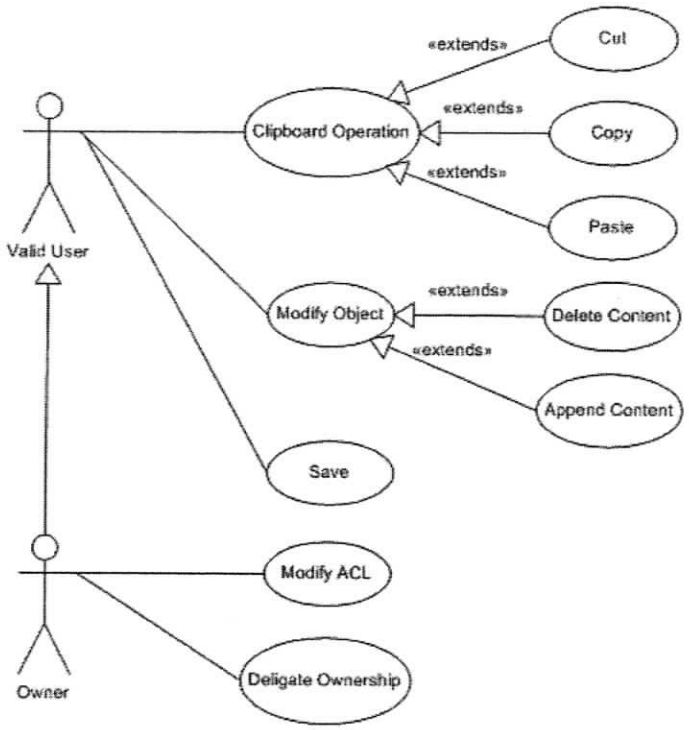


Figure 11 Valid user access rights.

Consider the case where the owner, john_doe, from the previous section sends his document (road.orc) by email to another subject, jim_doe. The new subject then attempts to open the document using the ORCS prototype. Once the subject is authenticated, the environment enforces the rules of the owner on the document (see Figure 12).

Through the enforcement mechanisms of the ORCS environment, jim_doe can only view the contents of the protected character data. All other permission sets were denied by the owner john_doe.

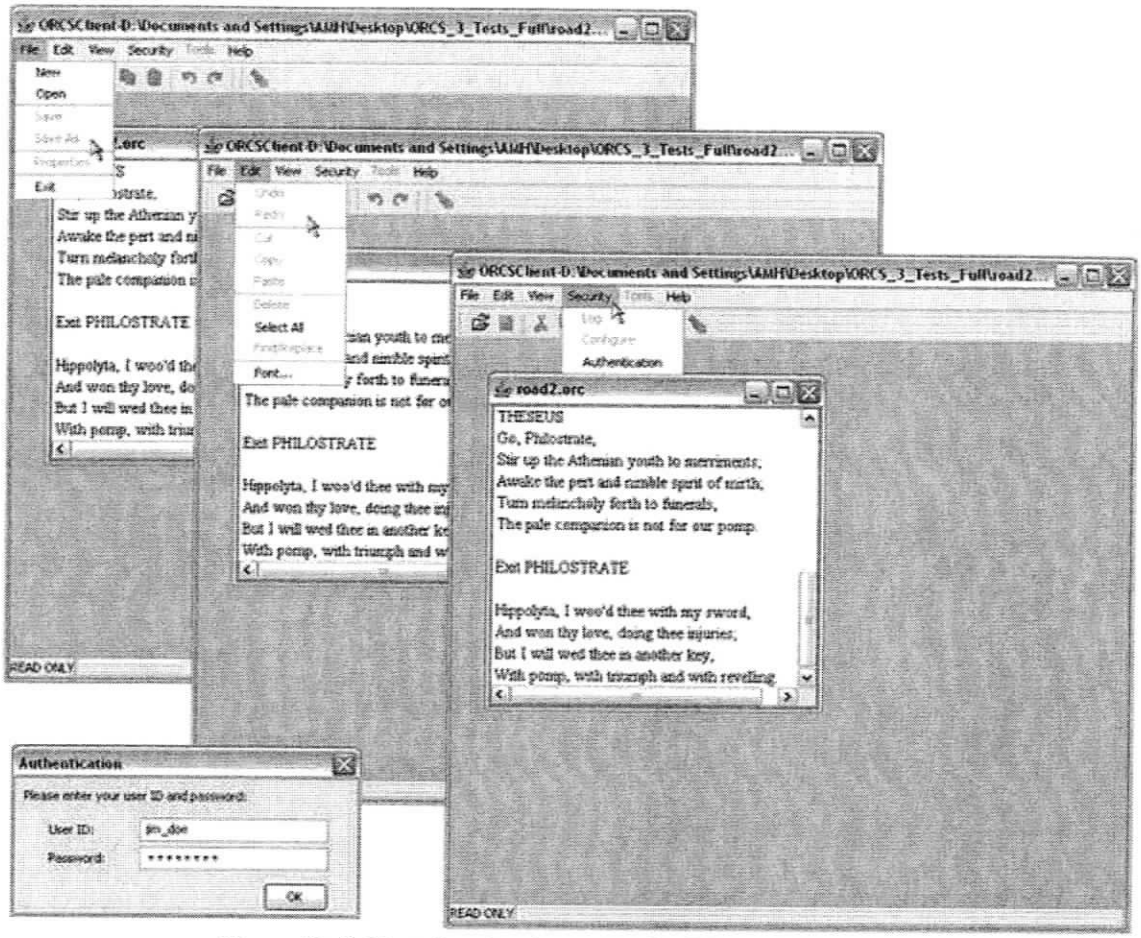


Figure 12: Subject, jim_doe, is restricted to READ_ONLY access.
The screen captions above depict the disabled features of the environment when the subject is granted READ_ONLY access to the document. Features such as saving, cutting, copying, pasting, and viewing of the ACL, log, and properties windows are all disabled.

4.3.2 Encryption and Key Distribution in ORCS

Based on the proposed theoretical model and following the guidelines set out above, we have chosen to implement the encryption in two levels. First, we implement a symmetric encryption algorithm to encode the body of the document. Using symmetric encryption we are able to enforce a high level of encryption on the most integral information contained in the document. The key for the symmetric encryption is stored in a field within the header of the document. This is similar to how SSL accomplishes security. First, public/private key encryption is used to establish a secure connection, and then a symmetric key encrypts the data traveling between the stakeholders. We have chosen the advanced encryption standard (AES) algorithm for this portion of the project;

specifically, the 256-bit long version of the Rijndael algorithm, the successful candidate of the AES competition. Second, the need to secure the transmission and storage of the document leads us to encryption of the message headers. For the actual message passing we also need to encrypt several of the headers in the document. These include the ACL, symmetric key, and stakeholder information. For this portion we take advantage of asymmetric key encryption, potentially using public key encryption techniques. In order to better understand the need for encrypting the headers, we will consider the diagrams in Figure 13.

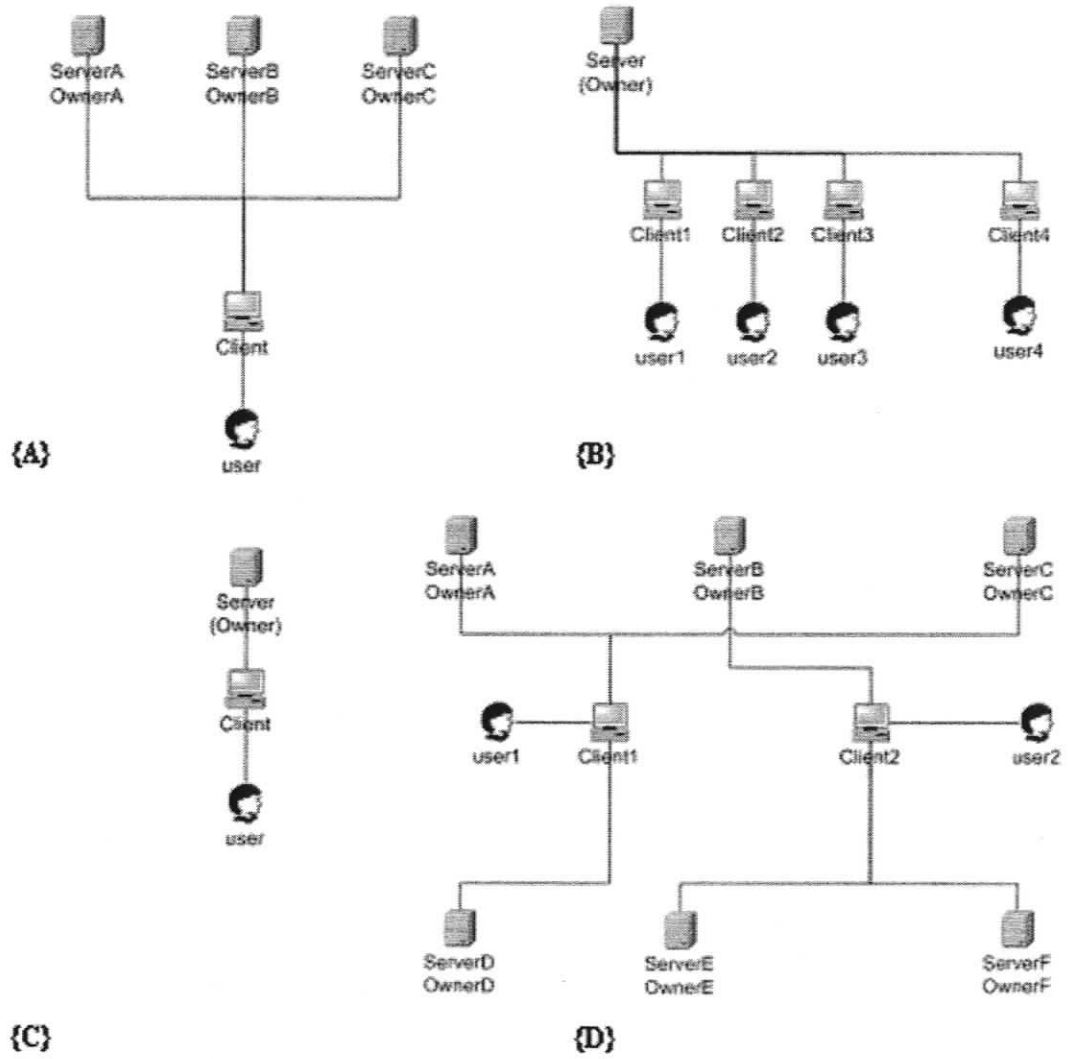


Figure 13: Possible Network-based Collaboration Configurations

In the most basic example, we have the traditional application/server (one-to-one, Figure 13 *C*) relationship where the owner shares a single file and the user accesses the file through the application software (the application and the server could be the same machine). In a slightly more complicated example, we have the single application manipulating and contributing to multiple documents originating from multiple owners at different servers (many-to-one, Figure 13 *A*). The converse of this scenario is found in the diagram depicting the single server sending out a single document to multiple contributors (one-to-many, Figure 13 *B*). Finally, we need to consider the case were we have multiple users operating on their respective applications on multiple documents originating from various owners at their respective servers (many-to-many, Figure 13 *D*).

In the four different collaborative configurations (Figure 13), we need to consider the best encryption technique for the task. We chose the above scheme because in the one-to-one example, the use of broadcast encryption would require more overhead in both keystore size and the encryption/decryption of data between collaborators than is warranted. Colin Boyd [14] gives three example situations that would suit a multiple key cipher configuration. These examples fit well with the one-to-many network configuration, as it allows for distribution of data to many users while keeping keystore sizes in check. It would seem, however, that this system would not scale well to the many-to-many environment. For example, it would require that each center (owner) generate his own set of unique keys, so that his key distributions do not overlap with those of others. The resulting solution may involve having a keystore for each owner involved in the collaboration scheme installed on each application. Therefore, we chose to apply a public key encryption scheme based on RSA for the document headers. ORCS provides support for RSA key sizes ranging from 512-bit to 4096-bit (see Figure 14).

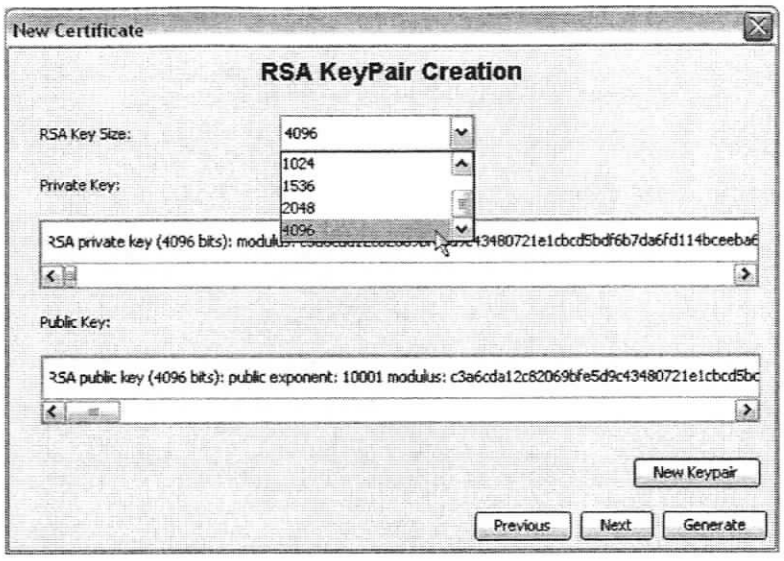


Figure 14: Supported RSA key sizes ranging from 512-bit to 4096-bit.

To add a new subject to an ACL, the subject must have his public key in the owner's keyring. As the model specified, the only way that a public key can be added to the keyring is via a digitally signed certificate from a trusted CA.

Figure 15 shows how the owner of a document can manage his keyring by adding or removing subjects to/from his group of trusted subjects. Once a subject has his public key in the owner's keyring, he is a candidate for inclusion in an ACL.

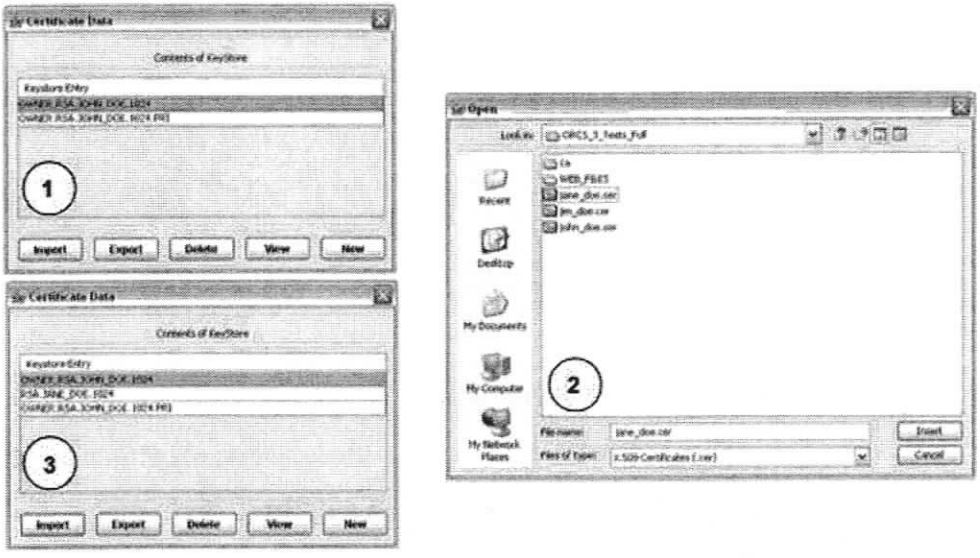


Figure 15: Importing digitally signed certificates into the owner's keyring.
An owner opens his keystore/keyring (1), then selects to add a new subject to the keyring by selecting a trusted digitally signed certificate (2), resulting in a new subject who may be added to ACL's for documents that he owns.

4.4 Summary

The ORCS prototype is a platform independent security enforcement application that has a detailed understanding of the policy model, enforcement model, and document format. Through the creation of this prototype in the application layer, we are able to build on the powerful platform independent features of Java. There are, however, several security risks in placing the security mechanisms at this level. For example, if an attacker were able to supplant the JVM or CORE API's with a modified version, information leakage could result. The advantage of having the implementation at this level, however, is that it remains independent of architectural specific mechanisms. In addition, the data is protected at an earlier point in time.

The resulting application has many features similar to those found in existing DRM; however, ORCS also provides free use.

Chapter 5 Evaluation Results

In this chapter we will evaluate the performance and security features of the ORCS prototype.

5.1 Performance Analysis

The two most expensive operations in the ORCS framework are that of parsing and encrypting an object. Both of these activities are involved in the save operation of the ORCS tool. To sample the expense of these operations, we have tested the save operation against multiple input objects, using multiple runs on two different operating environments.

In order for the ORCS model to be a viable approach to dealing with the concept of owner-retained access control, we need to show that the necessary components can be implemented in such a way that performance concerns are satisfied. Since the encryption and parsing phases of the model are expected to be the most time consuming phases, we aim to show that changes to key parameters of these phases have acceptable effects on performance. The performance targets include the following:

- Increasing asymmetric key sizes has minimal effect on CPU time.
- Increasing the length of ACL for an object has minimal effect on CPU time.
  - Small changes in ACL should result in negligible CPU time changes.
  - Large changes in ACL should result in a linear change in CPU time relative to ACL length.
- Increasing the data size has acceptable effect on time.
- Changing the computational environment has reasonable effect on time.

The ORCS prototype is an attempt to show that not only can owner-retained access controls be implemented to secure data; they should also be able to do so in a reasonable amount of time.

5.1.1 ORCS Environment

The ORCS prototype was designed to portray the ability of the ORCS framework to span multiple heterogeneous environments. To address such requirements, the execution environment for the ORCS prototype was implemented using a Java Runtime environment. Java applications are portable to any hardware platform where a Java Virtual Machine has been created for an operating system running on top of the operating system's architecture. While Java does provide these desired features, Java also has the following negative side effects: slower runtime execution due to bytecode interpretation, larger memory consumption due to garbage collection, and default heap behavior of the JVM. For the performance testing of this prototype, we used two different configured environments as described in Table 1.

Environment 1:	Laptop
Processor	Pentium 4, 2GHz
Memory	768 MB of RAM
Operating System	Windows XP Professional
Java Virtual Machine	Version 1.5.0_06 With the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 update
JVM Parameters	%JAVA_HOME%\bin\java.exe –cp .;./ORCS_2.jar;./ORCSDocs.jar; ./jhall.jar;./xml-apis.jar;./xmlParserAPIs.jar;./xercesImpl.jar; ./iaik_jce_full.jar;./jce.jar; -Xms350m -Xmx750m ca.uvic.parrot.gui.application.ORCSApplication

Environment 2	Server
Processor	Dual-Processor Server, 2.8 GHz Xeon (2x)
Memory	2 GB of RAM
Operating System	Windows 2003 Server (Enterprise Edition)
Java Virtual Machine	Version 1.5.0_06 With the Java(TM) Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 5.0 update
JVM Parameters	%JAVA_HOME%\bin\java.exe –cp .;./ORCS_2.jar;./ORCSDocs.jar; ./jhall.jar;./xml-apis.jar;./xmlParserAPIs.jar;./xercesImpl.jar; ./iaik_jce_full.jar;./jce.jar; -Xms350m -Xmx1446m ca.uvic.parrot.gui.application.ORCSApplication

Table 1: Environmental parameters for test machines.

Environment 1 represents the runtime environment for a laptop computer, typical of the operational hardware for an average subject; Environment 2 represents the runtime environment for a server, typical of the operational hardware for an average server that could be used in larger software architectures.

For these performance tests the Sun Java VM was used to run the prototype; however, other vendors also provide Java VM's that can provide improved performance for

applications. Vendors who provide Java Virtual Machines include Sun, IBM, HP, and BEA.

