

Enterprise Digital Rights Management: Solutions against Information Theft by Insiders

Yang Yu Tzi-cker Chiueh

Experimental Computer Systems Lab
Computer Science Department
Stony Brook University
{ yyu, chiueh }@cs.sunysb.edu

ABSTRACT

Insider attack is one of the most serious cybersecurity threats to corporate America. Among all insider threats, information theft is considered the most damaging in terms of potential financial loss. Moreover, it is also especially difficult to detect and prevent, because in many cases the attacker has the proper authority to access the stolen information. *Enterprise Digital Rights Management* (E-DRM) protects sensitive information by managing and enforcing access and usage rights to the information throughout its lifecycle, no matter where the information is distributed. However, the self-protection strength of the DRM client software has always been a potential weakness for all DRM solutions, and application-specific implementation also restricts the deployment of many E-DRM systems. In this report, we review the general DRM architecture and several commercial systems, and describe the design, implementation and evaluation of an industrial-strength system called *Display-Only File Server* (DOFS), which can transparently and effectively stop information theft by insiders in most cases, even if the insiders have proper authorities to read/write the protected information. The DOFS architecture ensures that bits of a sensitive file never leave a protected server after the file is checked in and users can still interact with the protected file in the same way as if it is stored locally. Essentially, DOFS decouples “display access” from other types of accesses to a protected file, and provides users only the “display image” rather than bits of the file. Therefore, DOFS can have less dependency on the trusted client software against information theft by insiders.

1. INFORMATION THEFT

Nowadays most organizations such as government agencies, financial institutions and professional companies have been storing and processing their confidential information in digital format in their daily life. The confidential information might include product overviews, marketing plans, customer lists and sales reports in the format of Microsoft Office, Adobe pdf, HTML, etc. Organizations normally process the information by sharing those digital files from protected file servers and distributing them by downloads or email messages. Compared with the traditional print format, the digital format can significantly improve the efficiency of handling the confidential information as well as maintaining its reliability. However, the digital format also makes the

information more vulnerable to be exposed to unauthorized parties. For example, a company employee may accidentally send a sensitive document describing a new product's design to his friend, who deliberately publishes it on the Web for fun. A disgruntled employee may also save the company's secret market plans with a CD or USB drive and sell it to a competitor company.

Unlike the external attacker who tries to infiltrate into an organization through the network, people inside an organization, such as a company employee, have more opportunities to interact with the sensitive digital information, and thus are more likely to have the information disclosed to non-trusted parties, either unintentionally or deliberately. This kind of threat, called *insider attack*, cannot be stopped effectively by traditional methods like firewalls and intrusion detection systems and is receiving more and more attention from system administrators and security researchers. Insider attack is more dangerous than external attack because the attackers are already inside the security perimeter and therefore are considered trustworthier than those outside. In addition, they have more knowledge about the internal security mechanism and sometimes even have the authority to bypass the associated security checks without raising alerts.

Among all insider attacks, information theft is considered the most damaging in terms of potential financial loss. Examples of information theft are downloading sensitive files into personal removable media, copy-and-paste of confidential file content, screen capture of protected document image, etc. It is reported that in 1999, Fortune 1,000 companies sustained losses of more than \$45 billion from theft of their proprietary information. And according to the 2003 CSI/FBI Computer Crime and Security Survey [5], which polled 530 computer security practitioners, theft of proprietary information was the single largest category of losses in the 2003 survey totaling \$70.1 million or 35% of the total financial loss reported in that survey, and disgruntled employees were blamed for 77% of these attacks. In a time when proprietary corporate information represents crucial technical or business advantages, carefully safeguarding such information is essential to stay competitive.

A number of regulations and laws have been published since 1996 to address the information security issue, such as *California's SB 1386*, *Sarbanes-Oxley Act*, *the Gramm-Leach-Bliley Act (GLBA)* and *the Health Insurance Portability and Accountability Act (HIPAA)*. These emerging legislative standards are forcing enterprises to take specific steps to protect their digital information. In order to comply with these standards, enterprises need to safeguard their digital information throughout its entire lifecycle, and to prevent unauthorized access even after the protected data has been distributed to end users. On the other hand, it is equally important that the protection mechanism for such information should not disrupt the normal business processes.