The environmental settings for running the ORCS prototype include runtime parameters for the Java Virtual Machine.

```
%JAVA_HOME%\bin\java.exe –cp .;./ORCS_2.jar;./ORCSDocs.jar;./jhall.jar;
./xml-apis.jar;./xmlParserAPIs.jar;./xercesImpl.jar;./iaik_jce_full.jar;
./jce.jar; -Xms350m -Xmx1446m ca.uvic.parrot.gui.application.ORCSApplication
```

The *–cp* option provides a unique classpath for the Java program instance, telling the JVM where to locate necessary libraries that are used by the application. To allow the application to obtain more memory on the system the following options are used:

-Xms<size> *Set the initial size of the Java heap*
-Xmx<size> *Set the maximum size of the Java heap*

The defaults for these settings on a server class machine (2 CPU's and at least 2GB of memory) are the following:

-Xms<1/64th of physical memory>
-Xmx<1/4th of physical memory, up to 1GB max>

We use these values to improve parsing times for large data files. Additional information on performance tuning for the JVM may be found at *http://java.sun.com/performance/reference/whitepapers/tuning.html*.

While these performance results were obtained using Microsoft operating systems, it is important to note that the same tests could have been done using any operating system that had a deployed JVM.

5.1.2 Document Size

The data used in the performance tests were created using initial character data of fixed sizes. Table 2 shows the basic input data files and the effect of the size of character data on the number of pages, lines, words and characters in the ORCS document. It is important for our testing that we be able to replicate data sets of increasing sizes to show how the encryption and parsing operations are affected by changes in parameters (key sizes, ACL lengths, data size/length, and environment). The entire ORCS prototype

environment was reset before the beginning of each new document test, helping ensure that the test environment had not become polluted.

Date Size [KB]	# Pages	# Lines	# Words	# Chars
1	1	34	165	958
5	4	162	871	4798
10	7	331	1772	9578
15	11	533	2678	14296
20	14	685	3565	19112
25	18	840	4493	23922
30	21	1019	5422	28684
35	25	1193	6331	33456
40	29	1386	7250	38190
45	33	1575	8111	42932
50	36	1757	9031	47688
55	40	1926	9953	52470
60	44	2118	10870	57206
65	48	2323	11733	61918
70	52	2505	12621	66673
75	55	2685	13554	71434
80	59	2857	14441	76210
85	62	3023	15322	80998
90	67	3249	16216	85666
95	71	3456	17103	90371
100	75	3634	18044	95135

Table 2: Basic data sets used in the baseline test cases.

There is roughly $3\frac{3}{4}$ pages of data for every five KB size increase. Numbers based on a character count including spaces, and a font set to Times New Roman, Size 12.

Not only do we want to test against relatively small data sizes, it is also desirable to test against larger data sizes to show how the framework can deal with different types of data. Table 3 depicts the same information as the previous table, but for larger data sizes. Note that the raw character data creates large documents with relatively low data sizes.

The prototype simply works against raw character data; however, these performance tests serve as a means to validate the feasibility of using this framework for different types of data. In collaboration environments and other targets for this technology, much of the data is transferred over network sockets; consequently, we can expect relatively small bursts of data over time. It is also possible that users may wish to share protected images and other documents during an interactive session; therefore, larger data sizes are required to test how the prototype can scale to larger data objects.

Data Size (KB)	# Pages	# Lines	# Words	# Chars
1024	760	37198	184962	974201
5120	3796	185986	924810	4871009
10240	7861	385149	1834736	9716352
15360	11657	571134	2759545	14587361
20480	15452	757119	3684355	19458371
25600	19248	943104	4609165	24329381
30720	23044	1129089	5533975	29200391
35840	26840	1315074	6458785	34071401
40960	30635	1501059	7383595	38942411
46080	34431	1687045	8308405	43813423

Table 3: Larger data sets that are used in determining performance.
The following test cases use these larger data sets to clearly show the changes in performance as data sizes increase. Numbers based on a character count including spaces, and a font set to Times New Roman, Size 12.

While average file sizes are virtually impossible to calculate precisely, several large scale studies have been conducted that can afford us an estimate on the size of data for particular file types. One such study, "How much information 2003" (http://www.sims.berkeley.edu:8000/research/projects/how-much-info-2003/), was conducted by researchers at the University of California at Berkeley, produced usable results. Table 4 provides a summary of some of the relevant data types that could be used with the ORCS model (source: How much information 2003, Table 8.11).

File Type Extension	Size (bytes) Min.	~Max.	Average
txt	1	161,000,000	51,415.17
doc	28	11,816,199	125,125.30
htm	4	18,030,044	20,782.51
pdf	162	134,000,000	5,411,649.00
jpg	75	29,016,816	44,829.72
gif	34	1,598,766	15,275.78
mp3	1	558,000,000	4,120,597.00
exe	1	816,000,000	7,293,783.00
mpeg	162	875,000,000	20,693,836.00
mov	162	382,000,000	8,790,495.00

Table 4: Sample file sizes for common file types.
The focus of the ORCS prototype is on character based data. But based on average sizes, many other data formats would be candidates for our model. For example, the average size of a Microsoft Word document (doc) is ~125KB for character data while the maximum was ~11MB; therefore, our tests should cover this range. The sizes are for the document size, which in the case of Word are larger than the raw character data that is contained within. Note: maximum sizes are approximate since original data was in scientific notation; for example, 558,000,000 was given as 5.58E+08 (source of table: How much information 2003).

Performance tests in this section work on data files containing up to 45MB of character data. The results obtained give an estimate on how long it takes to convert the raw data and relevant header information into the protected ORCS format. The average size of character based files (txt, doc, htm, pdf) found in Table 4 demonstrates that we can expect the average person to be working on text file sizes ranging, on average, from 20KB (htm) to 5MB (pdf). If the ORCS model were to be modified to place owner-retained access controls around any nested data type, we would expect the user to be working with files ranging from 15KB to 20MB on average.

5.1.3 Effects of Key Length

One of the two largest time consumers in a save operation is the encryption phase. Encryption for the prototype is broken down into a two-phase algorithm. The first phase encrypts the collected data using a symmetric encryption algorithm, while the second phase encrypts the output of the first phase using an asymmetric, or public key, encryption algorithm.

Results were obtained by creating an ORCS document containing 10MB of character data, a single user ACL (with control access rights), 256-bit AES symmetric key, and an RSA asymmetric key length ranging from 512-bits to 4096-bits. The performance experiments were conducted by invoking multiple save runs against ORCS documents with all parameters fixed, except for the one particular parameter being validated in the particular test (i.e., RSA key length). By fixing all parameters, within reason, except for the asymmetric key length, we collected timing data to validate our claim that changes in the RSA key size should have relatively low overhead in regards to increased CPU time. Since there are two phases to encryption in the model, we need to consider both the symmetric and asymmetric passes.

Results for this experiment are presented in Table 5. We will now analyze the results that were obtained from the multiple runs against our test data in an attempt to validate our first target.

First, we should expect that the AES encryption times during the save operation should not change during the tests, as the data being encrypted by AES is not changing

size (Figure 16). We can validate this expectation by comparing the median AES encryption times for 512-bit RSA (4430.0 ms) and 4096-bit RSA (4429.0 ms). There is no significant change; however, there are slight variations in the save times of the intermediate asymmetric key sizes. An example of these fluctuations can be found in the seventh and ninth runs of the 2048-bit RSA key tests. These variations can quickly be observed under the sample standard deviations for the 2048-bit RSA key size. We can reasonably account for this fluctuation by considering that the scheduling algorithm of the operating system will have some affect on times. We theorize that if the number or test runs is increased, for example above 100, the anomalies in the median save times should fall into the expected range.

Secondly, the RSA encryption time should change as we vary the key sizes for the save operation. The median RSA phase encryption times range from 62.5 ms (512-bit RSA) to 79 ms (4096-bit RSA) according to the results in Table 5. Small increases in save times were expected for the RSA phase, as can be visualized in Figure 17. In comparison, the AES encryption phase save times are significantly longer since we are encrypting the protected object, rather than just an AES key in this RSA encryption phase. The advantage of this approach is that the AES key must be encrypted for each user in the ACL to ensure that only those users specified in the ACL can be granted access to the contained information. Therefore, we can expect that the RSA encryption phase, worst case save time for a 50-user ACL would take roughly 3,950 ms (50 x 79 ms) for a 4096-bit RSA key size. In contrast, it would take roughly 221,450 ms (50 x 4429 ms) if we were to encrypt the document separately for each user using a unique symmetric AES key. This last value clearly indicates that it would not be an appropriate approach to re-encrypt the entire document with separate symmetric keys for each user.

RSA Key Size (Bytes)		Iteration (Time in ms)										Statistical Summary				
		First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Mean	Median	Min	Max	STDEV
512	AES Save	4641	4312	4469	4375	4469	4391	4500	4391	4515	4375	4443.8	4430.0	4312	4641	94.751
	RSA Save	78	63	62	63	62	62	62	62	63	63	64.0	62.5	62	78	4.944
	Total Save	8469	8141	7609	7922	7515	7735	7563	7656	7531	7750	7789.1	7695.5	7515	8469	307.541
1024	AES Save	4531	4390	4532	4422	4469	4313	4468	4375	4484	4375	4435.9	4445.0	4313	4532	72.680
	RSA Save	78	63	62	78	78	62	63	63	63	62	67.2	63.0	62	78	7.465
	Total Save	8360	8265	7672	7937	7609	7672	7562	7704	7547	7672	7800.0	7672.0	7547	8360	291.375
2048	AES Save	4625	4375	4515	4437	4532	4437	4531	4438	4531	4438	4485.9	4476.5	4375	4625	73.074
	RSA Save	78	62	78	63	62	63	63	62	78	78	68.7	63.0	62	78	8.015
	Total Save	8453	8219	7625	8000	7641	7766	7609	7906	10672	7719	8161.0	7836.0	7609	10672	924.871
4096	AES Save	4516	4297	4468	4375	4468	4390	4468	4390	4468	4375	4421.5	4429.0	4297	4516	66.155
	RSA Save	93	93	94	94	79	78	79	79	78	78	84.5	79.0	78	94	7.764
	Total Save	8391	8235	7625	7859	7609	7578	7578	8657	7563	7672	7876.7	7648.5	7563	8657	402.193

Table 5: Server save times over several runs and corresponding statistical summary for asymmetric key size variations. Iterations use 10MB of character data, with a 1-subject ACL, and a 256-bit AES symmetric key. By varying the RSA key size we are able to determine that key size has very little effect on save time. For example, if we compare the median save time of the RSA phase for a 512-bit key (62.5ms) to the median save time of the RSA phase for a 4096-bit key (79ms), we observe that this is a very small amount of time relative to the total save times for these two key sizes. Fluctuations in individual total save times, such as the total save times for the seventh and ninth runs for 2048-bit, are most likely the result of variations in scheduling timing. This fluctuation at the 2048-bit key size is made obvious by its uncharacteristic sample standard deviation (924.871) for total save time when compared with the sample standard deviations of the other key sizes. Neither of the sample standard deviation values of either the RSA-phase or AES-phase of the save operation, appear to have extreme values, indicating that the deviation is in the parsing phase of the save for the 2048-bit key size.

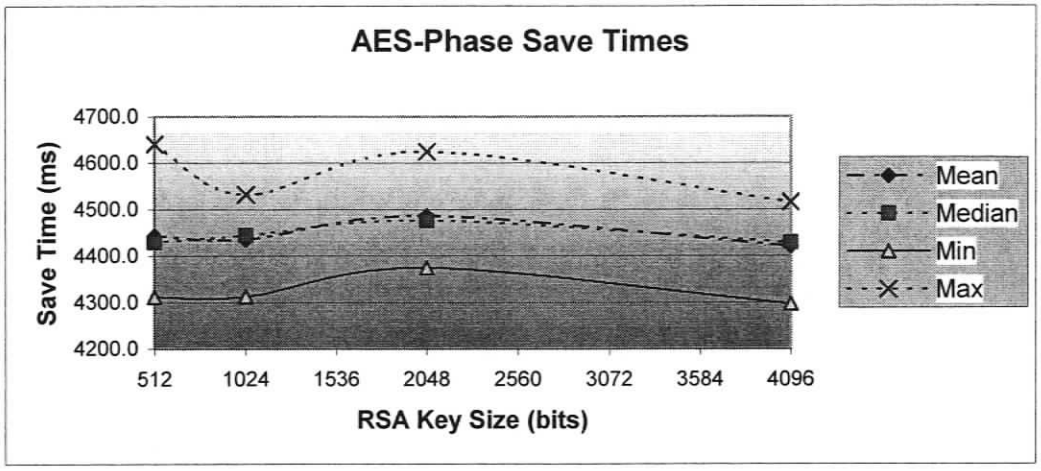


Figure 16: AES-phase save time statistics for varied RSA key sizes.
The median time for the AES-phase of the save operation does not change as we increase the symmetric key size. At the 2048-bit RSA key size we have an anomaly, most likely due to scheduling times. We would expect the anomaly to correct itself if we increased the sample size of the experiment. By performing 100 or more iterations we should notice a flattening of the median curve. This anomaly can be confirmed by the uncharacteristically large sample standard deviation recorded in Table 5 for the 2048-bit key size.

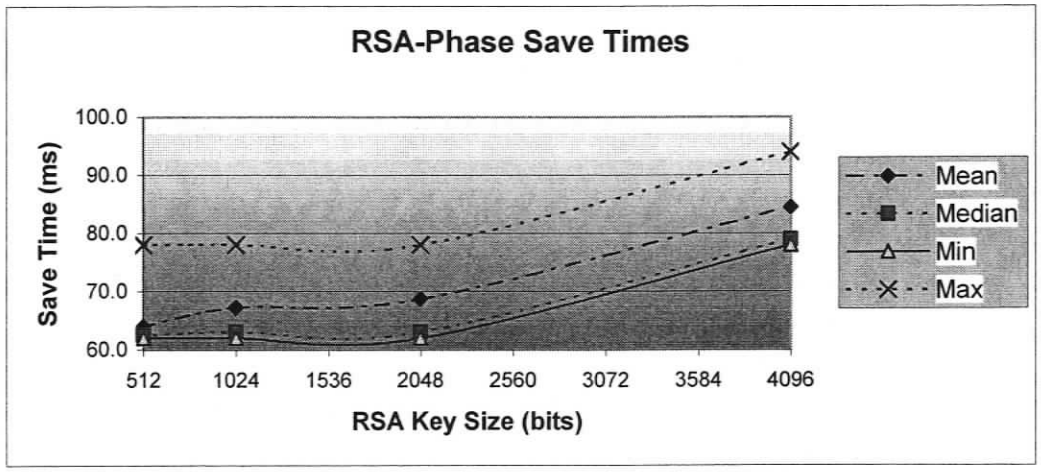


Figure 17: RSA-phase save time statistics for varied RSA key sizes.
As we increase the RSA key size we notice a slight increase in save times. For instance, there is a steady increase in time from 2048-bit to 4096-bit that may seem significant; however, the change in time is roughly 15ms, not a significant amount of time for such a large increase in protection bits.

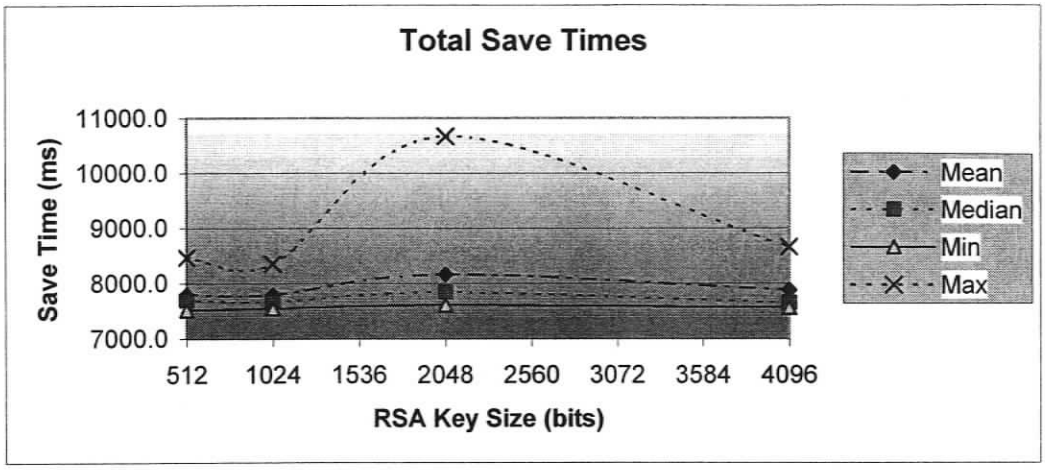


Figure 18: Total save time statistics for varied RSA key size.
Changing the RSA key size has very little effect on the overall save times. The median time above shows a virtually flat line, indicating that the increase in RSA key size has negligible effect on overall save time.

Figure 18 visualizes the total save time statistics for the 10MB file as we increase the RSA key size. With the exception of the anomaly regarding the 2048-bit save times (discussed above), there is very little change in overall save times. We can conclude from the median time results that the effect of increasing the RSA key size has very little effect on total save times, thus satisfying our first target.

5.1.4 Effects of ACL Size

Another variable affecting the save time operations is the length of the ACL. More subjects in an ACL should warrant small increases in the save operation. As we have already seen, the increase of ACL entries effects the RSA encryption phase of the save operation.

The approach used in this experiment is similar to the steps used above for the asymmetric key. Two experiments were performed for the ACL size. The first experiment was conducted by creating ORCS documents containing 10MB of character data, a 256-bit AES symmetric key, an RSA asymmetric key length of 1024-bits, and using ACL lengths from 1 to 10 subjects (with the owner having the control access right and all others possessing read access). The second experiment was the same as the first, except we ranged the ACL length from 1 to 50 subjects using increments of 10 subjects.

The performance experiments were conducted by invoking multiple save runs against ORCS documents with all parameters fixed, except for the one particular parameter being validated in the particular test. By fixing all parameters within reason, except for the ACL length, we collect timing data to validate our claim that changes in the ACL size should have relatively low overhead in regards to increased CPU time.
In the following tables and figures, we examine what effect ACL changes have on the overall save time. Small changes in the size of an ACL should warrant small changes in time, as can be seen in Table 6 and Figure 19.

In Table 6, we can see that the median time it takes to save a 10MB file (7,861 pages of character data), using 1024-bit RSA key sizes, takes roughly seven to eight seconds. Adding additional subjects to the ACL of the file has very little effect on the save time. According to Table 5, it should take roughly an additional 63ms for each additional user in the ACL. In fact, according to Table 6, adding a single user to the ACL costs anywhere from –101.5ms to +101.5ms (based on median times), and a mean time difference of 14.1 ms for each additional subject over the ten subjects. All of these values fall within the median of 63ms that is indicated in Table 5 as the time needed for the RSA phase of the save operation for an object with a 1024-bit key.

Fluctuations in time can be attributed to scheduling variability in the operating system, as can be seen in the maximum times in Figure 19. The negative save time required for adding a subject in the above ACL statistics, and the relatively long range, are attributed to the fluctuations in the sample standard deviations found in Table 6 for values found for ACL sizes {1,4,5,9}. We can conclude that small changes in ACL length result in very small changes in overall save times.

To obtain a better understanding on how ACL length affects the save times, we need to test with considerable longer ACL's. Table 7 and Figure 20 depict save times for data sets containing ACL's with lengths ranging from 1 to 50 users in 10 user increments. The RSA key size and other parameters are the same as those for Table 6 and Figure 19.

# Subjects in ACL	Iteration (Time in ms)										Statistical Summary				
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Mean	Median	Min	Max	STDEV
1	8391	8187	7672	7875	10563	7563	7547	7797	7609	7609	8081.3	7734.5	7547	10563	916.338
2	8375	8219	7656	8015	7640	7750	7532	7594	7656	7688	7812.5	7672.0	7532	8375	288.140
3	8469	8281	7688	7844	7687	7781	7656	7828	7750	7640	7862.4	7765.5	7640	8469	282.295
4	11438	8203	7719	10906	7625	7750	8078	7844	7656	7719	8493.8	7797.0	7625	11438	1428.959
5	8500	8187	7641	7953	7766	7922	10687	7750	7875	7813	8209.4	7898.5	7641	10687	905.016
6	8500	8266	7766	7890	7688	7828	8219	7890	7703	7782	7953.2	7859.0	7688	8500	276.573
7	8594	8266	7813	7969	7813	7734	7828	7891	7859	8000	7976.7	7875.0	7734	8594	262.713
8	8563	8234	7875	8078	7843	8016	7906	7937	7875	7969	8029.6	7953.0	7843	8563	220.849
9	8531	8343	7687	7922	7687	7781	7797	7906	10734	7797	8218.5	7851.5	7687	10734	927.253
10	8562	8469	7860	8000	7765	7907	7797	7891	7844	7859	7995.4	7875.5	7765	8562	282.130

Table 6: Server save times over several runs and corresponding statistical summary for baseline ACL size variation. Iterations use 10MB of character data and a 1024-bit RSA key. Small changes in ACL length result in small changes in overall save times. For instance, the difference of median save times for a three-subject versus four-subject ACL is only 31.5ms. We expected to see slight increases in save times as we added subjects to the ACL; however, there is a decrease when changing from a one-subject ACL to a two-subject ACL. This decrease can be explained by viewing the larger value for the standard deviation for a one-subject ACL. When the standard deviation is higher, we know that the median value is going to be less precise.

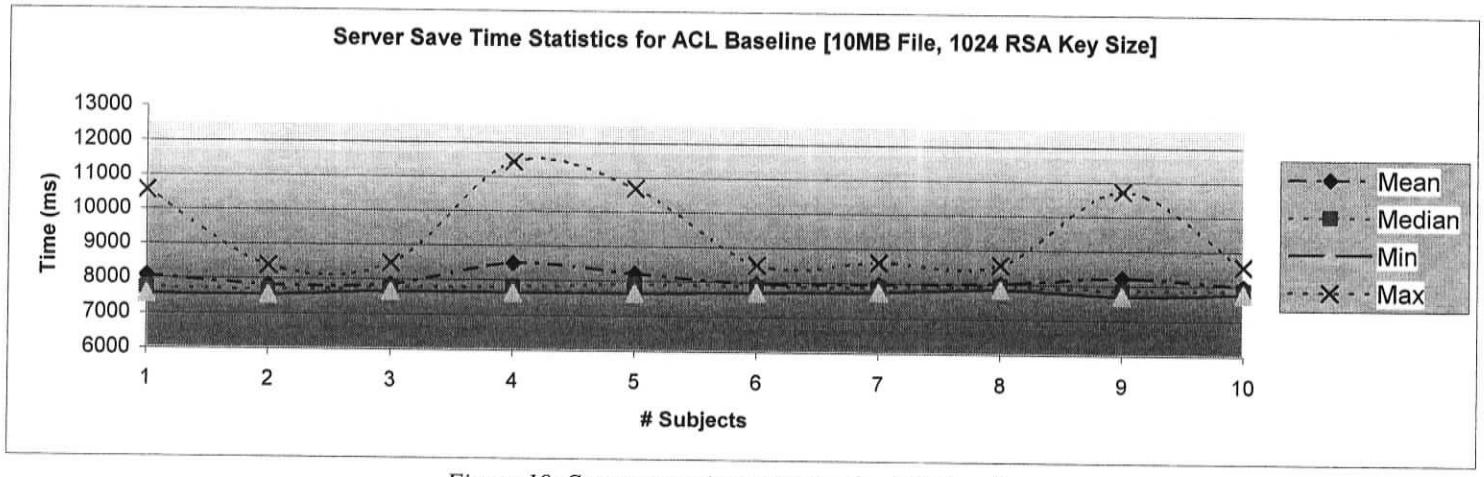


Figure 19: Server save time statistics for ACL baseline. Maximum save times caused by scheduling irregularities result in slight inconsistencies in the median save times. The median save times above confirm our second target that small changes in ACL length result in small increases in total save time.

# Subjects in ACL	Runs (Time in ms)										Statistical Summary				
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Mean	Median	Min	Max	STDEV
1	8672	8875	8360	8110	7859	7875	7828	7969	7719	7875	8114.2	7922.0	7719	8875	392.871
10	9281	8969	8437	8625	8343	8500	8437	8609	11515	8563	8927.9	8586.0	8343	11515	951.795
20	9922	9578	9218	9672	9015	9219	9141	9266	9109	9156	9329.6	9218.5	9015	9922	292.870
30	10688	10531	9969	10078	9765	9781	9671	9937	9766	9937	10012.3	9937.0	9671	10688	339.110
40	11437	11047	10641	10766	10656	10703	12125	14032	10547	13703	11565.7	10906.5	10547	14032	1306.108
50	12032	11734	11328	11297	11172	11375	11203	11390	11391	11265	11418.7	11351.5	11172	12032	265.225

Table 7: Server save times over several runs and corresponding statistical summary for large ACL size variations. Iterations use 10MB of character data and a 1024-bit RSA key. Save times for large changes in ACL length result in save time increases relative to the length increase. Recall that the save time of the RSA phase of encryption on 10MB of data was roughly 63ms for a 1-user ACL with a 1024-bin RSA key. Taking the median time for a 20-subject-ACL, subtracting the median time for a 10-subject ACL, and then finding the average per user gives (9218.5-8586)/10=63.25ms. This value is a rough approximate, as the standard deviation varies over the difference ACL lengths. Thus the cost of increasing the ACL list results in a linear increase in save time.

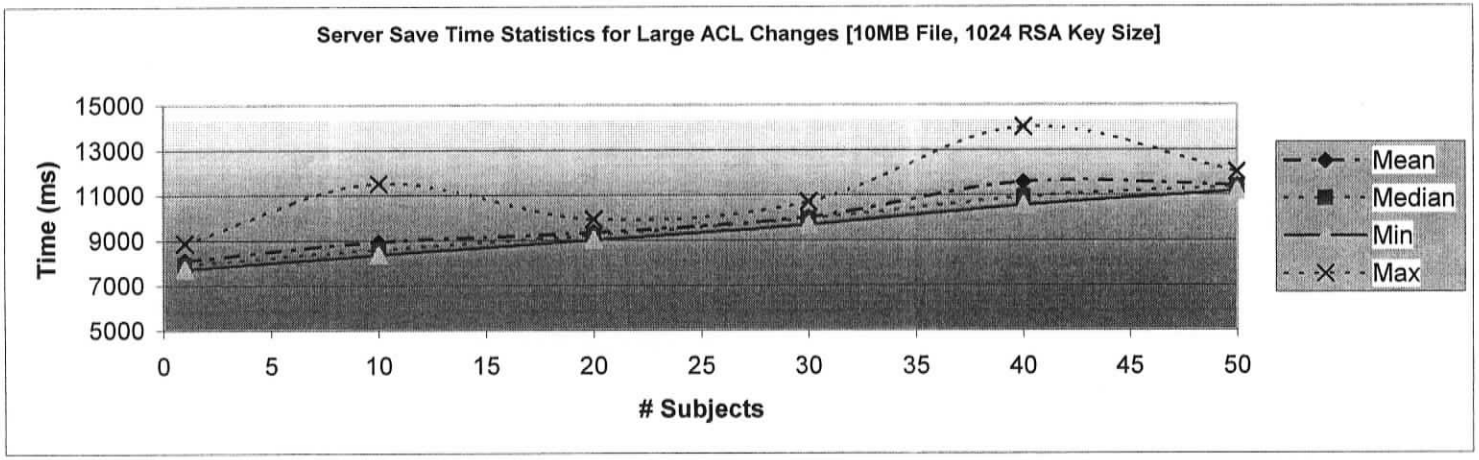


Figure 20: Server save time statistics for large ACL increases. Larger changes in ACL length result in a linear increase in save times relative to ACL length.

When the ACL length increases, we observe that the amount of time it takes to save the document increases in time relative to the length of the ACL. In fact, this linear increase in time can be directly established by referring back to the encryption times we examined earlier. Recall that the median time it took to encrypt a file containing 10MB of character data using an RSA key size of 1024-bits and an ACL length of one was 63 ms. If we then compare the save times for a document of the same size, but with differing ACL lengths we observe the following:

-ACL length = 1 Save time (median) = 7922 ms
-ACL length = 10 Save time (median) = 8586 ms

From these values we can see that the difference in save times for the two documents is 664 ms. Dividing this number by 10 subjects we obtain the time difference per user, resulting in 66.4 ms per user. This number is within range of the expected 63 ms per user RSA encryption time. This value, however, is a rough approximate as the standard deviation varies over the difference ACL lengths.

We can thus conclude that ACL length results in linear time growth in accordance with asymmetric key encryption times.

5.1.5 Effects of Data Size

We have tested the framework with data files of various sizes to obtain examples of how this model scales when we increase the size of data. Table 8 depicts the results obtained for multiple runs against the various files using Environment 1.

These tests were done on ORCS formatted files containing various amounts of data. The test was centered on one of the most time consuming tasks: saving the data to an ORCS file. The file had only one person in the ACL, namely, the owner. Data sizes spanned two different ranges: the first range was from 1-100KB and was used to establish a baseline for performance expectation over small data size increases (Figure 21A); the second range was from 1-15MB in 5MB increments and was used to establish an understanding of how the algorithms performed as size increased over large data size increments (Figure 21B).

The wide time gap between the minimum and maximum in several of the sizes in the tables below, are a result of, as we have mentioned before, the scheduling algorithm running on the operating system. By viewing the values in the standard deviation column of Table 8, these performance gaps can be easily recognized.

In Table 8 and Figure 21 we have established that on a laptop in Environment 1, we have save times that increase linearly with respect to data size. Now that we have addressed both the encryption time and ACL length, we still desire to show that a change in environment does not have any significant effects on save times.

Data Size (KB)	Runs (Time in ms)										Statistical Summary				
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Mean	Median	Min	Max	STDEV
1	541	451	421	430	411	1042	380	380	441	1602	609.9	435.5	380	1602	400
5	561	441	420	1612	410	391	430	400	391	401	545.7	415.0	391	1612	378
10	541	450	441	421	400	1612	3194	420	1572	481	953.2	465.5	400	3194	920
15	591	430	1622	421	420	400	381	391	941	871	646.8	425.5	381	1622	400
20	581	420	421	420	401	361	430	430	471	420	435.5	420.5	361	581	58
25	1302	420	391	470	411	390	371	1662	420	952	678.9	420.0	371	1662	464
30	611	420	411	410	390	421	420	391	431	450	435.5	420.0	390	611	64
35	651	410	461	391	390	1602	450	441	421	450	566.7	445.5	390	1602	371
40	3566	410	1652	431	431	370	431	981	1642	410	1032.4	431.0	370	3566	1026
45	631	410	471	981	390	431	400	400	421	441	497.6	426.0	390	981	184
50	641	430	451	480	451	451	1622	410	461	441	583.8	451.0	410	1622	370
55	4086	441	1682	471	411	481	440	431	932	450	982.5	460.5	411	4086	1161
60	1392	430	431	431	470	1602	421	431	420	1673	770.1	431.0	420	1673	547
65	631	1642	440	431	431	420	431	390	440	1652	690.8	435.5	390	1652	508
70	631	471	441	1642	2744	401	430	401	431	401	799.3	436.0	401	2744	782
75	3615	451	421	450	471	401	380	400	451	430	747.0	440.0	380	3615	1008
80	641	451	1042	1632	410	1592	431	441	1642	1071	935.3	841.5	410	1642	531
85	651	450	421	471	541	421	420	451	450	451	472.7	450.5	420	651	72
90	631	1082	440	471	440	1612	441	400	441	421	637.9	441.0	400	1612	399
95	620	461	470	501	421	420	491	431	991	391	519.7	465.5	391	991	177
100	631	430	461	510	450	461	691	491	1623	450	619.8	476.0	430	1623	363
1024	1232	961	872	862	881	941	911	881	932	911	938.4	911.0	862	1232	108
5120	3996	3836	3725	4076	3746	4977	3635	4697	3765	6740	4319.3	3916.0	3635	6740	960
10240	9424	10084	10675	8332	8212	7941	8513	8252	8151	8402	8798.6	8367.0	7941	10675	932
15360	16984	21280	31786	17505	11537	11476	12027	11206	11326	11156	15628.3	11782.0	11156	31786	6695

Table 8: Environment 1 save times over several runs with corresponding statistical summary for data size increases. Iterations use a 1-subject ACL, 512-bit RSA key, 256-bit AES symmetric key, and a 2-level DOMParser. Increases in save times are minimal for 5KB data size increases, as can be observed by the median times for files sized from 1-100KB. Relatively wide gaps in individual times for a particular data size, such as the min and max save times for the 90KB data size, are most likely attributed to scheduling at the operating system level. Small changes in data size result in small changes in save times. Furthermore, large changes in data size, such as going from 1KB of character data to 15MB of character data, increases in acceptable proportions. Observe the median time for both 1KB and 15,360KB data sizes, note that the 15MB file takes roughly 27 times longer to save than the 1KB, while it is 15,360 times larger. Thus we can conclude that our third target, increase in size produces acceptable time increases, is also satisfied.

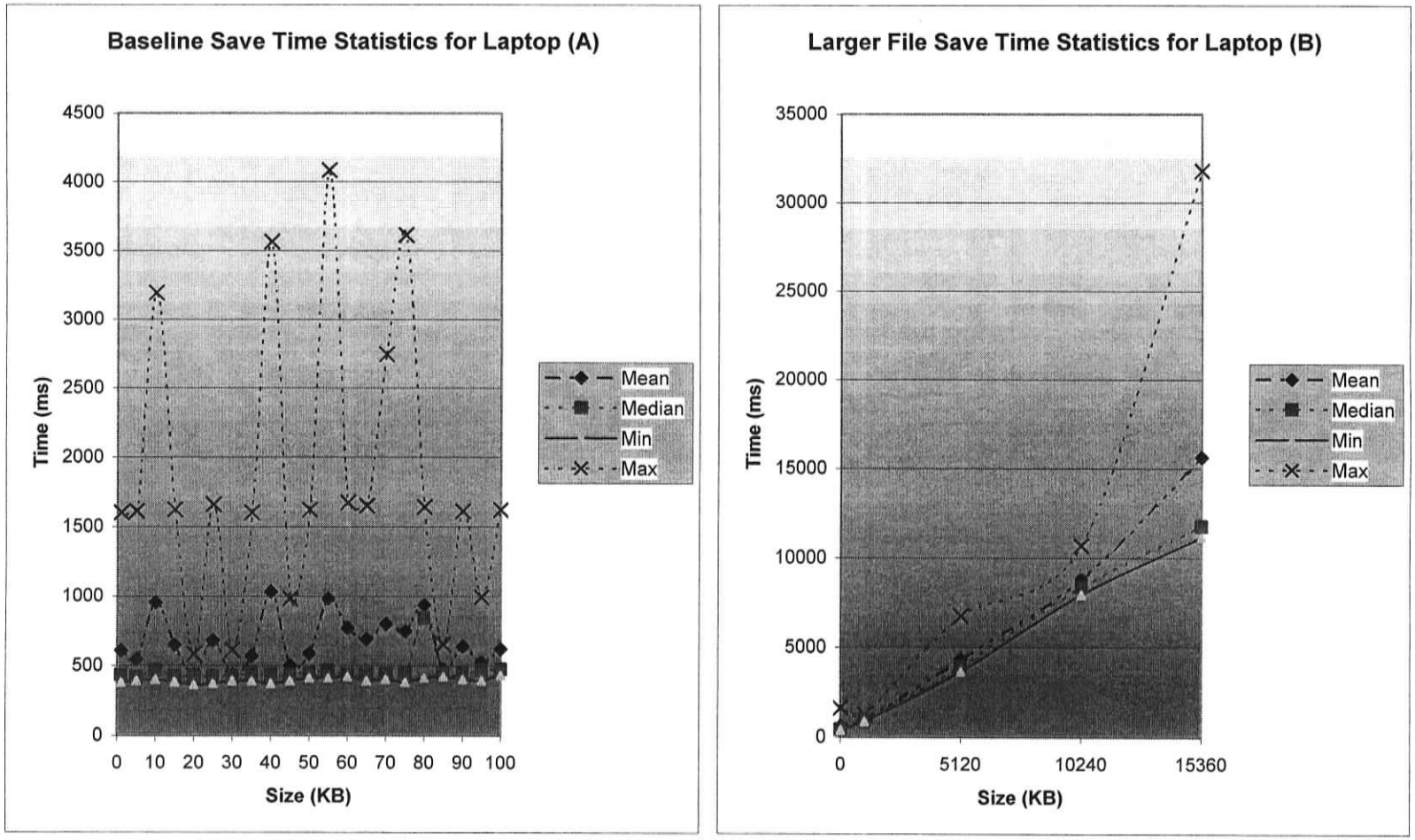


Figure 21: Laptop save time relative to size: (A) Baseline sizes, (B) Larger file sizes.
The median save times for both the baseline and larger file sizes remain close to the minimums. In the baseline, there is one irregularity in the median times at 80KB; this results from several runs where the scheduling algorithm caused longer save times. Again we see that the median save time is essentially linear.

5.1.6 Effects of Environment

Table 8 and Figure 21 depict the save times for objects in Environment 1; Table 9 and Figure 23 focus on the save times for objects in Environment 2. It would be desirable to find that the runtime in Environment 2 shows improvements in save times since it has faster processors, a greater number of processors, and a larger amount of physical memory. The results in Table 9 were obtained by using the same parameters as the tests used for Environment 1. The JVM in use is also the same as that used in Environment 1; however, we are able to use more physical memory, and thus have changed the –Xmx parameter accordingly.

In order to compare the runtimes of the two environments, we will depict the median save times, collected from Table 8 and Table 9, for files ranging from 1KB to 15,360KB in size. We observe in Figure 22 that Environment 1 (Laptop) begins to loose ground on Environment 2 (Server) when the file sizes begin to grow; however, there is no significant gain until the sizes are approaching the 10's of megabytes. We have validated and confirmed our last target for performance testing.

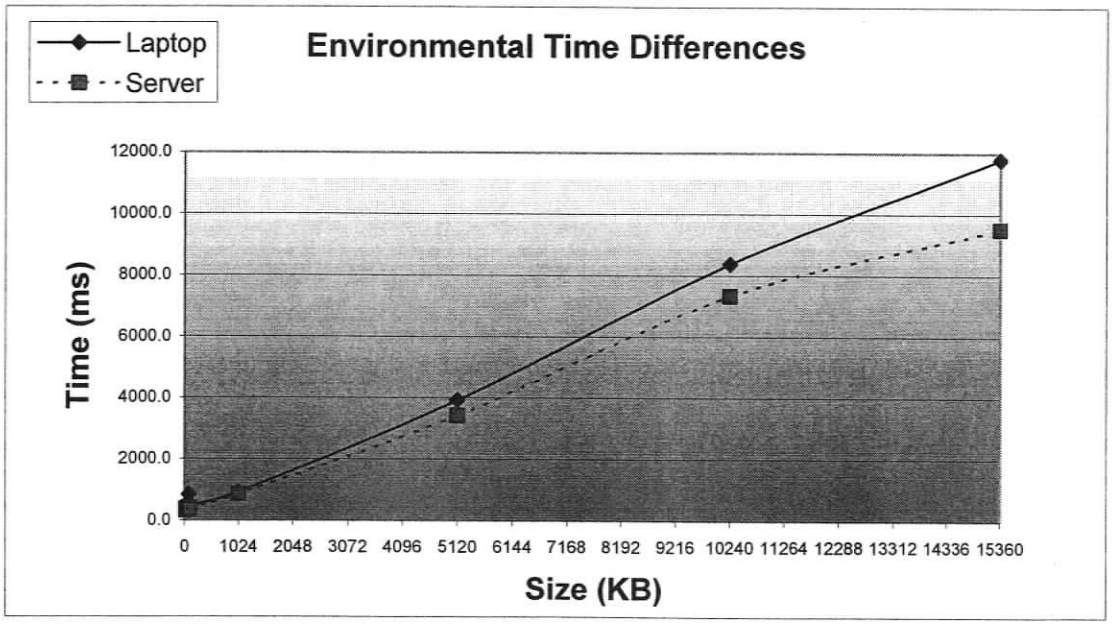


Figure 22: Environmental save time differences over varying data sizes. The server environment outperforms the laptop as the data sizes increase. Notice the increased distance between the time values of the server and laptop as the size increases past the 5MB size.