Traditional methods for securing networks such as firewalls and intrusion detection systems are deployed at the boundary between an enterprise's internal network and its Internet connection. These perimeter-based solutions are unable to help protect digital information after it has been accessed or delivered to an authorized individual within the internal network. Standard access control mechanisms are not sufficient either, because in many cases the attackers are authorized users and there is simply no access control violation when the confidential information is stolen. Other possible solutions include: (1) restricting usage of removable media on end user's computer, which may bring inconvenience to end user's working environment and increase the workload of system configuration/maintenance; (2) content filtering to monitor email messages, which may not effectively detect sensitive data if encrypted or obfuscated by the attacker.

A well-known model that may provide high assurance against information leakage is Multilevel Security (MLS) systems, in which data and users are assigned to one of many security levels, and the data can only be accessed by a user whose security level is at least as high as the data's classification. More generally, MLS adds security labels to files, processes and other

computer resources so that information cannot flow directly from higher to lower security levels. Although there have been some emerging MLS-based operating systems and database management systems, they are largely ignored in the commercial enterprises because they are very expensive and difficult to build and deploy. Moreover, most commodity applications are not designed to assign or interpret security labels [22] and thus may have to be rewritten or at least modified to run on the MLS operating systems. Other annoying problems in MLS include the “blind write-up”, hidden data within downgraded information [23], and etc.

A more comprehensive and practical solution is Enterprise Digital Rights Management (E-DRM), which is a category of information protection techniques that aim to manage rights to digital intellectual property and help organizations protect sensitive information from unauthorized use. E-DRM solutions provide information owners the capability to specify fine-grained rights ---- such as view, copy and edit ---- with specific files that need to be protected and to enforce these rights at the time when the files are accessed. For example, the information owner can specify whether the file content can be copied, whether copy-and-paste is allowed, or even how long a particular user can view the file content. Once the rights are specified, they can travel with the protected files together and stay effective until the information owner or privileged users change them. Compared with the traditional perimeter-based solutions or access control mechanisms, the E-DRM protections are obviously stronger and more flexible for enterprise digital information.

The rest of the report is organized as follows: Section 2 gives an overview of digital rights management, including the rights model, DRM system architecture and two rights expression standards. Section 3 introduces three commercial E-DRM systems from Microsoft, Liquid Machines and Authentica, and evaluates their advantages and limitations. In Section 4, we will describe a new E-DRM variation named Display-Only File Server (DOFS), which takes a radical approach to the information theft problem caused by insiders: preventing insiders from touching any content bits and at the same time still maintaining the rights control of the protected information. Section 5 summaries the DOFS system and outlines the future work.

2. DIGITAL RIGHTS MANAGEMENT OVERVIEW

Digital rights management is an industry category that represents the protection of various forms of digital content, such as electronic books, on-line digital music, financial reports, product specifications, and etc. When the content owners create the digital content that needs to be protected, they should specify a set of rights to the content and become the content’s rights holder. When other users intend to make use of the digital content, they have to obtain adequate rights to the content through some kind of licenses or tickets issued by the rights holders. The business and technological process for controlling and managing rights to digital intellectual property is thus called *Digital Rights Management (DRM)*. Originated from operating system’s file protection mechanism, DRM is essentially a fine-grained access and usage control in the application level. Encryption and watermarking are widely used in this field to encrypt content, authenticate users and track content usage. The current DRM techniques mainly have two types of applications: (1) Systems for distributing content to consumers in a controlled way against piracy; (2) Systems for managing access to sensitive document content within an enterprise. The second application is often called *Enterprise Digital Rights Management (E-DRM)*. E-DRM plays an extremely important role in fighting against information theft, especially the theft due to insider threat. In this section, we will introduce the rights model and a typical DRM system architecture that are the fundamentals of any DRM systems.

2.1 Rights Model

A rights model is a specification of content rights that the system can keep track of and what the system can do to or with those rights [3]. It describes types of rights (what users can do with protected digital content) and attributes of those rights (for how long, by which user, etc). A properly defined rights model is the first step to build a DRM system for an enterprise: It should be able to determine what kind of rights a particular user can have on a specific sensitive document, and how long the rights can be valid before expiration. In general there are four types of content rights: *render rights*, *transport rights*, *derivative work rights* and *utility rights* [17].