Data Size (KB)	Iteration (Time in ms)										Statistical Summary				
	First	Second	Third	Fourth	Fifth	Sixth	Seventh	Eighth	Ninth	Tenth	Mean	Median	Min	Max	STDEV
1	406	344	375	360	344	328	328	296	343	312	343.6	343.5	296	406	31.447
5	406	328	391	312	313	328	313	391	360	360	350.2	344.0	312	406	36.282
10	422	312	328	406	296	375	297	281	359	297	337.3	320.0	281	422	50.151
15	438	391	359	375	328	328	328	375	343	297	356.2	351.0	297	438	40.389
20	484	359	406	328	281	312	328	359	328	391	357.6	343.5	281	484	57.749
25	391	359	360	344	328	453	219	469	343	391	365.7	359.5	219	469	69.585
30	454	406	406	328	390	344	312	297	375	328	364.0	359.5	297	454	50.122
35	406	422	328	344	360	328	531	329	375	250	367.3	352.0	250	531	74.728
40	469	391	422	266	391	172	375	329	375	406	359.6	383.0	172	469	85.214
45	422	344	344	328	359	406	313	391	391	329	362.7	351.5	313	422	37.292
50	468	328	203	359	359	359	375	437	391	406	368.5	367.0	203	468	71.374
55	484	328	406	391	344	391	329	329	407	375	378.4	383.0	328	484	49.063
60	468	359	422	313	375	453	328	406	406	375	390.5	390.5	313	468	50.300
65	437	422	421	453	390	328	297	312	391	438	388.9	406.0	297	453	56.797
70	406	328	344	375	406	390	328	250	406	344	357.7	359.5	250	406	49.531
75	500	328	422	312	375	391	390	375	391	406	389.0	390.5	312	500	51.370
80	485	360	250	391	375	359	344	328	359	344	359.5	359.0	250	485	58.163
85	453	359	391	375	344	312	344	390	297	343	360.8	351.5	297	453	44.477
90	500	344	344	453	375	328	375	359	391	406	387.5	375.0	328	500	53.490
95	453	328	265	406	406	375	375	359	359	375	370.1	375.0	265	453	50.096
100	468	390	312	375	328	359	375	391	390	359	374.7	375.0	312	468	42.222
1024	1000	906	813	891	906	844	828	844	844	860	873.6	852.0	813	1000	54.716
5120	3531	3360	3375	3500	3406	3422	3406	3422	3468	3407	3429.7	3414.5	3360	3531	54.035
10240	8063	7750	7390	7344	7406	7250	7312	7313	7313	7188	7432.9	7328.5	7188	8063	267.674
15360	10797	10656	9750	9500	9609	9453	9516	9484	9484	9515	9776.4	9515.5	9453	10797	509.047
20480	15422	12157	12203	12687	12000	11938	12016	12109	11875	11938	12434.5	12062.5	11875	15422	1074.733
25600	17703	15625	16031	16000	16079	15687	15562	15906	15750	15828	16017.1	15867.0	15562	17703	617.868
30720	20766	20062	19562	19984	19875	20140	19984	19953	19891	19969	20018.6	19976.5	19562	20766	303.639
35840	25187	23500	44531	22782	20156	19625	19579	19735	19656	19735	23448.6	19945.5	19579	44531	7676.349
40960	28985	23469	24187	24406	23937	24203	24016	24187	23796	23531	24471.7	24101.5	23469	28985	1614.138
46080	31000	29125	29125	29719	28875	28343	28532	28984	28797	28453	29095.3	28929.5	28343	31000	779.199

Table 9: Environment 2 save times over several runs with corresponding statistical summary for increasing data sizes. Iterations use a 1-subject ACL, 512-bit RSA key, 256-bit AES symmetric key, and a 2-level DOMParser. Save times for Environment 2 follow the same pattern as those in Environment 1 (Table 8); however, Environment 2 begins to perform at a better rate than Environment 1 as the data size increases above 1MB toward the larger data sizes. The standard deviation for our sample runs also become larger and more obvious with the larger data sizes because the larger data sizes are more likely to demand more than one time slice.

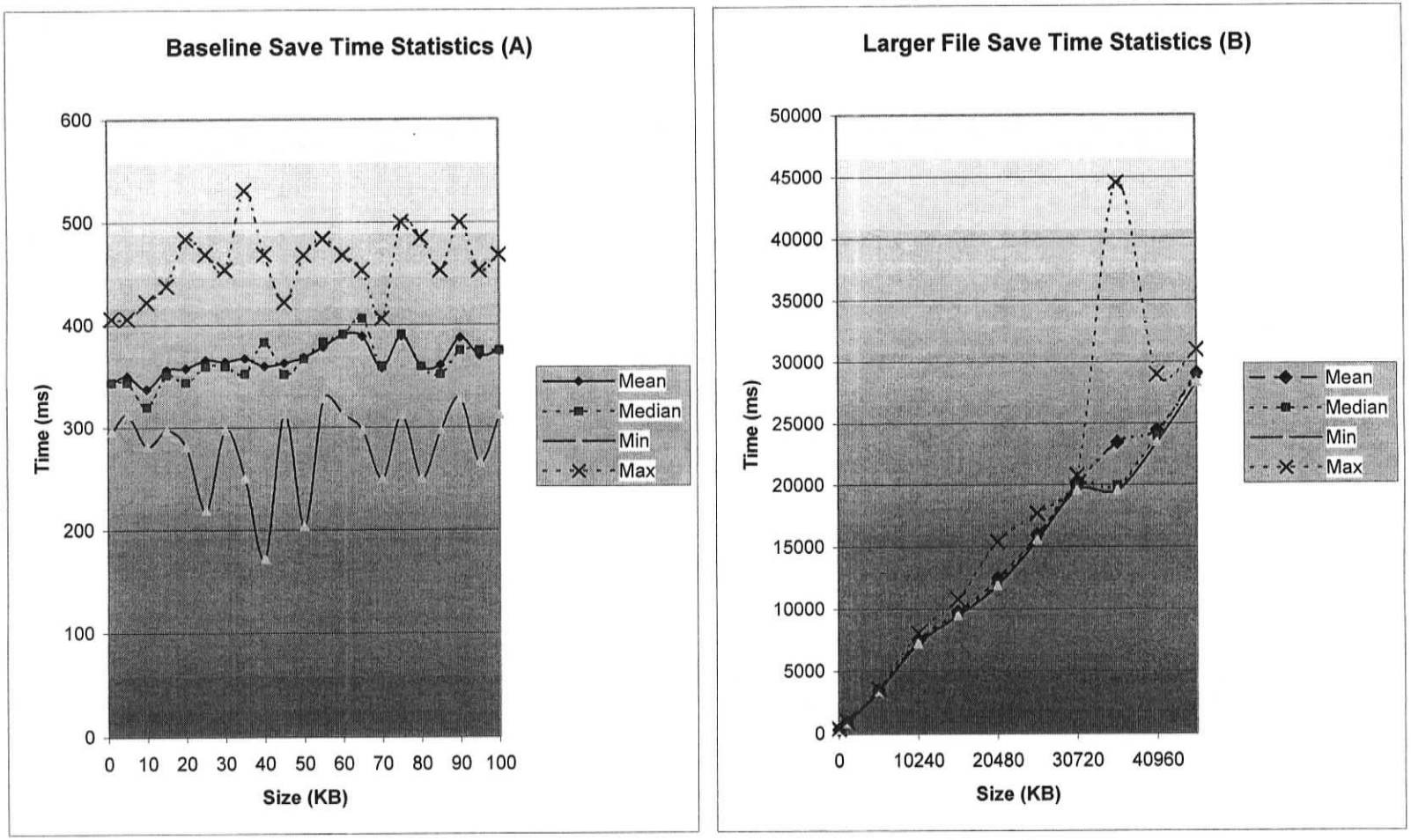


Figure 23: Server save time relative to size: (A) Baseline sizes, (B) Larger file sizes.
Again, similar to Environment 1, we see that the median save time is essentially linear. The large spike in max values at the 30MB size is most likely caused by scheduling irregularities of the process in the operating system.

Before closing the performance discussion, a comparison against some common text editors is warranted to put our timing values into perspective. By populating a document with the same character data that was used for the ORCS tests, performance tests against the save operation were performed for three text editors, Microsoft Notepad, Microsoft Word 2003, and Adobe Acrobat 7. For these tests we copied the exact data from the ORCS tests into a .txt file for the Notepad iterations, a .doc file for the Word iterations, and performed conversion on a .txt file to generate a .pdf file for the Acrobat iterations. When the respective editor saved the data to disk, a digital clock was used to estimate the total save times, resulting in rough estimates for a sample comparison. These results are in Table 10.

At the 15MB file size ORCS outperforms the other text editors, as can be observed by the save time values for Environment 1 across the three editors.

At the 35MB file size, ORCS in Environment 1 cannot process the operation because of memory restrictions caused by the JVM garbage collection and heap settings. This limitation could possibly be overcome by changing the heap parameters of the JVM or by implementing ORCS in a pre-compiled language such as C++. Microsoft Notepad performs without error; however, when we compare the times under Environment 2, ORCS is able to outperform Notepad approximately four times faster (while consuming considerably more physical memory). Microsoft Word 2003 performs its save operation slightly slower than ORCS, as does Adobe Acrobat 7. We need to keep in mind the fact that Adobe Acrobat also has the costly operation of converting existing data to PDF. This conversion against a 35MB text file takes roughly 823 seconds in Environment 1.

Finally, at the 45MB data size under Environment 2, only Microsoft Notepad, ORCS and Adobe Acrobat 7 are able to complete the save operation. ORCS cannot perform the operation under Environment 1 because of memory constraints (as noted above), and Microsoft Word 2003 fails with a warning message stating the following:

> You have exceeded the maximum number of pages supported by Microsoft Office Word.

The maximum number of pages that Word was able to paginate appeared to be 32,767. After this warning message, different feature of the Word environment did not behave as expected.

These results are not unexpected. Recall from Table 3 that the following file-size to page-length ratios hold (15MB : 11,657pages), (35MB : 26,840pages), (45MB : 34,431pages).

	ORCS		Microsoft Notepad		Microsoft Word 2003		Adobe Acrobat 7.0	
Data Size (MB)	Env. 1	Env. 2	Env. 1	Env. 2	Env. 1	Env. 2	Env. 1	Env. 2
15	12	10	35	30	17	10	18	10
35	Z	20	90	75	35	22	45	27
45	Z	29	125	95	X	X	73	35

*Table 10: Approximate save times (measured in seconds) for comparable text editors. The **Z** marker in the ORCS columns indicates that the test was not completed due to heap size boundaries of JVM for the particular environment. The **X** marker in the Microsoft Word 2003 column indicates that the test was not completed because the document exceeded the maximum size for Word.*

Additionally, we should briefly compare the sizes of files using the different editors. The 35MB data set, in Environment 2, produced a 25MB file when saved as a PDF, and 118MB when saved using Microsoft Word 2003. The same data set saved using ORCS resulted in a file size of roughly 61MB. The file size for Notepad was 35MB, as it contains no additional markup. These tests were performed against Word and Acrobat files that contained no additionally protection attributes. We would expect save times and file sizes to change once protection mechanisms were placed on these other document types. Further testing of these formats is planned for the future.

In summary, the ORCS model performs well for both small and larger amounts of data. The changes in encryption key strength, ACL length, and environment all have a minimum impact on performance. Both key strength and ACL length cause a slight increase on demand for CPU time. Environment changes result in increased performance as CPU speed and the number of processors increase. Lastly, implementing ORCS using a platform-specific language such as C++, rather than Java, could address the inability for ORCS to save the larger file sizes in Environment 1. The change to C++ would remove the platform-independent feature of ORCS; however, note that none of the other applications can claim to be platform-independent, a quality found in ORCS. The other tools had limitations of their own as well. For example, Notepad has no protection mechanisms, Word has problems handling large amounts of data and creates stored

documents that are much larger than any of the other tools, and Acrobat requires a significant (comparably) amount of time to generate the PDF document from a source document.

5.2 Security Analysis

In this section we will first explore and analyze the security of basic features of the ORCS model and then discuss theoretical extensions to this model. The basic features include the following: user registration, information protection update, and document distribution. Extensions include collaboration scenarios, and document synchronization.

We will use an informal attack tree to summarize the types of attacks that may be attempted against the ORCS framework. This will be followed by an informal discussion on the identified security threats.

5.2.1 Discussion and analysis of ORCS features

The basic features of the ORCS model involve user registration, information protection update, and document distribution.

User registration with the ORCS model would require a security administrator, in our case, the trusted certificate authority (see Definition 21) to register the subject with the system. The security administrators are responsible for verifying the credentials, background check, and personal identification information of subjects before issuing a digital certificate for the public key of a subject.

An information protection update would be required in a real-time collaboration environment where multiple subjects work on the same data simultaneously. The ORCS prototype does not implement this functionality, as the prototype represents a stand-alone application for processing ORCS documents rather than for real-time collaboration. Instead, the prototype implements information protection update as the save operation of the tool, taking all of the protected information and updating the secured document on a storage device.

Document distribution is the primary feature that is secured by the ORCS prototype. We wanted to create a tool that would allow us to test the feasibility of creating documents that would remain protected independently of all forms of data transport.

Document distribution may be accomplished in many different ways in an operating environment. These mechanisms may include the following: email attachment, FTP, SCP, HTTP, diskette, memory key, etc. In order for document distribution to be secure, the owner must produce a document that has been secured for transport. Three principles are required for the basic case of document distribution:

(1) The owner (A)
(2) The consumer (B)
(3) The trusted certificate authority (C)

The following protocol is derived from the Needham-Schroeder Public-Key Protocol. In the document distribution protocol the goal is to send a document, secured, from *A* to *B* and have no other principle gain access to the data. Once the data is in the possession of *B,* the rules of ORCS must apply. Here are the steps involved in the idealized Needham-Schroeder Public-Key Protocol [37][1]:

(1) $A \rightarrow C : \{A, B\}_{K_c}$

(2) $C \rightarrow A : \left\{ \overset{K_b}{\mapsto} B \right\}_{K_c^{-1}}$

(3) $A \rightarrow B : \{N_a\}_{K_b}$

(4) $B \rightarrow C : \{B, A\}_{K_c}$

(5) $C \rightarrow B : \left\{ \overset{K_a}{\mapsto} A \right\}_{K_c^{-1}}$

(6) $B \rightarrow A : \left\{ \left\langle A \overset{N_b}{\rightleftharpoons} B \right\rangle_{N_a} \right\}_{K_a}$

(7) $A \rightarrow B : \left\{ \left\langle A \overset{N_a}{\rightleftharpoons} B \right\rangle_{N_b} \right\}_{K_b}$

[1] Idealized protocol derived from Burrows et al. An idealized protocol is a description of the protocol independent of any implementation-specific details [16, y6b].

The steps of the Needham-Schroeder Public-Key Protocol are well known. Steps (1) and (4) from the protocol above, are simply requests for C to send the public keys of the other principle to A and B. Steps (2) and (5) satisfy these requests. The assumption we make in the case of the ORCS prototype is that both A and B initially have the public key of C. Therefore, the only keys that A and B will accept into their keyring (see Definition 19) are those keys they receive in a digitally signed certificate, signed by C's secret key. Consequently, we ensure a certain level of security for the integrity of the public keys.

In step (3) principle A sends a secret to principle B using the public key of B. Step (6) has B send a secret back to A combined with the secret that was passed from A, encrypted using the public key of A. At this point A now has confidence that N_a is a shared secret between A and B; however, B does not yet have confidence in the secret N_b until step (7) when A sends back both secrets encrypted with B's public key.

Historically, there is a well-known weakness in this protocol and their other protocol for shared keys, because it allows a replay attack against the interactions with the certification authority if a key is compromised. This problem is discussed in [16, 17]. As pointed out by Burrows et al., the problem can be addressed by including time stamps in the message passing steps (2) and (5) above.

In order for us to create a save protocol and a document distribution protocol, we first discuss the qualities the above protocol by Needham-Schroeder provides us and then compare it to what we desire in our ORCS framework. Parts of the above protocol are directly needed in our system, such as the ability to trustingly obtain the public keys of principles that are candidates for the document the owner is creating. The other portion can be somewhat simplified and separated into two portions. First, when an owner is creating a document, the owner needs only trust that the public key of the candidate is authentic and current. The owner does not need to share an additional secret with the candidate at the moment of document creation. Second, the need for a secret to be shared between a candidate and the owner is dependent upon the access rights that the owner grants to the candidate. For example, if the candidate is given only the access attribute read, they will not need to share another secret at all. If, however, the owner has given write permissions to the candidate, a secret may be passed back to the owner.

The secret can be anything that the owner wishes to share with the candidate, in many cases this is an integer; however, in our model we choose the secret to be the data that is to be shared (the protected data). When a candidate is given the write access attribute, the secret being returned to the sender is simply the modified data.

Thus, in summary the Needham-Schroeder protocol can be used to exchange keys and secrets. While this protocol was initially designed for use as an authentication protocol for large distributed networks of computers, we could use it in the future towards the design of communication protocols for the live collaboration environment. Key exchange protocols have continuously been an area of research, and we do not attempt to attack this issue in this thesis.

Next, we desire that for each subject in the ACL of an object the data be secured/saved in such a way that no other subject can gain access to the information contained in the protected object.

The essential step we needed for the save operation to secure the data is step (3). Since we assume that the keyring of each subject has only been populated with public keys for trusted subjects (see Definition 25 and Theorem 1 of the ORCS model), we only need to protect the secret using $A \rightarrow B : \{N_a\}_{K_b}$ for each user B in the ACL. However, we now need to answer the question: what is N_a?

For the save operation we use Definition 28 and Theorem 2 which form the basis of N_a. Recall that an object is considered secured if the following holds:

Secured object $SO_{b,c} = \{o_b\}_{SM_c}$

Secured entry $SE_{c,a} = \{SM_c\}_{PK_a}$

and let SE be the set of secured entries for an object (ob) that has n entries in its ACL:

$SE = \{ SE_{c,1}, SE_{c,2}, \ldots, SE_{c,n} \}$

Thus, to secure the object using step (3) we notice that $SE_{c,a} = \{SM_c\}_{PK_a}$ is the same as $A \rightarrow B : \{N_a\}_{K_b}$ when we let ($SM_c = N_a$) and state that ($PK_a = K_b$).

Recall from Axiom 4 that only the subject who has a matching private key for one of the entries in the ACL will be able to extract the symmetric key that is used to free the data. Our beliefs regarding the confidentiality and integrity of the protected data are

warranted, based on the fact that the subjects, via a digitally signed certificate using C's secret key, receive the public keys.

5.2.2 Attack Tree

Attack trees are commonly used to perform security and risk analysis. These trees allow us to record and document attacks in a clear and reusable form [35]. Attack trees have a root node that represents the ultimate compromise that the tree represents. The remaining tree nodes consist of sub-goals that can lead to the compromise that is represented by the root. The branches of the tree can be combined using AND and OR rules to specify that a particular goal becomes realized when either all nodes in the branch (AND), or one node in the branch (OR), are satisfied.

Using an attack tree we can categorize the different ways in which ORCS protection may fail, such as a malicious person trying to extract protected information from an ORCS protected document. When considering an attack tree, we want to consider the different types of threats and vulnerabilities of a system.

> Threats might originate from any of the following:
>
> > natural and physical (fire), unintentional (lack of training), intentional (attack initiated out of the desire for some sort of gain). Possible outsiders include the following: spies, terrorists, criminals, corporate raiders, crackers [a malicious hacker]), insiders or outsiders (disgruntled employees or crackers).

LEGEND:

(1) Physical

1.1 Dumpster Diving
1.2 Wire Tapping
1.3 Emanation Eavesdropping
1.4 Denial of Service

(2) Personnel

2.1 Masquerading
2.2 Social Engineering
2.3 Harassment
2.4 Software Piracy

(3) Communications and Data

3.1 Data Attacks
3.1.1 Unauthorized Copying of Data
3.1.2 Traffic Analysis
3.1.3 Covert Channels
3.2 Software Attacks
3.2.1 Trap Doors
3.2.2 Session Hijack
3.2.3 Tunneling
3.2.4 Timing Attacks
3.2.5 Trojan Horses
3.2.6 Viruses and Worms
3.2.7 Logic Bombs

(4) Operational

4.1 Data Diddling
4.2 IP Spoofing
4.3 Password Sniffing
4.4 Scanning
4.5 Excess Privileges

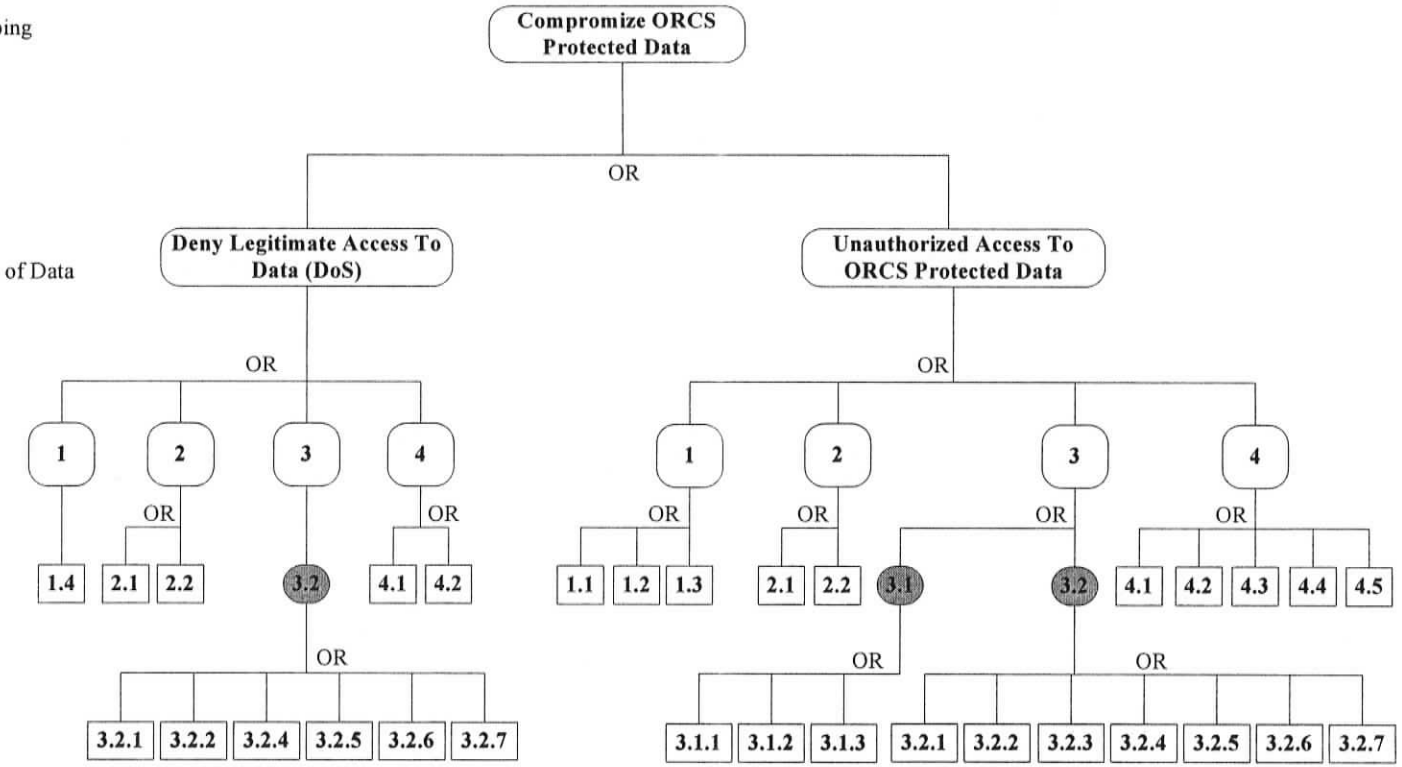


Figure 24: High-level Attack Tree for ORCS.

The main risk to an ORCS system is unauthorized access to protected data; however, denial of access for legitimate subjects is also a concern.

Vulnerability may come from any of the following:

> physical security weakness (unauthorized physical access to resources), natural disaster (destruction by fire, water, etc.), hardware and software (hardware failure, software weakness), media (damage), emanation (recording of electrical signals), communications (interception and attack on messages), human (lack of virtues).

In the first of two attack trees, we use the above threats and vulnerabilities to categorize different types of computer crimes that could be committed to try and obtain protected information from an ORCS protected document (Figure 24). There are two primary objectives that an attacker would have in regards to the ORCS tool: deny the subject access to the data that he rightfully should be able to access, and gain access to data that the attacker should not be able to obtain. We will now give brief explanations for these possible attacks and consider whether each is worth exploring further.

LEGEND:

(1) Physical Related		
1.1	Dumpster Diving	Scavenge through material that has been thrown away.
1.2	Wire Tapping	Eavesdrop on communication.
1.3	Emanation Eavesdropping	Eavesdrop on the electrical signals from computer equipment.
1.4	Denial of Service	Termination or slowing down of a service.
(2) Personnel Related		
2.1	Masquerading	One subject uses the identity of another to gain access.
2.2	Social Engineering	Manipulation of others with the intent of gaining information.
2.3	Harassment	Threatening messages and media with malicious intent.
2.4	Software Piracy	Copying and/or selling illegal copies of copyrighted software.
(3) Communications and Data Related		
3.1	Data Attacks	Attacks on confidentiality, integrity and availability of data.
3.1.1	Unauthorized Copying of Data	Copying of data by a subject who does not have authority.
3.1.2	Traffic Analysis	Monitoring, and analyzing data traffic without authorization.
3.1.3	Covert Channels	Hiding stolen data in seemingly innocent output.
3.2	Software Attacks	Attacks on software to corrupt the normal functionality.
3.2.1	Trap Doors	Backdoor built into a piece of software that bypasses security.
3.2.2	Session Hijack	Taking over a network session that is not your own.
3.2.3	Tunneling	Using one data transfer mechanism to carry data for another.
3.2.4	Timing Attacks	Abuse of race conditions and asynchronous attacks.
3.2.5	Trojan Horses	Insertion of instructions resulting in unauthorized function.
3.2.6	Viruses and Worms	Replicating, modifying code that affect normal operation. Virus depends on another program. Worm is independent.
3.2.7	Logic Bombs	Instructions executed when particular conditions are met.
(4) Operational Related		
4.1	Data Diddling	Modifying data before or after it is entered into the computer.
4.2	IP Spoofing	Form of masquerading where IP addresses are falsified.

4.3	Password Sniffing	Monitor data traffic to obtain user ID's and passwords.
4.4	Scanning	Repeatedly trying an operation varying a value until success.
4.5	Excess Privileges	Breaking into an account that has sufficient permissions to cause damage.

This listing is intended to introduce a wide variety of computer related crimes that could be used, in general, against any system. In Figure 24 we relate, where reasonable, these attacks to our two possible attack goals we stated earlier (unauthorized access to data and denial of access to an authorized user). The second attack tree further explores only those attack scenarios that are more realistic for the ORCS prototype (Figure 25).

5.2.3 Consideration of Possible Attacks

All applications run the possibility of being attacked by malicious users. Of the possible attacks shown in Figure 24, only the following are worth considering as valid attacks: dumpster diving, denial of service, masquerading, social engineering, unauthorized copying of data, traffic analysis, timing attacks, Trojan horses, data diddling, IP spoofing, password sniffing and excess privilege. The other possible computer crimes are attacks that have very little to do with the current ORCS prototype.

When considering the possible attacks that could be attempted against the ORCS prototype, there are several items that could be attacked. They are as follows:

- **An ORCS document**:

  Attacker attempts to deny access to the data by corrupting the contents of the file (data diddling).

  Attacker attempts to do a brute force attack to decrypt the symmetric key (cryptographic).

- **The keystore of a subject who is in the ACL**:

  Attacker attempts to gain access to the keystore by gaining authentication information or by brute force decryption attack (dumpster diving, eavesdropping, social engineering, masquerading, password sniffing, cryptographic).

  Attacker attempts to deny access by corrupting or replacing keystore (denial of service).

LEGEND:

(1) Physical

1.1 Dumpster Diving
1.2 Wire Tapping
1.3 Emanation Eavesdropping
1.4 Denial of Service
1.5 Theft

(2) Personnel

2.1 Masquerading
2.2 Social Engineering

(3) Communications and Data

3.1 Data Attacks
3.1.1 Unauthorized Copying of Data
3.1.2 Traffic Analysis
3.1.3 Cryptographic Decryption Attack
3.2 Software Attacks
3.2.1 Trap Doors
3.2.2 Timing Attacks
3.2.3 Trojan Horses

(4) Operational

4.1 Data Diddling
4.2 IP Spoofing
4.3 Password Sniffing
4.4 Scanning
4.5 Excess Privileges

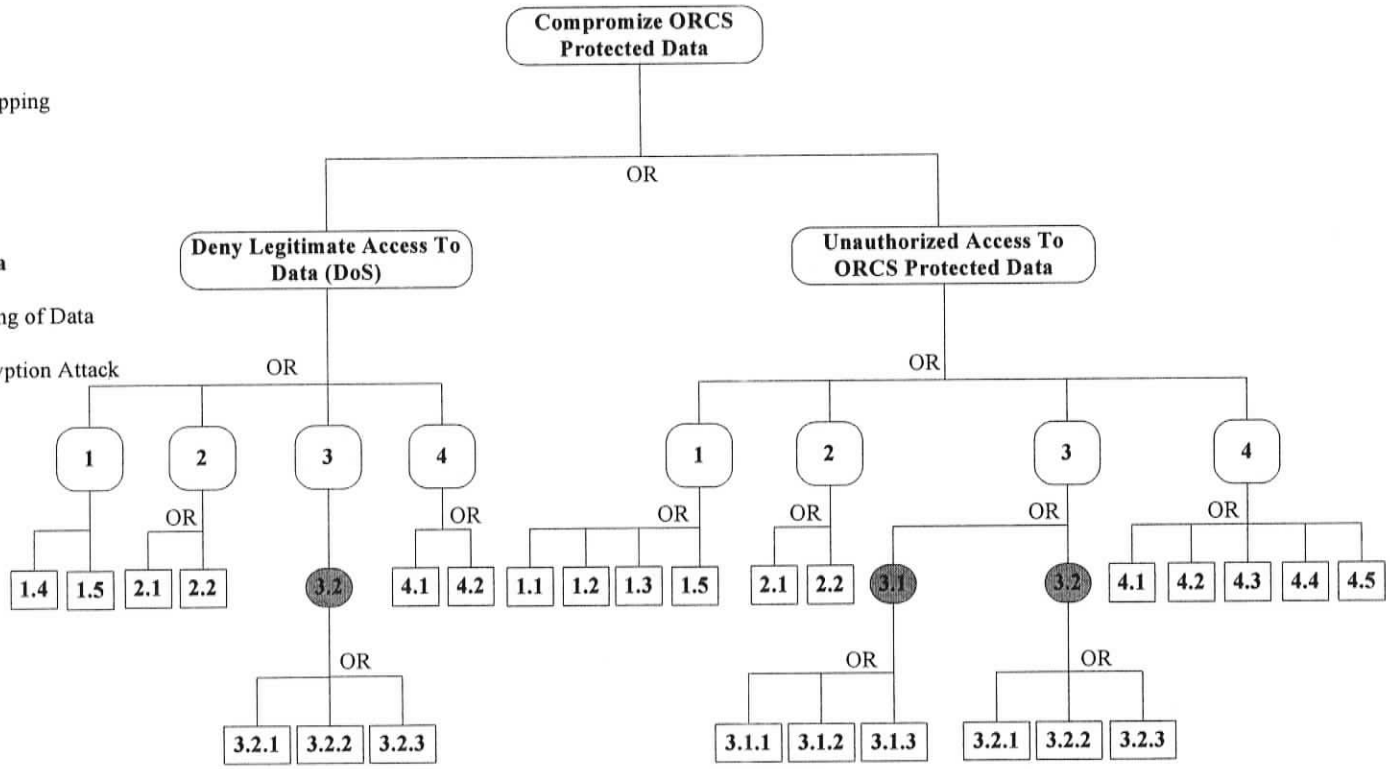


Figure 25: High-level Attack Tree for specific attacks against ORCS.
The main risk to an ORCS system is unauthorized access to protected data; however, denial of access for legitimate subjects is also a concern.

- **The JVM**:
  Attacker replaces the JVM with a modified version that provides access to data (trap door, excess privileges, timing attacks).
  Attacker corrupts the JVM to deny access to data (denial of service).
- **The dependent libraries**:
  Attacker replaces the libraries with a modified version that provides access to data (trap door, excess privileges, timing attacks).
  Attacker corrupts the libraries to deny access to data (denial of service).
- **The certificate authority**:
  Attacker infiltrated CA to generate a false certificate that is provided to the owner who adds the attacker, unintentionally, to the ACL (social engineering, masquerading).
- **The owner**:
  Attacker convinces the owner to give up a secret, such as the password to his keystore (social engineering).
- **One of the subjects in the ACL**:
  Attacker convinces the subject to give up a secret (social engineering).
- **The operating system**:
  Attacker replaces or adds a feature from/to the operating system to provide access to data (trap door, excess privileges).
  Attacker corrupts the operating system to deny access to data (denial of service).
- **The network**:
  Attacker records and analyzes data traffic in an attempt to learn secrets such as the ORCS data, the subject's password, etc. (password sniffing, traffic analysis).
  Attacker overloads network ports to cause denial of service (scanning).
- **The storage device**:
  Attacker steals the storage device to cause denial of service or attempts to learn secrets leading to data access (theft, denial of service, cryptography).
- **The physical computer of a subject in the ACL**:
  Attacker damages the device leading to a denial of service (denial of service).
  Attacker captures the data by taking photographs (social engineering).

We will briefly list some of the types of attacks that could be mounted against the ORCS prototype using the attacks above, and then decide whether they are possible.

- **Dumpster diving**: Potentially leading to the learning of secrets for gaining access to a subject's keystore. Performed by going through trash looking for secrets. Provided users are properly trained in the details of the corporate security policy; dumpster diving should not be fruitful since no privileged data should ever end up in the trash.
- **Denial of service**: Potentially deny a subject access to protected data that he has permission to access. Modification of an ORCS file or any part of the runtime environment the tool is dependent on could result in denial of service. This is not likely, provided that the underlying computer system is trusted. Attackers should not be able to modify the system binaries or operating system, let alone gain access to a protected file.
- **Masquerading**: Potentially pretend to be another subject to gain access to protected data. This is also not likely to occur since the attacker would first have to obtain information required for authentication.
- **Social engineering:** This type of attack is one of the most difficult to prevent, since it preys on human emotion and natural trust. A social engineering attack could be attempted against an ORCS subject by gaining access, physically, to the target machine hosting the ORCS prototype for the target subject. The difficulty here is that the attacker would then have to convince the subject to give up the secret that provides access to the subject's secret key (authentication data). This is not likely if proper training is provided to the user community.
- **Traffic analysis**: Replay attack is a type of traffic analysis. A replay attack occurs when an attacker records some information representing a valid communication exchange and then replays it in the hope that secrets will be revealed. There are no secrets being exchanged over the network in this prototype; therefore, it is not a real concern.
- **Timing attacks**: Timing attacks take advantages of synchronization and race conditions in software. After analysis, it would appear that there are no race conditions in the ORCS prototype; therefore, this attack is not likely.
- **Trojan horse:** This is a realistic concern in the current implementation of ORCS. If an attacker is able to supplant the core libraries, ORCS tool, or JVM with a maliciously modified version, information leakage is possible. This also is common to most environments; however, an attacker would first have to gain privileged access to the host machine of the subject (excess privilege).

- **Data diddling**: An attacker modifies data after it has been entered but before it is used. This can lead to all kinds of problems. The attacker first must gain access to the data and if this is a secure system, gaining access should be a very difficult task.
- **Man-in-the-middle**: is a type of spoofing attack. These are not likely, since we are not responsible for the key exchange mechanism. This will be researched in the future with the live environment considerations.
- **Password sniffing**: Password sniffing is the practice of monitoring (usually network connections) for transactions using user identification and password data. The data is recorded and then used for masquerading. This could be a weakness if the subject were to connect remotely to his machine over a clear-text connection and then use his password. Provided that users are properly trained, this should not be a problem.
- **Excess privilege:** Using an account that has been given elevated privilege that is not warranted for the subject, is commonly known as an excess privilege attack. It results in the attacker having access to a larger set of privileged operations than the subject deserves. This would be a problem with the trusted computing base, rather than ORCS.

In conclusion, the largest security threats to ORCS come from an attacker first obtaining privileged access to the target's host machine. If the security policy (physical, and protocol related) is being followed properly, this should be a difficult task to complete. Using encryption techniques, as we have, is an industry-accepted means of securing information. Using a certificate authority as a trust base for integrity of public keys provides a system that should keep the protected data secured from attack.

Chapter 6 Case Studies

This chapter explores two different security policies and relates them to the owner-retained access control issue. We first explore a patient medical record protection scheme based on the British Medical Association (BMA) Policy, followed by an exploration of the Chinese Wall Policy that is commonly used in the financial sector.

6.1 Patient Medical Records Protection Scheme (PMRPS)

The purpose of this case study is to explore how the ORCS model can be used to protect patients' medical records that are maintained by a hospital or other medical environment. The medical records are either kept in a central database (perhaps attached to a departmental mainframe computer), and are accessible remotely from various terminals (operated by doctors, nurses, etc.), or they are kept locally and travel where the patient is located.

We use in this case study a target security policy, the BMA policy, since this is one of the few policies that was developed originally for securing patient's records. Anderson developed the BMA policy in response to a growing demand for the protection of personal health information in countries including Canada, Britain, USA, and Germany [5]. In this study, we introduce the BMA policy and discuss its strengths and limitations. Concluding with a discussion on how we can establish that ORCS can be used to capture the strengths of the PMRPS and address its limitations [5, 6, 7, 8, 9].

We explore the BMA policy in the hospital environment in Appendix B, where we first identify the actors and elements of the environment, followed by a threat and vulnerability analysis for patient medical records.

6.1.1 Introducing the BMA

We will now introduce the British medical association (BMA) security policy for securing the system and discuss how it responds to the threats that we have identified in Appendix B. It is understood that clinician is defined as a valid medical clinician (doctor, nurse, radiologist, etc.).

The BMA Security Policy includes the following key components:

1. *Access control:* each identifiable clinical record shall be marked with an access control list naming the people or groups of people who may read it and append data to it. The system shall prevent anyone not on the access control list from accessing the record in any way.
2. *Record opening:* a clinician may open a record with herself and the patient on the access control list. Where a patient has been referred, she may open a record with herself, the patient and the referring clinician(s) on the access control list.
3. *Control:* One of the clinicians on the access control list must be marked as being responsible. Only she may alter the access control list, and she may only add other health care professionals to it.
4. *Consent and notification*: the responsible clinicians must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency case or in the case of statutory exemptions.
5. *Persistence:* no one shall have the ability to delete clinical information until the appropriate time period has expired.
6. *Attribution:* all accesses to clinical records shall be marked on the record with the subject's name, as well as the date and time. An audit trail must also be kept of all deletions.
7. *Information flow*: information derived from record A may be appended to record B if and only if B's access control list is contained in A's.
8. *Aggregation control*: there shall be effective measures to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add their access control list already has access to personal health information on a large number of people.
9. *Trusted computing base*: computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be subject to evaluation by independent experts.

In general the policy does little towards defending against social engineering attacks and "dumpster diving"; however, it does stand up well against most of the fundamental goals of security: prevention, detection, and recovery.

Access control, consent and notification, and the trusted computing base modules address prevention. Detection is addressed by the consent and notification and aggregation control. Unfortunately, the policy does not seem to make any preparation for recovery; however, this may be specified in the administration policy.