Render rights are the rights to render the digital content through output devices, including: (1) *View* (Display the content on a computer screen); (2) *Print* (Render through printers as permanent hardcopies); (3) *Play* or *Execute* (Render the content in sequence from beginning to end).

Transport rights are the rights to copy or move the digital content from one place to another, including: (1) *Copy* (Both users can have access after the first user copies the content to the second user); (2) *Move* (The first user gives up access after moving the content to the second user); (3) *Loan* (The first user gives up access temporarily after loaning the content to the second user, and obtain the access back later).

Derivative work rights are the rights to manipulate the digital content to create additional works, including: (1) *Edit* (Modify the content); (2) *Embed* (Take part of the content and use it in other content, like copy-and-paste); (3) *Extract* (Use pieces of content on their own).

Utility rights are the rights to manipulate the digital content to maintain the data integrity and improve the system performance, including: (1) *Backup* (Make a copy of the digital content for restoring the original content if compromised); (2) *Caching* (Make additional copies of the digital content to improve system performance); (3) *Data integrity* (Create error-correcting codes or checksums to ensure that data is not corrupted).

Associated with each type of rights are the rights attributes: *extent*, *types of users* and *consideration*. Rights extent represents how long, how many times or in what places the rights can apply. Types of users indicate that the rights holder can specify different sets of rights and rights attributes for various classes of users. And consideration refers to what the user has to give in return for the rights. Each right the rights holder confers on the digital content can have its own set of attributes attached to it.

In enterprise digital rights management, the rights-protected content refers to any sensitive documents like product overviews and market plans. Therefore, the typical rights that E-DRM system will consider could be a subset of all the listed rights in the above rights model, including view and print in the render rights, copy and move in the transport rights, the derivative work rights and the utility rights. Based on the rights model, the information author can define a precise rights specification for one or a set of sensitive documents. In recent years there have been a number of standard rights languages that can be used to express rights for enterprise applications. These standard languages provide flexible interoperability in rights expression and interpretation across systems and platforms. We will talk about two rights expression standards: eXtensible rights Markup Language (XrML) and Open Digital Rights Language (ODRL) in Section 2.3.

2.2 A Typical DRM System Architecture

Most DRM systems can be considered as variations of a typical DRM system architecture, as shown in Figure 1. It conceptually consists of three major components: *the content server*, *the license server* and *the DRM client*.

2.2.1 Content Server

The basic functionality of the content server is to store the protected content files in a *content repository*, which is essentially a file server or a database system. The content in the repository can be already in the proper format for distribution, but in most cases is converted into the proper format on demand in response to users' request.

Another functionality of the content server is to prepare a content package when users access a particular piece of sensitive content. The preparation process includes encrypting and packaging the sensitive content and related metadata, and creating the rights specifications for the content. A standalone module called *DRM packager* is responsible for completing these tasks. The resulted content package will contain encrypted content metadata, which can be a unique identification number for recognizing the content and tracking its usage. It may also contain the encrypted sensitive content by symmetric-key algorithms, or sometimes simply contain a link to the content. The content link is often used for streaming media formats, and we present a similar idea in our Display-Only File Server framework to enforce radical controls over the sensitive content in that even authorized inside users cannot have sensitive bits in hand.

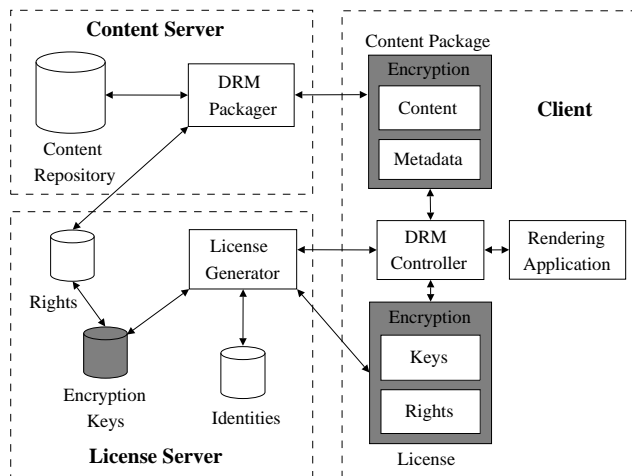


Figure 1. A typical DRM system architecture

2.2.2 License Server

Licenses contain information about the rights specifications, identification of the content to which the rights apply, and the identity of the user or device that wants to exercise rights to content. A user of a DRM system needs to get a license before she can access a particular piece of content. The licenses are generated by a *license generator* on the license server.