It seems that the BMA model is extremely dependent on how we define their point <9> "Trusted computing base". The majority of the technological attacks outlined in the attack tree are tied directly to this definition. For example, is it realistic to expect that a doctor will not want to install new applications on her mobile computer? If so, she will usually require administrative rights for that computer, thereby opening a huge potential security risk. If not, which criterion determines the definition of a trusted computing base? The BMA policy also seems to omit all physical security measures.

In summary, the BMA model is centered around two principle ideas: all access to patient data must be authenticated and permitted, and customer notification and consent is required for several key operations. While this is potentially enough to secure patient data in a computer storage system with protected access points, it does not deal with several other issues. Anderson raises the possibility of using encryption to secure the patient data that is consumed in the hospital environment [5]. This encryption can help to achieve a realistic secure computing base. With this in mind, the BMA policy can be used as the foundation for a patient medical record system. We must ensure that the definition of secure computing base is thoroughly evaluated to ensure the security of the entire system. As such, consideration must be given to how we can develop applications that can protect data at all access points not only on the system, but also within the physical environment and networked architectures.

6.1.2 Suitability of ORCS to address BMA security policy

Through the exposure of the interfaces and modules involved in the system, along with their interactions, we can provide a potential software-architecture for the protection system.

In general we desire a system that will enforce the security policy for the patient records. Ideally, a system that will keep all patient data un-accessible/unreadable until a valid user request is processed should be created. One approach to obtain this is to store all data in an encrypted format with associated security restrictions. In order for such a system to work, we need a "trusted computing base" as noted by Anderson. Another approach would be to use a centralized database that stores all patient data. This

approach would produce ORCS protected records and appends ORCS protected entries based on database queries.

We cannot assume all users will not modify their operating system environments. Therefore, we must create a secure runtime environment that resides on top of the existing systems (operating system, file system, networking subsystem, etc.). This would also require a certification protocol to ensure that a particular resource is trusted.

Let us consider a Use Case diagram that exposes many of the principle uses of the patient medical record system.

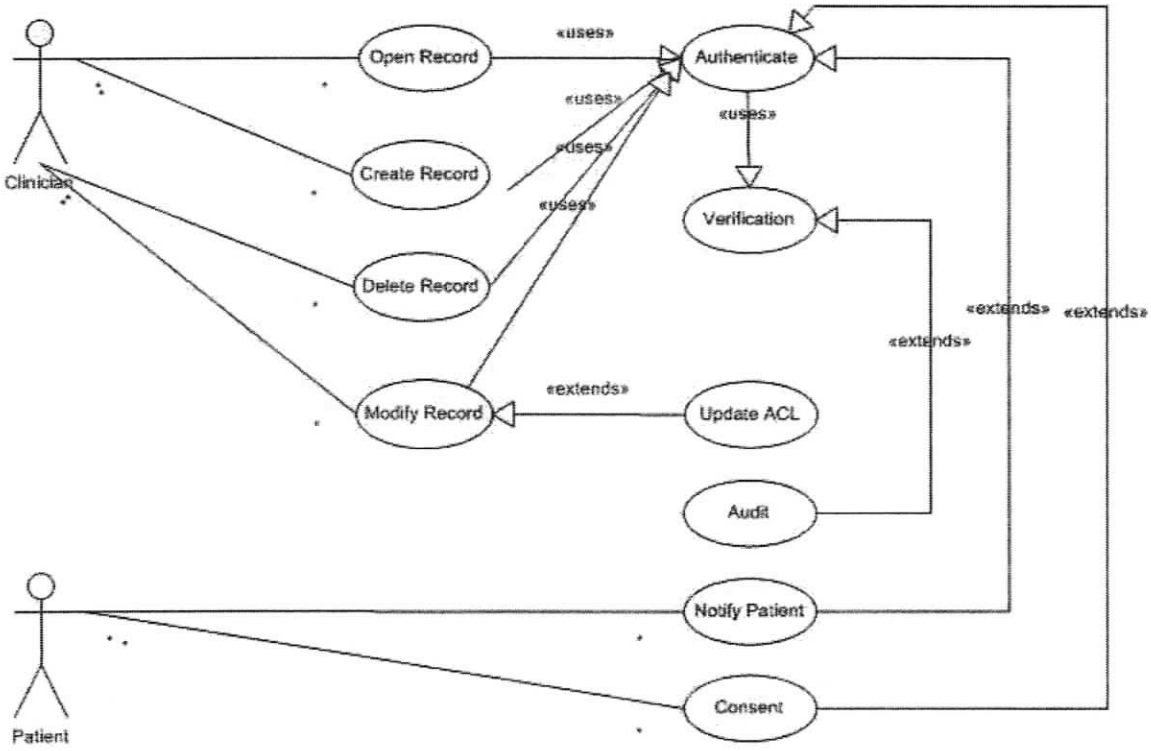


Figure 26: Actions and mechanisms of patient medical record access.

When a system participant requests any patient data, she must make her requests via a security module that enforces the security policy. All user operations are monitored, validated, and performed via the security module.

With the above use case (Figure 26) in consideration, we need to question whether we want to have records stored centrally in a database with an authentication/validation mechanism for data access, or have the data protected at the

record level. In fact, we could feasibly create a system that uses both of these features. There are three primary components needed for this to become a reality:

1. A secure data format.
2. A database system that interfaces with an authentication/verification system.
3. A secure computing base that enforces the security policy.

All clinician requests for data will be processed by the authentication/verification system. If authenticated, the database will produce a record (file) that conforms to the secure data format. This file will be encrypted in such a way that no data inference or data loss can occur (with any realistic probability). Finally, the secure computing base will enforce the security policy when attempting to access the produced record.

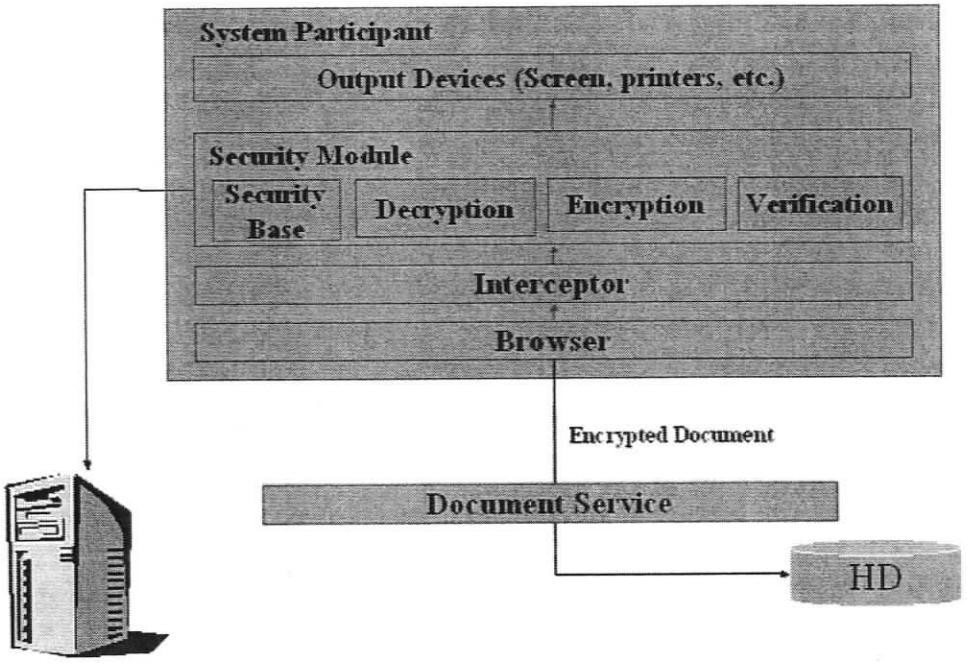


Figure 27: Potential architecture

Since this architecture resides on top of an existing system, several attacks may still be possible. For example, an attacker obtains "root" access to a machine where he then installs a form of "Trojan Horse" that captures the contents of physical memory over time. To add further protection, we consider using a hospital based Certificate Authority (CA) to produce a public key based certificate authentication system for data access.

By using our owner-retained access control model, we would be able to support the strengths of the BMA while addressing most of its weaknesses. Figure 27 displays a potential architecture for the BMA policy for addressing concerns related to patient record protection. The document service would retrieve the patient data from storage device; this could be any document retrieval system. Then, by implementing our ORCS model, we would ensure that the data is stored in encrypted format and that access control attributes would be enforced. To enable this, the security module would be represented by a trusted ORCS application, similar to our implementation above, and would be responsible for all record access, encryption, decryption, verification, and the enforcement of the security policy. There are at least two possible scenario choices for the data owner.

(1) The record owner could be an independent, such as agents representing a government health authority (HA), which issues the initial (empty) medical record to the doctor; the doctor stores the record on his server and has it circulated (in protected format) when the patient visits another physician. Other physicians will be allowed only to read designated sections and append (not write) their comments/prescriptions to the record, and then return it to the main doctor. Updates to the ACL will be done by the health authority, on behalf of the doctor after obtaining consent from the patient. Recall that patient confirmation is required for such changes to the ACL. The patient may be notified by sending an ORCS protected form by email asking him to accept or reject the request and then return the modified form back to the HA. The patient will only be allowed to read the record; in this case, he can request a (protected) copy from his doctor's clinic.

(2) The record owner could also be the hospital, which may create an administrative role for such purpose. This approach is appropriate in the case of a centralized database used to store all patient records at the hospital; since there is no network support for this version of ORCS, records may be circulated in case of visits as above. All medical records would be stored in the centralized database with the ORCS rules applied against the database tables. All requests would be against an ORCS authorization engine attached to the database that is responsible for enforcing the rules of both ORCS and BMA. Other doctors would only be allowed to append to entries into the database once authorization was received from the patient.

The advantage of the first approach is that the HA takes responsibility for enforcing the BMA policy rules, rather than the doctor. This keeps the doctors and other clinicians free to deal with health concerns. The first approach also assists in keeping the patient data private. Anderson discusses the privacy issues related to medical data in [6, 8]. Centralized database approaches, such as in our second option, tend to lead to privacy concerns such as data leakage. Many reported cases of data theft, sale of private medical information and privilege misuse have arisen in the media in recent years. The principle problem with the centralized database approach is that the data is collected in one place, creating a target that is very tempting to criminals.

Components (4) and (7) of the nine key components specified in the BMA policy are the only components not currently supported by the ORCS model. Consent and notification would require an extension to our model to allow for notification being sent to the principal owner of the data whenever the ACL of the object is revised. To support such a feature, we would need to expand our shared ownership portion of the model to include shared ownership collaboration. Perhaps adding a concept where ACL entries could be vetoed when changes to ACL entries occur could be researched. Information flow would also require a new mechanism to support the comparison of ACL's to ensure subset compatibility before copying data.

In summary, the ORCS model could be extended to cover all nine requirements of the BMA policy by expanding the shared-ownership portion of the model and adding several new access control attributes to deal with issues such as information flow. Trusted computing bases remain an issue with all security implementations and the testing, certification, and maintenance of these resources is our safeguard against some attacks.

6.2 Chinese Wall Security Policy

Many other security policies also exist in academia and industry. Much of access control that we know and use today is derived from military research; however, other policies have been created to serve the needs of various industrial sectors. One such policy is the *Chinese Wall Security Policy* as introduced by Brewer and Nash, which was created for the financial sector and used by many financial organizations.

The Chinese Wall Policy merges commercial discretion with legally enforceable mandatory controls. It is reportedly as important to the financial world as Bell-LaPadula is to the military, and cannot be correctly implemented by a Bell-LaPadula model [15]. The basis of this policy is that subjects are only allowed access to information that is not in conflict with any other information that they already possess; in other words, a subject cannot access data in a dataset that has been deemed to be in a conflict of interest with information that the subject already possessed [15]. The formal mathematics is covered in [15].

To understand the fundamentals of such a policy, consider how financial analysts cannot breach the confidentiality regulations that have been put in place to prevent problems such as insider trading. The policy enforces rules that would prevent an analyst with knowledge of company A to provide consultation to a competing company B. The policy uses conflict of interest classes to group datasets, and by mandatory ruling all subjects are permitted access to at most one such dataset. The potential of the Chinese Wall Policy with regards to ORCON is that it is theoretically possible for the owner or originator to assign Chinese Wall Policy to their objects, and since both ORCON and the Chinese Wall Policy can be combined with MAC policies, we have a feasible merger.

The access attributes, subjects, and objects in a Chinese Wall Policy are a subset of what can be represented using an owner-retained access control framework. In essence, the Chinese Wall Policy is based on two primary rules: read and write. These two access attributes are within the subset of those that can be covered by an owner-retained control system. The subject, as in Bell-LaPadula, can be expanded not only to include the user/subject, but also programs acting on behalf of the user.

Brewer and Nash implement a matrix structure to keep track of which subjects have accessed what objects, and briefly point out the advantages of having policy related to legal recourse. The matrix structure can be used by ORCS for those objects/subjects that have been specified to use the Chinese Wall Policy and the legal recourse remains to enforce the system rules.

Possible problems exist with this model. One such problem is that it assumes the subjects knowledge base consists only of information stored on the computer and that the

subject has previously accessed. This is not usually the case. Other problems with the Chinese Wall Policy are noted by R. Sandu in [45] and also discussed in [10].

Another difference is that the Chinese Wall Policy is fundamentally different from the BLP model in the sense that BLP can only model the mandatory part of Chinese Wall OR the free choice part, but NOT both.

In summary the Chinese Wall Security Policy restricts access to data based on the history of actions of the user. If the user has accessed data from another company within the same conflict-of-interest classification, access is denied. There are two other primary qualifiers in the Chinese Wall Policy:

1) Write access is granted only if the "simple security rule" permits it AND no object can be read which is in a different company dataset to the one whom write access is requested *and* contains unsanitized information.
2) Sanitized information is that information for which company information cannot be inferred from.

To implement the Chinese Wall Policy with ORCS, we would need to extend the model to include the access matrix to enforce the additional rules of the policy.

Chapter 7 Concluding Remarks

Access control is required to meet the guidelines specified in a given security policy. It is used in conjunction with other services to ensure secrecy, integrity, and availability. The distinction between MAC and DAC is clear but these two policies do not solve many practical needs. RBAC provides an attractive alternative to historical access control and is currently in the standardization process through NIST [46]. None of these policies, however, address the issue of ORCON and owner-retained access control. A significant amount of new research is being done in the area of access control and DRM, as can be seen in [51, 47].

The ORCS prototype allows for the creation, modification and saving of documents in a protected form, secured using encryption techniques. Once a document has been created and saved using the ORCS application environment, the data is never stored in unencrypted form on any storage medium (other than main memory when the document is being accessed legitimately). This document may then be transferred to another computer, where a contributor may attempt access to the data. XML derived technologies allow for a form of data that can be marked up with security information to package data with its security policy.

This approach has shown that a clean separation of access control mechanisms and access control policy can be achieved while maintaining owner-retained control. The work of Abrams et al. has shown that it is possible to implement additional security policies by extending an existing Trusted Computing Base (TCB) [1].

7.1 Summary and Contributions

This thesis has proposed both a theoretical policy model and enforcement model. The models work towards addressing the issue of owner-retained access control. We also presented a prototype application to test the model, which was used as the basis for conducting a performance and security analysis of the framework. The contributions of the thesis have included the following:

1) Definition of a formal security model for the concept of owner-retained access control in collaborative environments.
2) The implementation of an application named ORCS for the enforcement of owner-retained access control.
3) Evidence that owner-retained control can realistically be applied to data using existing technologies in such a manner that CPU performance is not a major issue.

7.2 Future Work

With this prototype, the focus has been on creating a proof of concept for the overall tool interface and design architecture. Through this process several areas for improvement have arisen. In regard to conforming to industry standards, the XML side of the application has two key areas that will need to be examined. First, the World Wide Web Consortium (W3C) has produced a specification for specifying encrypted content in XML documents. This specification provides a means to represent cipher text in an XML document in an industry standard manner. This specification would be worth inspecting for consideration along with the XACML specification (an extension of SAML for ACL) at OASIS.

In order for ORCS to be considered for production use, further research needs to be done for the following:

1) Standardization of a security policy interchange (grammar) format for the exchange of security policies/restrictions between heterogeneous environments.

2) Certification procedures for trusted computing bases and applications that implement (1).
3) Determination of whether the features of all access control policies can be generalized under an owner-retained control hierarchy.
4) Expansion of the model and implementation to also cover live collaborative environments. Features would include client/server modules, an expansion of shared-ownership to deal with contention issues (for example, when two owners disagree with what access a particular subject should be granted), and a set of secure communication protocols for ORCS message passing.
5) Changes to hardware drivers so that screen captures, and other display issues related to the potential theft of information, can be restricted.
6) Changes to networking subsystems so that network sniffing can be restricted.
7) Changes to the I/O subsystems so that all access to data flows can be restricted.

In future revisions to the tool, we will expand support for key storage to handle multiple users in a dynamically changing user base, messaging for communication with the owner, and integration of new functionality for public-key certificate distribution. Sun has now added support for AES to the JCE built into the SDK. We will also consider changing this implementation, thereby removing the dependency on the commercial product IAIK.

Bibliography

[1] M.D. Abrams, J.E. Heaney, O. King, L.J. LaPadula, M.B. Lazear, and I.M. Olson, "Generalized Framework for Access Control: Towards Prototyping the ORGCON Policy," in Proceedings of the 14th National Computer Security Conference, Washington, DC, 1-4 October 1991.

[2] M. Abrams and P. Schneck, "Controlling Primary And Secondary Access To Digital Information," 23rd National Information Systems Security Conference, 2000.

[3] M.D. Abrams, L.J. LaPadula, K.W. Eggers, and I.M. Olson, "A generalized framework for access control: An informal description," in Proceedings of the 13th National Computer Security Conference, October 1990, pp 135-143.

[4] Adobe Systems Inc., "Protecting Electronic Documents with Adobe Security Solutions," (White Paper), http://www.adobe.com/security/pdfs/acrobat_security_wp.pdf, 2003.

[5] R.J. Anderson, "A Security Policy for Clinical Information Systems", IEEE Symposium on Security and Privacy, 1996.

[6] R.J. Anderson, "Patient Confidentiality – At Risk from NHS Wide Networking," to appear in Proceedings of Healthcare 96, March 96.

[7] R.J. Anderson, "Security in Clinical Information Systems," published by the British Medical Association, January 1996.

[8] R.J. Anderson, "NHS wide networking and patient confidentiality," British Medical Journal v. 310 no. 6996 (1 July 1996), pp 5-6.

[9] R.J. Anderson, "Clinical system security: interim guidelines", British Medical Journal v. 312 no. 7023 (13 Jan. 1996), pp 109-111.

[10] V. Atluri, S. Chun, and P. Mazzoleni. "A Chinese Wall Security Model for Decentralized Workflow Systems," in Proc. of 8th ACM Conference on Computer and Communications Security (CCS-8), Philadelphia USA, November 2001.

[11] D.E. Bell and L.J. LaPadula, "Secure Computer Systems: A Mathematical Model," The MITRE Corp., ESD-TR-73-278, Vol. II, HQ Electronic Systems Division, Hanscom AFB, Ma., November 1973.

[12] D. Bell and L. Lapadula, "Secure computer systems : Unified exposition and MULTICS interpretation," technical report ESD-TR-75-306, MITRE Corporation, 1975.

[13] M. Bishop, Computer Security: Art and Science, Addison-Wesley, 2002.

[14] C. Boyd: "Some Applications of Multiple Key Ciphers," EUROCRYPT, 1988, pp. 455-467.

[15] D.F.C Brewer and M.J. Nash, "The Chinese Wall security policy," In Proceedings of the 1989 IEEE Symposium on Security and Privacy, May 1989, pp. 206-214.

[16] M. Burrows , M. Abadi , and R. Needham, "A logic of authentication," Proceedings of the Twelfth ACM Symposium on Operating Systems Principles, November 1989, pp. 1-13.

[17] M. Burrows, M. Abadi, and R.M. Needham, "A logic of authentication," Report 39, Digital Equipment Corporation Systems Research Center, Palo Alto, Calif., Feb. 1989.

[18] J. Camp, "Access Denied," IEEE Security and Privacy, vol. 01, no. 5, 2003, pp. 82-85.

[19] B. Cox, Superdistribution: Objects as Property on the Electronic Frontier, Addison Wesley, 1996.

[20] P.J. Denning, "Fault Tolerant Operating Systems," ACM Computing Surveys (CSUR), Volume 8, Issue 4, December 1976.

[21] Dept. of Defense Standard, "Department of Defense Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, GPO 1986-623-963, 643 0, Dec. 26, 1985.

[22] J.S. Erickson, "Fair use, DRM, and trusted computing," Communications of the ACM, v.46 n.4, April 2003.

[23] D. Ferraiolo and R. Kuhn, "Role-Based Access Controls," [2005, Sept 12], Available at HTTP://csrc.nist.gov/rbac/ferraiolo-kuhn-92.pdf.

[24] D.F. Ferraiolo, R. Sandhu, S. Gavrila, R. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control," ACM Transactions on Information and Systems Security (TISSEC), Volume 4, Number 3, August 2001.

[25] R. Graubart, "On the Need for a Third Form of Access Control," in Proceedings of the 12th National Computer Security Conference, Baltimore, MD, Oct. 1989, pp. 296-304.

[26] M.A. Harrison , W.L. Ruzzo , and J.D. Ullman, "Protection in operating systems," Communications of the ACM, v.19 n.8, Aug. 1976, pp. 461-471.

[27] R. Hauser, "Does Licensing Require New Access Control Techniques?" in Proceedings of the First ACM Conference on Computer and Communications Security, Fairfax, VA, 1993, pp. 1-8.

[28] P.A. Karger, "The Lattice Security Model In A Public Computing Network," ACM CSC-ER, Proc. 1987 National Conference, December 1978.

[29] B. Lampson, "Protection," in Proc. 5th Princeton Conf. on Information Sciences and Systems, Princeton, 1971. Reprinted in ACM Operating Systems Rev. 8, 1 (Jan. 1974), pp. 18-24.

[30] M. Lesk, "The Good, the Bad, and the Ugly: What Might Change if We Had Good DRM," IEEE Security and Privacy, vol. 01, no. 3, 2003, pp. 63-66.

[31] P.A. Loscocco, S.D. Smalley, P.A. Muckelbauer, R.C. Taylor, S.J. Turner, and J.F. Farrell, "The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments," in Proc. of the 21st National Information Systems Security Conference, Oct. 1998, pp. 303-314, Available at HTTP://csrc.nist.gov/nissc/1998/proceedings/paperF1.pdf

[32] G. Lyon, "The Internet Marketplace and Digital Rights Management," National Institute for Standards and Technology, June 2001, Available at HTTP://www.itl.nist.gov/div895/docs/GLyonDRMWhitepaper.pdf

[33] C.J. McCollum, J.R. Messing, and L. Notargiacomo, "Beyond the Pale of MAC and DAC - Defining New Forms of Access Control," in Proceedings of the IEEE Symposium on Security and Privacy, 1990, pp. 190-200.

[34] A. Menezes, P van Oorschot, and S Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[35] A.P. Moore, R.J. Ellison, and R.C. Linger, "Attack Modeling for Information Security and Survivability", Technical note CMU/SEI-2001-TN-001, Available at HTTP://www.cert.org/archive/pdf/01tn001.pdf

[36] National Computer Security Center (NCSC), "A Guide to Understanding Discretionary Access Control in Trusted Systems," NCSC-TG-003, 30 September 1987, Available at HTTP://www.fas.org/irp/nsa/rainbow/tg003.htm

[37] R.M. Needham and M.D. Schroeder, "Using encryption for authentication in large networks of computers," Communications of the ACM, v.21 n.12, p.993-999, December 1978.

[38] A. Ott, "The Rule Sett Based Access Control (RSBAC) Linux Kernel Security Extension," 8th International Linux Kongress, Enschede, Netherlands, Linux-Kongress, November 2001.

[39] J. Park and R. Sandhu, "Originator Control in Usage Control," in Proceedings of the 3rd International Workshop on Policies for Distributed Systems and Networks (POLICY'02), Monterey, CA, USA, 5-7 June 2002.

[40] V. Rosset, C.V. Filippin, and O.M. Westphall, "A DRM Architecture to Distribute and Protect Digital Contents Using Digital Licenses," aict-sapir-elete, Advanced Industrial Conference on Telecommunications/Service Assurance with Partial and Intermittent Resources Conference/E-Learning on Telecommunications Workshop (AICT/SAPIR/ELETE'05), 2005, pp. 422-427.

[41] R.S. Sandhu, "The Typed Access Matrix Model," In Proc. of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, 4-6 May 1992, pp. 122-136.

[42] R. Sandhu and G. Suri, "Implementation considerations of the typed access matrix model in a distributed environment," in Proc. of the 15th NIST-NCSC National Computer Security Conference, 1992, pp. 221--235.

[43] R.S. Sandhu and P. Samarati, "Access Control: Principles and Practice," IEEE Communications, Volume 32, Number 9, September 1994.

[44] R.S. Sandhu, "Lattice-Based Access Control Models," IEEE Computer, Volume 26, Number 11 (Cover Article), November 1993.

[45] R.S. Sandhu, "A Lattice Interpretation of the Chinese Wall Policy," in Proc. Of 15th NIST-NCSC National Computer Security Conference, Baltimore USA, October 1992.

[46] R.S. Sandhu, "Future Directions in Role-Based Access Control Models (Keynote Lecture)," MMM-ACNS, 2001.

[47] R.S. Sandhu and J. Park, "Usage Control: A Vision for Next Generation Access Control," MMM-ACNS, 2003.

[48] R.S. Sandhu, "Engineering Authority and Trust in Cyberspace: The OM-AM and RBAC Way", ACM RBAC, 2000.

[49] R. Thomas and R.S. Sandhu, "Towards a Multi-dimensional Characterization of Dissemination Control," POLICY, 2004.

[50] M.M. Swift, P. Brundrett, C. Van Dyke, P. Garg, A. Hopkins, S. Chan, M. Goertzel, and G. Jensenworth, "Improving the granularity of access control in Windows NT," in Proceedings of the 6th ACM Symposium On Access Control Models and Technologies, May 2001.

[51] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu, "A Logical Specification for Usage Control," SACMAT, 2004.

[52] D. Icove, K. Seger, and W. VonStorch, Computer Crime. A Crimefighter's Handbook, O'Reilly & Associates, Inc, Sebastopol CA, 1995.

Appendix A: Protected Document

<?xml version="1.0" encoding="UTF-8"?>
<rsadoc:orcs_envelope xmlns:rsadoc="http://www.islandnet.com/~ahoole"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.islandnet.com/~ahoole orcsrsadocument.xsd ">
<rsadoc:user_list>
<rsadoc:user>
<rsadoc:user_name>john_doe</rsadoc:user_name>
<rsadoc:user_key>4KOMeKyA7nOBcj9TVNPFaP3ajOFaBmGzlOgXkPyEtQvZzfW1Q1JjMwMF2Yg2e
VJB
t8twgDPIwOwAJtAuUpxq7pYuL0Y+QyAQApjECSWf53UA6S3HDI5WJsNdIflOFRWa
KoJry5N9ZOTJra0+wyUAu/ZFddJxdvqD/aPjXj5oQJQ=</rsadoc:user_key>
<rsadoc:user_iv>Jel6t3sa5fhOf7vZd3r2HXjKL9wISCPHkxybVVgAqToG5BtYvZOiaH/4Iz1e+I7M
+jhy+WxXiExYBv13ShLUy64rhKWFHco+DS4jt05h/Px/pwFYFNKr85FPYsujwODB
gLgEJLKwxXwcZ+tn0kOLmNyAhPBP6gZMZmV3XRtFdSg=</rsadoc:user_iv>
</rsadoc:user>
<rsadoc:user>
<rsadoc:user_name>jane_doe</rsadoc:user_name>
<rsadoc:user_key>0ZA1KLU1Rd3sA/YUz6mYjNG7TB0vIu5qPzO3YTJq97I5n8N3cL5Vl+cYYush6tQb

Jsz+j5hN4ZOVa44eSaRYry/5oK2yPAQUBWxiwbyLm/CXn0Sa2b/K5UCVwP/E+PLh
2w2Wq32beac8eXhKtOR1+gY+WtOLRBKn4rH4zZFidE4=</rsadoc:user_key>
<rsadoc:user_iv>qlnlcvUv/YgMsxp0Z3fwoJskl1kzVdCaBOXu1C98TB/BkO/hd8E5rb/c1VoI0HqI
fuUfPmEpjhRdUBb5/SH+NaCADyjbIklD+UYk1NM+FXlSuuLlEojP7NFrS8Ltboi5
JV8Kek7oX6EM4fuEZO7B+qdhTwmkocFeu6GYZ1Tlr0Y=</rsadoc:user_iv>
</rsadoc:user>
<rsadoc:user>
<rsadoc:user_name>jim_doe</rsadoc:user_name>
<rsadoc:user_key>RmUa3laNn5IO+uMtA+2dz5oNTVMfFGXTDp4WSyLdKK4GPpsrKexxMG/0XyBsTt
yR
2jxpAJMGu+FLEc5FQymySpGMzwwMAh4pzCBGfqpBHKgo4rWV6ac01cZqZXQ7mLco
2gb//NnBDh+xO3jci3znjiPiw65Fkc8nreFRoaVWCXo=</rsadoc:user_key>
<rsadoc:user_iv>m+Q1DYWYgaSlt7cjF9S75k3MO8GmkcUWuSucZqmEUAonrVy0UpvRZw92KzDfk7
wG
tfvSnNF+zTNpIs+0meo+ATgT6qTEJfu/HRUghs4KVGgLlqEDKgDf8dRIcGzudono
j/jGXjhhjmiMJjJffFvWwhOUXNZiCAruuNFV+awaqKw=</rsadoc:user_iv>
</rsadoc:user>
</rsadoc:user_list>
<rsadoc:orcs_document>NTfom0sRe0mJGvJYsOPM+wwi2GuTCckcgA1/EZUnYxRK3L1z+TkvzZHtIb4
67Kav
UgE+nmQ9iCXVTOmjmXhbSgedIqXHqLC+m/E50rfkbYZ5b/sek+7eYlhOQeP9xcfI
n65dWhqCSxni/2zGzfLolRET0IZS81YFKYTTR4NPpUY8RjbG0ZXbiKkrJ9/hONNo
ecmSqAOKFczQVtIEq+STsDvT/A70DE4wNCaTDqENSG3w/S6gQe8SkSTxni5jxz6h
hKroHZRYzrmPSE8OVuUBRtjMCd2/X0nHG2ss1/bmGCtdMtCwsG6AE/XuaqsFSuv5

+QyFtjMhTxej+/HwFjhotQ3p065YPcQov7T4f6IuMxrlpU63D7680Jf+uu1IUS6R
+++kyfa3dy/IytR4KEbkoj/ax0jFP/BTitG+mVegfY6PynWEgfGWsNiQq/gCJ7d0
G3FiB53XSTixNYBDk5f9LXvjSMz6/3snxIuxrG7tAfp0LlrAyypZjnbPmCjsaC60
S0V0IPdvZo7p1+CgrXCOxXTg5eMnTXwmEzx6M0/EMlrYV2o9bxxzKuGolgXN4sp4
HtCbkDCXyFUDv9RYLdWKrOWeL7z7GMbC3ZqiMK6Z5FztEB+q6YEt5HpJJUm7n770
EtzdO1SOHj2JsCe3OL/q5LvOyfSvWUzJ0DQMJVfmGxht5Z+EwVO6L04iI1lMpYkT
e/EqkFvlkE3j10zh+pLRrV8PVheiIGfgqZh9Sh4idM34FuSx1iCAXOhIkAYtswXu
guP3kUdHWccAtt/BR3HcFRs5xI5HG+einpGplRYTCUA3a8xcNUt1Ka/VzjbS3kFC
aiE4krO/d1Qz8JVJ1MHMnBouoZBi6wvUK7CLuIXxsl/ohIgtw+yrTq972kSNbXES
pv+yJ2dysAqzCwoyeW+wyTg5Eisy7w0s/0iycqkRSvUS/g7CXJF6J76l+Xv9YNDB
NOX12cuEKZX1DfCQvj6ucMUFt2/jUaO3+Hgz6SIfLRAVg5vEpVUpsxDMIAql+upP
vWfADQdrXzeqTP59ZS74lSweX0OkSO6FA8kfX963kIHb3b3oEVwp7h41PNqolsEa
YnFV86kGxCL7KGatk6gSo+FUAVvrF/N8oKz4PRdKYl52+ThmRFxeJi3UFhC7Yrc6
olZx8OhMb2LlZqb2Zgr2qPUPGlGpFU/kVkmFWbbaFmcAkTJUbYPrp9gUPTPDPYhj
j3I39ole+tO4LbqkFhNRa7k9z40evr72aFsPyOgWqL939VbhTU8iV/QT37VRu5hj
lygUjMJN6U2Fdbsh1L7BGT9GDJO6DLm/Q9GiT51GSlDCfN1YvysL/n2yvrfnz34A
cRhCi2yBu89aNAaTSpwWMcq9pHkR4TAMWD5uf01QuouZUBGtqcOQsKstvoXlf5pe
/oM5iOOTsTXuDY+zoslhC4fG1P8KNEtBOxka18sCTupRfUpY5HQdnIU5RH6G4DzG
d7G4qoLsPxCg79BbP7BUvjAQ4tG0mOBu6Ovfa+xcxHnfh6HanTgPFnwHr7F/bUDW
4mfO6MTUkfCuwmWz7j9ascXKNBYoINhyEZvH7sznX1BrKPQhbTtJmaOdUbNQIXuz
OZxTjBL60UNjv3cdLwLIvDzHx0IsGN8TUFzyJEn0b2rAwcjtKNg8s4R9/jKrCq2f
BEn52G3b4WW/rPW8zpncQL7J7wUGjMNc3lDUdpMEyf5/MHw5ek3gTCfh5/T0UPCe
WpDVpGjmCGjP3UmcaBDC+3RS1gkdy2xES30FPn6838A71vZ5xjxkwCrxIgyfmp/S
dphEWNZiZGkD6YKLt168QfQu1ph9MAtwq2X4YWZvLNXYLRUOu/mlkSeFxH+A5HLF
TyeJOpLkCwaleSBofbEuz/8HiPNOZU6nEQn7tb8x/fL4rHU1zM59JlFZAipDMy8E
mA0u/3MbGybMehBs7/RdC8pQzQ6+/J2Br174KJkSV+4zHPU0wlkqBmxZ5DtfsYQ+
JO+HxUO7/o7mptAWKUgoV+qsgZFYmZlXSNUmvrkZyCXit62REeRGMoOzKjE6AU1y
R+ZTsvUaiG/C4GL8BzfskyxcJgtHlhcSVCvIA6R8oQeAYI3J7mQb50XVi2N68aPS
DFfoJgQyhujaZgLAC1j0DoZMXbC9TOyaiJjIbO3Y5hQeBQxFqbT4yF7OmxchWAA9
bPo7vQmSgJQR36UMH5geh9q+6t27jT5w4bdxr2YkpML+n1IHRseV0L5bBwpP9o54
gsSVs8Hod6ygH+/3MmBDLu1lNb4Op8xkBF4WunmrjWOIPqvPycyMs7S9slgQjjCY
oePVz80lvJSR4KybNtBhA0CWZVu91yIaVBoCh2f6qkzg1dVKRCD+d1y8Fops3hUC
MQEHgNKR4ieVuPrQ880Ls39frcDz9Pkz0snFTpTD5+v6G2X+egW5bXqbz6ayRxkz
44YFWUTOajcTWea2BicnjjtJ/0255osx4XIKzErr0GDdElbKAYaBtxQ34SZwHWS4
1aa0SM/BBowPZMaoyX8fDsIJXEds2JfC+ePZFz65z39ZXG1Dg7+Mi8/TyTHqUZeH
QsI6vxHtNERXpi9hynb0A0CqReYjWIkVo3qmct27xf+55l1eqxzv1B08/6LW1Khn
8ZhalkwZBDr8MVlhD2pZtGycdDh9xttu9QvbMfa4SDFfx/03adKTaPGc3ErxVhVx
gTB7EUcJ+HltabfBi/zUK1YirQWSp/XFMfCdmSWQ5q3p5AN3Vz+B0IaJPiWifVmO
u3f0SMz8xQYDqTXym5r9JU6/By8U2tx2Mv7OeU0IuMzhkASLqCzj4mFu6fxjPgQB
z2GPPL65/cuurVWg2RF+BmQU1KBJ7qK3t6XupWvsbMhctJR7ulVcBiFrvvP7keF9
+nMXeQmeealxqfvEU3QzEsu7rb28yuJCu80LxQL+bQzZgF3izGOvd0EY3ysjCGyJ
us0yMgfx6gCB15azZIMI6tqR+E+G0W3c28R4PqSVwJhzQcOiBNkJkoDXExT/OPZ8
ejkVfyVkiHGSep4jkW21eS7vxZLTjr4Ae0mAyYEnjtKeWAcaCa9eJTTL9M1PE/Lo
tZcGd0hsr9gG401KDT4q+6OKDgM4FYkIjhMZNb4XilAGHfy2cCbdPqZMGFt0Kp57
b6UIatl1OXYchuha/FvkL+FbzW7GKpT9ecCk3rGjOGqzPshFgZNo7duEx5Bt6Xq/
/5/8mAFIhknggI2u09JNRfRq8vapkCruqHDSaKa+ZCdEJT/7RiSrE/WaPxMe/t1j
g4UtQuitS2msAWbRp2ExRAAfSl0UOTh2ZF7IvDTZ3ns7PZWrtiZTLQPNh32ZSRxZ
b5aIzD/Chmx5JEuSBsUFh4rRmbYBFWnkYRyPQHDMV4z1vKsiAiyR/WoKUYGBeGRj
FxMVaqdCcaCXy1smxkhVdj3Btn8lx4AVistcA/66Zldtrg43AHzrRaATE9u6kCwU
Hn8c+X17S1jQ2sepEZKeyLnbb1UNUPQMP8vXcjfih554tLY4T6X2rEF0Pvql22+2
ZTyxPkHnHX8anZ4iidSwwachAAiFB1hWU1zFvdQiilzDjMR3W29oS8iTSXg3HKZb
95tKr9chRBxfiIgYuMgn7UdLE9IlvWgo6pd+POUYMzSeaZrmdg5SkjU/GBdTLHAX
a42CQdjlGL4ZYoeYJuONoFkMJ4V1QkQKFAbKFI83mFZIAr6QeP2Yk95jdj+Nin3k
0rznbx4JawnfnaCc5GIPD041hmIQnmc8ZsUwNuhI6L3AUzHhgnVBJ7n9hobvdJTv
v9iOGxtzeyjxm9D6uZJU0jcbqQ+QK0tLJR0gQxN1+YF550L/3fhhoR/4XGh9xUwr

jhnJZr6wlGuGIdqQjsLFJNoW4cSFsz0LrczfADYmu1M7eF2DuJ5Fxuvopp9QXADg
Mpg/agMDqeLur2hHZSuFKBqkTmd3tYTRhZW5BHgurf8TThj3cZ/28f17Dt165W9Y
gPGTagt4gfIOeRaJOea5NpKFX9/qfNk+oFJupw4qdc6qYyQ2il9dKOXHtN18fnEs
qfJMZ2UOpDGrnQZMJhaLr5Et7fY0poookNy2JA5pbHOxz0VJ30ZgdXxofkVvfv4t
mWGJpPlAcrZHs1MsEzxmv6fA2GHFGczSKm5B5acZHw9wl6mEbSMnEetiPnQPMgQi
gYjuy3usTgz1CM26nONkRMwpWieUD7pSJ3paIk7RJjx9NXj7IL8PtSMqNUkHcRBw
IgB26ydu1qtzd5jq1IaUax4cPDENlGGuA9vXntyZDNG6IUXFwbfWhTwv+5bE9EfV
VZG0EdWv/UrX7/Jbw9oM5tmQePqqdSVfQ4SO/0EsJcetJX8bQ2EJtM1ARwOOWCoo
gkkNRGJJz5Ycu+yyspg2od2LzSqiS//b0aSoEx4zBBgi8rn3OpmG+2GOCr9MnV9Q
f8p+qqcLNlEhet0rYg4+qUi+pH32DScNoRW8gqwW99vle18FJw9kBo2cSSV6MSPd
XPo10cXHWG10ViBMwQUavz3LlhOUVi/EbYD5mOPDIzuiumtncLcl4fiCnbFwKdeF
NjW7c8I1ZepgZ8H/BuopxCz6RTy5KVku88RDq5V30e5PQ99PeopHXxQ+7gHqG4O0
QDJmw1fFPwsLCvaM4pqNZ74/YxuVuWannm8AD/VYV/65sHyzJDjxZu2f5mhdhhV6
X3egZKmephfjNNqNYMTRgQL5cd0OfnhVH34qAMt7KAXVzMjNbrx6rMS361WlUPHn
czNpU7Seqyc3kNmlMiLT0QWdbNupAGcKVse38lY8d4o=</rsadoc:orcs_document>
</rsadoc:orcs_envelope

Appendix B: BMA Background Information

B.1 Hospital Environments

To better understand the system, let us first consider the variables involved in the enterprise. These variables should include: physical layout of buildings, network architecture, actors' involved, potential threats, resources to be protected, and a security policy. We will first introduce potential physical and networked architectures for a theoretical hospital; other variables will be discussed later (an example hospital layout may be found at: http://www.mgh.harvard.edu/directions/FloorPlans.jpg).