The DRM packager creates rights specifications and a set of encryption keys that are used to authenticate users and decrypt content. The rights specifications and keys are then stored in separate databases on the license server. Both of the two databases contain content identifications so rights specifications and keys can be associated with the specific content items. In addition, the

license server also stores user identities, which are information about users who exercise rights to any protected content. The user identification can be a user name, a user's biometric information or a digital certificate. From the users' license request and information in the rights, keys and identities database, the license generator issues proper licenses to legal users and send them back to DRM client.

2.2.3 DRM Client

The DRM client consists of a *DRM controller* that receives the user's request and then communicates with the license server, and the rendering applications that decode and render the content in supported format. The DRM controller can be a standalone software program, or a simple module residing within a rendering application. The main responsibility of the DRM controller is: (1) receiving the user's request to exercise rights on a content package; (2) collecting the user's identity information and applying for a license from the license server; (3) retrieving encryption keys from the license and decrypting the content for the rendering applications.

There are two types of rendering applications. The first type is the applications that are built for a particular DRM system and for a set of specific content formats. These applications normally integrate the DRM controller together and support only a limited number of customized content formats. In general they may not be feasible in an enterprise environment because most users are unwilling to give up their existing application programs they are very familiar with, and in many cases it is rather difficult to develop new applications that can have the same functionality as the existing applications programs from original vendors. The second type is the general-purpose applications that the DRM system modifies to restrict their behavior. The modifications can be achieved by developing and loading plug-in modules, if the application vendors have provided plug-in interfaces, or by intercepting underlying system calls or API calls issued by original applications. The modified applications do not change the existing working environment, but they can have an additional step for checking rights before each important user operation (Open, Print, Save, Edit...) is really executed.

2.2.4 Content rendering

A DRM user obtains a content package from the content server by standard file system commands, email attachments or rendering requests. Later the user can make a request to exercise rights on the content by choosing a menu item in a rendering application, executing file system command or double-clicking a file icon against the content package. The DRM client is designed in such a way that the DRM controller can always be activated in response to such a request.

Once it is activated, the DRM controller starts to collect necessary information to apply for a license, including the user's identity information, the content identifier in the content package, and the rights the user needs to exercise. If the user has not registered with the DRM system before, the DRM controller will help to create the identity information and send it to the identity database on the license server.

The collected information from the client-side is then sent to the license server for a license. The license generator on the license server authenticates the user's identity in the identities database and uses the content identifier to look up rights specification for the content. For a legal user request, the license generator creates a license containing rights specification, client identity information and the encryption key, and then sends the license back to the DRM controller. The encryption key is used to decrypt the content and is normally encrypted again in the license by the

user's public key. Finally the DRM controller on the client side decrypts the content and releases it to the rendering application, whose operations to the protected content can be restricted by the rights specification in the license.

2.3 Rights Expression Standards

The fundamental issue for a DRM system is how to express the rights specification of protected digital content. For a specific DRM system, the content rights can be expressed in arbitrary syntax, and its interpretation totally depends on the system developers. However, it is true that a piece of protected content can be distributed to other DRM systems, whose users may expect that the rights management for the guest content can work there as well. Therefore, different DRM systems need a standard way to express and interpret the rights specification for realizing interoperability, and a common language that can be shared among all the participants in the digital workflow is required.

In order to support a wide variety of business models, the standard rights expression language must be:

- Comprehensive, providing a framework to express rights at different stages of a work flow or life cycle.
- Generic, defining a large body of format and business neutral terms that you use to specify rights for any digital content or service.
- Precise, using a grammar and processing rules to ensure unique interpretation of the language.
- Extensible, allowing any third party to define elements to meet specific business needs [7].

2.3.1 *eXtensible rights Markup Language (XrML)*

Extensible rights markup language (XrML) is currently the only rights expression language used in working DRM systems, including Windows Rights Management Services from Microsoft. XrML was originated from Xerox Palo Alto Research Center (PARC)'s Digital Property Rights Language (DPRL) and its version 2.0 has been accepted as the Motion Picture Experts Group Rights Expression Language (MPEG REL).

XrML is a type of specification language that allows programmers to specify high-level rights structures in detail without having to understand the implementation of the structures. Therefore, an XrML-enabled DRM system needs an interpreter for XrML to parse the license file and translate content rights to code that actually enforces the rights. The XrML interpreter can be implemented individually by different DRM systems, but the better way is to follow existing high level APIs such as MPEG REL SDK to reduce the risk that a rights expression may be interpreted differently by different systems.

The data model in XrML 2.0 includes the concepts of license, grant, principal, right, resource and condition. The key top-level construct is a license, which contains a set of grants that convey to certain principals certain rights to certain resources under certain conditions [11]. A principal encapsulates the identification of principals to whom rights are granted. It can be represented by a *keyHolder*, someone identified as possessing a secret key such as the private key of a public/private key pair. A right is the “verb” that a principal can be granted to exercise against resources, such as play, edit, print, copy, delete and etc, and is similar to what we have described about the

rights model in Section 2.1. A license file for a specific piece of protected content is composed from these concepts to describe the rights specification.

2.3.2 Open Digital Rights Language (ODRL)

Open digital rights language (ODRL) is another rights expression language that has been proposed for a standard and its version 1.1 has been accepted by the Open Mobile Alliance (OMA) as the rights expression language for mobile content.

ODRL's data model consists of three core entities: *Assets*, *Rights* and *Parties*. Assets refer to any physical or digital content that can be uniquely identified. Rights include *Permissions* that can then contain *Constraints*, *Requirements* and *Conditions*. And Parties include both rights holders and the end users. In fact there are many concepts in ODRL that have direct equivalents in XrML, and the difference between the two languages is that ODRL might be more applicable to actual transactions in the media and publishing world. For example, it can explicitly specify resolutions or encoding rates for protected content. Although ODRL 1.1 can provide more compact and elegant rights-modeling languages than XrML, its development is not as successful so far as XrML in the application of practical DRM systems and services.

3. ENTERPRISE DRM SYSTEMS

In this section, we will introduce three commercial E-DRM systems from Microsoft, Liquid machines and Authentica. All of the three systems have particular aspects that are different from the traditional DRM architecture, and all of them have gained relatively dominant positions in the E-DRM market against information theft.

In general, any E-DRM system design should consider the following principles:

- Secure content by distributing encrypted files or files' metadata that links to related files on a protected repository;
- Control and audit access to protected content, including view, edit, save (export), print, email, copy-and-paste, screen capture, and even rights modification;
- Introduce minimum changes to enterprise business process and existing user applications;
- Utilize existing account management and authentication mechanisms as much as possible, such as NT domain, Active Directory and digital certificate;
- Secure content rendering and rights enforcement by trusted client software;
- Secure client software by tamper-resistant techniques;
- Watermark sensitive content to trace its distribution process;
- Enable off-line access and dynamic rights update;
- Enable external users like business partners to access rights-protected content;
- Adopt standard rights expression languages (XrML and ODRL) to enable interoperability among different DRM systems;
- Secure the license server or policy server against attack or system failure.

3.1 Microsoft Windows Rights Management Services

Microsoft Windows Rights Management Services (RMS) works with RMS-enabled applications like Microsoft Office 2003 to help safeguard enterprise digital information from unauthorized use, both online and offline, inside and outside of the firewall [19]. Compared with the DRM system architecture described in Section 2.2, the main difference of Windows RMS is in the following two aspects:

(1) Windows RMS does not have to keep sensitive document files in a content repository. All the sensitive information can be distributed freely once created by trusted authors. There is no separate content server in Windows RMS systems. The content packaging process is done locally.

(2) Rights policies are stored in a publishing license attached with the sensitive document and travel with the document together. The license server does not keep any rights information. This feature allows mobile users to access the rights-protected information off-line.

Windows RMS consists of three components: (1) RMS server software on Windows Server 2003, which is a web service for enrollment and XrML-based certification of trusted entities, and for licensing of rights-protected information; (2) RMS client software, which is a group of APIs that allow RMS-enabled applications to work with the RMS server; (3) RMS-enabled applications or web browsers, which allow users to create rights-protected information and assign usage rights. Windows RMS technique provides a secure and flexible way to create, distribute and consume sensitive digital information. Its logging utility can help track RMS licensing activity and generate a record each time a use license is granted or denied. The RMS SDK also enables developers to extend existing applications and make them RMS-enabled. Moreover, adopting XrML to express usage rights and conditions gives RMS interoperability in processing rights-protected information.