In such a hospital configuration we can expect some level of physical security at building access points. Hospitals are essentially a public service, and as such, are accessed by everyone. This raises security concerns that we will address later in this design document.

For a networked PMRPS to be designed and protected we need to consider a theoretical network diagram. Many locations at the hospital will have networked computing resources; such a configuration may resemble
Figure 28.

Within a hospital environment we can expect that valid users will be working with hospital records (including patient data) at the following locations:

- Hospital labs
- Hospital offices
- Hospital nursing stations
- Hospital information stations
- Home offices
- Off-site labs
- Off-site offices
- Mobile devices
- Remotely through a Web interface
- Anywhere (in printed form)

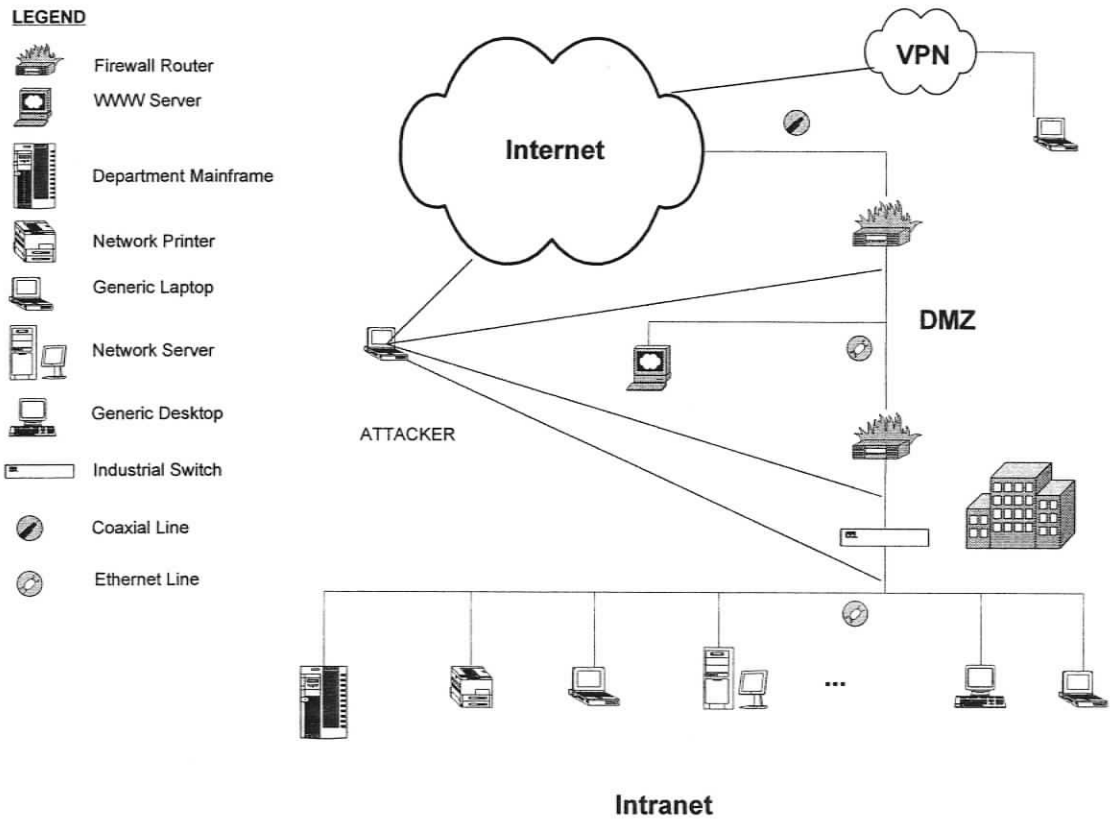


Figure 28: Computing resources of a hypothetical hospital.

Knowing this, a potential attacker could attempt to gain access at any point. In the remainder of this appendix we will explore threats and vulnerabilities.

B.2 Threats and Vulnerabilities to Patients' Records

In this section we will explore and identify the threats and vulnerabilities that characterize the system. Through corresponding attack trees and intrusion scenarios, we will be able to visualize the threats and vulnerabilities.

In order to understand the threats and vulnerabilities we need to first explore the different components that exist in the hospital environment.

Actors: {Doctors, Nurses, Lab Technicians, Cleaning Staff, Management, Visitors, Patients, Contractors, Students, IT Administrators, IT Support, Gift Shop Employees, Food Services Employees, Information Desk Employee, Volunteers, Security Staff, Orderlies, Ambulance Paramedics, Visiting Professionals}

Security Services:	{Confidentiality, Integrity, Availability}
Security Threats:	{Disclosure, Deception, Disruption, Usurpation}
Specific Threats:	{Snooping, Masquerading/Spoofing, Non-Authorized Access, Repudiation of Origin, Information Leakage, Deduction of Information, Data Modification/Destruction, Denial of Receipt, Delay, Denial of Service, Social Engineering}
Protected Resources:	{Network, Systems, Users, Programs, Files/Data}

Now, let us consider the High-Level Attack Tree for our Patient Medical Records Protection Scheme (PMRPS). The tree was devised by instantiating relevant attack patterns suggested by Moore et al. [35]. As recommended by Moore et al. [35], we will depict the attack tree in a textual fashion since the size of the attack tree is large. The root node of the attack tree for our PMRPS is the disclosure/compromise of patient information. Recall that the principle goal of access control is the protection of information.

System Compromise: Disclosure of Patient Information

OR 1. Physically scavenge discarded items from Hospital
- **OR** 1. Inspect dumpster content on-site
- 2. Inspect refuse after removal from site

2. Monitor emanations from machines
- **AND** 1. Survey physical perimeter to determine optimal monitoring position
- 2. Acquire necessary monitoring equipment
- 3. Setup monitoring site
- 4. Monitor emanations from site

3. Recruit help of trusted clinician insider
- **OR** 1. Plant spy as trusted insider
- 2. Use existing trusted insider

4. Physically access hospital networks or machines
- **OR** 1. Get physical, on-site access to Intranet
  - **OR** 1. Social engineering
  - 2. Become an employee
- 2. Get physical access to external machines
  - **OR** 1. Social engineering
  - 2. Attack external machine using its network connections
    - **AND** (refer to 5.3, replacing Web server with external machine)

5. Attack hospital intranet using its connections with Internet
- **OR** 1. Monitor communications over Internet for leakage
  - **AND** 1. Obtain connection shared with hospital
  - 2. Install/execute/record network communication using scanner
- 2. Get trusted process to send sensitive information to attacker over Internet
  - **OR** 1. Replace trusted process with modified version
    - **AND** 1. Located trusted process that attacker has access to
    - 2. Supplant trusted process with modified (maliciously) version
  - 2. Use trusted process (unaltered) to send sensitive information
- 3. Gain privileged access to Web server
  - **AND** 1. Identify hospital domain name
  - 2. Identify hospital firewall IP address
    - **OR** 1. Interrogate hospital Domain Name Server
    - 2. Scan for firewall identification

3. Trace route through firewall to Web server
3. Determine hospital firewall access control
OR 1. Search for default listening ports
2. Scan ports broadly for any listening ports
3. Scan ports stealthily for listening ports
OR 1. Randomize target of scan
2. Randomize source of scan
3. Scan without touching target host
4. Identify hospital Web server operating system and type
OR 1. Scan operating system services' banners for OS signature
2. Probe TCP/IP stack for OS characteristic information
5. Exploit hospital Web server vulnerabilities
OR 1. Access sensitive shared Intranet resources directly
2. Access sensitive data from privileged account on Web server
AND 1. Get access to privileged account on hospital Web server
OR 1. Exploit buffer overflow vulnerability to access privileged account
AND 1. Identify executable program on Web server susceptible to buffer overflow vulnerability
2. Identify code that would provide access to privileged account when executed with program's privilege
3. Construct input value that will force code to be in program's address space
4. Execute program in a way that makes it jump to address at which code resides
2. Exploit unexpected operator vulnerability to access privileged account
AND 1. Identify executable program on Web server susceptible to unexpected operator vulnerability
2. Identify (unexpected) operator that permits composing system calls
3. Identify system call that would provide access to privileged account when executed with program's privilege
4. Construct unexpected input by composing legal input value with system call using the unexpected operator
5. Execute program on Web server with unexpected input
2. Scan files for sensitive data
6. Attack hospital intranet using its connections with public telephone network (PTN)/VPN
OR 1. Monitor communications over PTN for leakage of sensitive information
2. Gain privileged access to machines on intranet connected via Internet
AND (Refer to 5.3)

If we preferred to view the attack tree as a tree diagram, we could visualize it as seen in Figure 29.

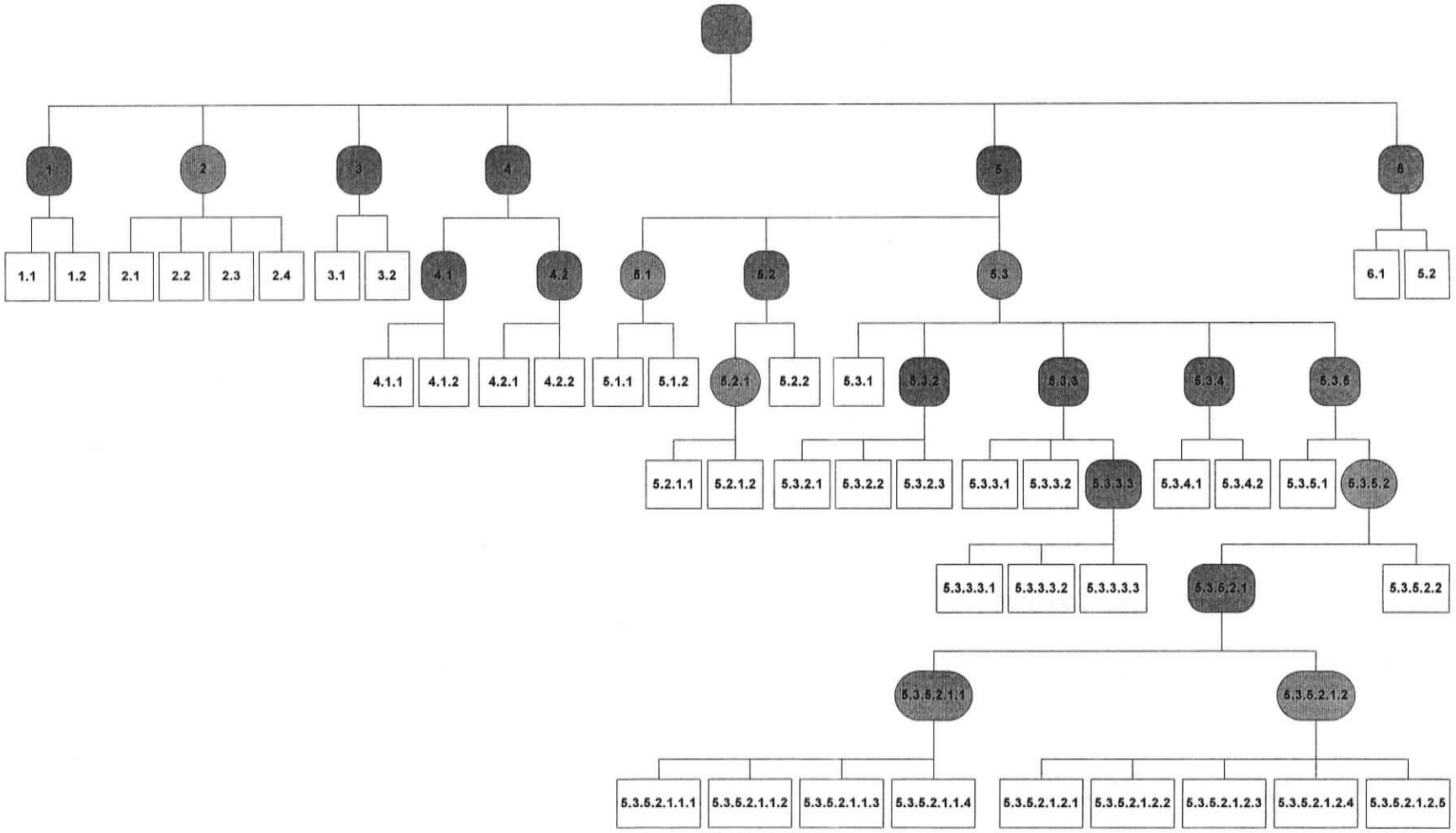


Figure 29: Attack tree for the Patient Medical Records Protection Scheme

With the attack tree described above it would be wise to consider the potential attackers for this system. We listed above the many different actors involved in the hospital environment; we will use the UML actor notation to group these individuals into their specific categories (see Figure 30). While the list of actors is not meant to be extensive or complete, it exposes a major threat to the system; an attacker of the system could be any person!

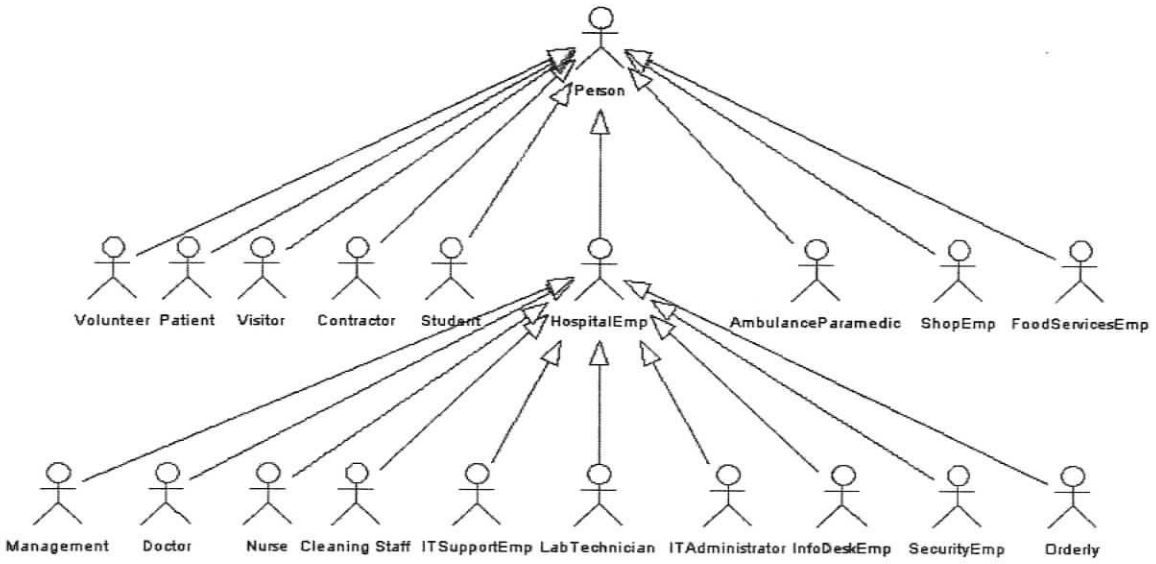


Figure 30: Hierarchy of actors/subjects within a hospital.

With the actors in mind (see Figure 30), we can consider the different intrusion scenarios that have been exposed by the attack tree. They are as follows (* indicates reference to other attack pattern):

<1.1>, <1.2>,

<2.1, 2.2, 2.3, 2.4>,

<3.1>, <3.2>,

<4.1.1>, <4.1.2>, <4.2.1>, <4.2.2.*>,

<5.1.1, 5.1.2>, <5.2.1.1, 5.2.1.2>, <5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.1, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.1, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.2, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.2, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.2, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.2, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.2, 5.3.5.1>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.2, 5.3.5.1>,

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.1.1, 5.3.5.2.1.1.2, 5.3.5.2.1.1.3, 5.3.5.2.1.1.4, 5.3.5.2.2>, --

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.1, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.1, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.2, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<5.3.1, 5.3.2.1, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.2, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,
<5.3.1, 5.3.2.3, 5.3.3.3.3, 5.3.4.2, 5.3.5.2.1.2.1, 5.3.5.2.1.2.2, 5.3.5.2.1.2.3, 5.3.5.2.1.2.4, 5.3.5.2.1.2.5, 5.3.5.2.2>,

<6.1>, <6.2.*>

An example of such an intrusion scenario is <2.1, 2.2, 2.3, 2.4>, which means that an attacker was able to obtain patient medical information by monitoring emanations from machines. The steps involved in performing such attacks are as follows: survey physical perimeter (2.1), acquire monitoring equipment (2.2), setup monitoring site (3), and finally, monitor emanations from site.

The BMA security policy does nothing to address several of the threats noted above. For example, those threats indicated by <1.1>, <1.2>, <2.1, 2.2, 2.3, 2.4>, <3.1>, <3.2>, <4.1.1> or <4.1.2> (see attack tree). Anderson, however, does address some of these issues with regards to the devices underlying the systems and the physical security in [9].

Anderson notes that the main threat to most modern-day systems comes from insiders [6]. As such, the definition of secure computing base comes into question. He points out that through the corruption of insiders, third parties may obtain private information.

While Anderson does address aggregation of information, the distribution of data is not fully addressed. When a clinician prints a medical record there does not seem to be a module outlined to address the accountability of the circulation of this form of the data; however, this could be dealt with at the implementation level (for example, in the print server functionality). The NHS model [6], only classifies data at four levels of sensitivity: Highly Sensitive, Sensitive, Non-Sensitive, and Open. Anderson deals with this issue and its inherent weakness in dealing with confidentiality. Since Anderson proposes that the system be exposed to the WWW [6], the integrity of the protection mechanism must be "bullet proof". Again, the definition of secure computing base must be thoroughly questioned. The distribution of data issue could be addressed by ORCS by requiring that all ACL information be printed on all pages of documents printed in the system. Revisions to the hospital security policy to ensure all printed documents are produced in a physically secure location with a security personal enforcing the ACL markings on retrieval would also help.

Currently, relative to Anderson's publication, hospitals operate with relatively little security [8]. Once aggregation of the distributed medical data occurs the risks to patients increased tremendously. He presents several convincing examples of where people with access to data, still have the ability to abuse their assigned "power". There are also multiple reports in the media of patient information being leaked from hospitals all over the world. To handle this situation, any policy put into effect needs to have appropriate legislation to ensure that those people who do step outside of the stated rules

of the policy can be legally prosecuted. This in effect will lessen the temptation for people to benefit financially and otherwise through the distribution of private information.

Anderson holds the authenticated individual responsible for all access to data through their ID's access rights. This could imply that all access to the system under a particular ID is attributed to that user. However, if the definition of "secure computing base" is flawed or there are inconsistencies between the policy and the implementation, perhaps at the network or operating system level, then we are holding the wrong person responsible. With potential legal ramifications this could set a dangerous precedent. Thus, any legislation must be explicit as to how they determine whom responsibility for patient information leakage.