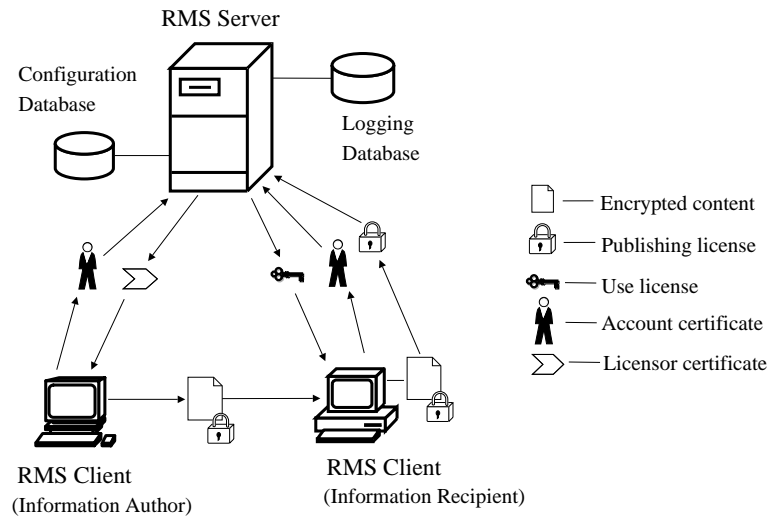


Figure 2. Windows Rights Management Service

The RMS architecture is shown in Figure 2. Before an RMS client computer can publish any protected files, a *publishing license* must be generated and bound to the file. The publishing license, expressed in XrML, contains the usage rights and conditions, as well as a symmetric key that is used to encrypt the file content and is encrypted itself by the public key of the RMS server. The RMS server is responsible for issuing a publishing license, but the client computer can also issue the publishing license when it is not connected to the RMS server. The latter case is called *offline*

publishing. Any computer that can issue a publishing license must obtain a *licensor certificate*, which contains a public key for this computer as a licensor, no matter whether it is the RMS server or client. The RMS server requires a licensor certificate from the RMS Server Enrollment Service, a Web service provided by Microsoft, while the client computer requires the certificate from the RMS server.

When a recipient of the protected files opens it using an RMS-enabled application or browser, she has to apply a *use license*, which includes the symmetric key that can be used to decrypt the file content and is itself encrypted by the public key of the recipient. This process ensures that only trusted entities can obtain the content encryption key from the use license. The use license may also contain relevant conditions like expiration restrictions. Although both the RMS server and client computer can issue the publishing license, only the RMS server can issue the use license. A recipient computer requests a use license by sending the publishing license attached to the file and the account certificate, which is issued by the RMS server the first time when the recipient creates or uses the rights-protected information. The account certificate includes the public key of the recipient and is also used to authenticate the users in the RMS system.

Once the recipient gets the use license, the RMS-enabled application can decrypt the symmetric key in the use license, and then decrypt and render the file content. The use license stays with the protected file so the recipient does not have to get a new use license to access the file content again unless the license expires. The recipient computer can reside on both the internal and external network of an enterprise. Anyone with a valid account certificate and access to the RMS server can obtain a use license, which helps the RMS-enabled applications to decrypt the file content and enforce the usage rights. For an internal user, the account certificate can be issued after validating the user's credentials by Windows authentication, while validation of an external user's credential can be carried out by using Active Directory or Microsoft .Net Passport.

For highly sensitive files, rights policies can also be set to require the users to request a new use license each time they open the files or at regular intervals. The concepts of *revocation lists* and *exclusion policies* are used to revoke certificates of trusted entities and deny requests of use licenses respectively. However, the architecture of RMS determines that rights policies are distributed with protected files together. Therefore, even if RMS can sometimes invalidate an account certificate or stop issuing new licenses, it cannot dynamically revoke or change policies on the protected files. If the rights policies are set in a way that users only need to request a use license the first time they open the file, RMS may lose control of the protected file and its attached policies. This is a drawback of RMS from supporting more user mobility and better offline access.

There are another two concerns with the security of Windows RMS system. The first concern is the security strength of the RMS-enabled applications (rendering applications). If the attackers understand the application code pretty well, they can modify the code and fake the scenario that they have full control on the protected file. The second concern is for authorized insiders, it is possible that they are granted enough rights on the protected document, such as extract content, export content or even change the rights. The result is that once these authorized insiders are intended to steal the bits of the protected content, the sensitive information will be in danger immediately.

3.2 Liquid Machines

Liquid Machines is another E-DRM solution that supports both offline access and rights update. Different from Windows RMS architecture, the rights policies for a protected file are defined on a central repository of the Liquid Machines server. Later these policies are distributed to respective

user desktops that are authenticated on the network through standard NT LAN Manager (NTLM). In this way, users can work on the protected information offline for a number of days that is configurable in the rights policies. The encryption keys can be cached on the client with a validity period and protected by Windows access controls. Meanwhile, the information author or system administrator can also adjust the rights policies on the Liquid Machines server. The Liquid Machines client polls the server for policy updates at a configurable time interval. Any policy modifications are also refreshed on the client each time a user logs into the Windows domain.

Liquid Machines works well in the super-distribution process of any protected information. For example, when a piece of sensitive text is copied from a protected file to a new file, the rights policies are automatically propagated to the new file. Besides it uses its own policy expression instead of XrML to express rights policies.

The most attractive function of Liquid Machines is that it does not require modifying application or specific application plug-ins to enforce the rights policies. The patent-pending *auto-integration* technology and API framework enable the Liquid Machines client to work with most existing applications transparently. As a result users can continue to work on the protected files in their native content format using existing applications. This auto-integration technology allows Windows RMS capabilities to be added into applications that are not initially RMS-enabled. Liquid Machines has started to integrate with Microsoft rights management to RMS-enable leading desktop and enterprise applications such as Acrobat and AutoCAD. In addition, previous Microsoft Office applications without RMS feature can also be RMS-enabled to view and modify RMS-protected documents created in Office 2003 [14].

3.3 Authentica's PageRecall

Authentica's PageRecall stores encryption keys and rights policies on a central server. When an Authentica client application intends to open a rights-protected file, it must connect to the central server to get the keys and rights policies. This feature differs from Windows RMS and Liquid Machines because the rights usually are not distributed with the protected files together.

The main components of PageRecall include: (1) Authentica Policy Server, which is a central server used to store decryption keys, rights policies, audit logs and authentication configurations. (2) Client authoring component, which converts any protected file to PDF format and encrypt it, and registers rights policies with the Authentica Policy Server. (3) Client viewing component, which authenticates to the Policy Server, retrieves the decryption key to decrypt the file content and enforce the rights policies. Both authoring and viewing components connect to the Authentica Policy Server over a SSL channel to authenticate users and exchange keys. The keys are destroyed immediately after decryption to reduce exposure on the client. All activities related to the keys are logged on the Policy Server, which provides a complete audit trail of activity for the lifecycle of the content.

The client authoring component encrypts the protected file content by a symmetric key from the Authentica Policy Server. It also computes a message digest and passes it to the server for verification of the protected content's integrity in future. Once the rights policies are registered on the policy server, a recipient must connect to the server and be authorized to receive the decryption keys. The separation of rights policies with its protected information also ensures that the rights policies can be changed dynamically. Client authentication to the policy servers can be supported by existing LDAP Directories, NT domain or digital certificate.

In addition to its centralized rights management, Authentica also supports a *work-offline* mode, if the rights policy allows the protected files to be accessed offline. In this case, once the user

authenticates to the Policy Server, a *lease* is generated and the rights policies will be distributed to the client computer temporarily. Therefore, users do not have to connect to the Policy Server to access the file until the lease permission expires, and the full audit log of the offline access is sent to the server when the user connects back to the Policy Server.

One problem with Authentica's PageRecall is that its client authoring and viewing components are based on application plug-ins, which works with Adobe Acrobat PDF files only. Although PageRecall can manage fine-grained rights policies using the plug-ins, i.e. different pages can have different rights polices, application plug-ins are not a general solution to various user applications. In many enterprises users may not be willing to, or cannot make conversions to PDF files from their existing content format. Although another product from Authentica, named Secure Office, provides plug-ins with the same functionality to Microsoft Office files, it is still not enough to handle all popular content formats used in enterprises. Also, a third party plug-in may compromise the rights enforcement procedure, but simply removing any other plug-ins may decrease the functionality of the rendering applications.

3.4 Limitations with E-DRM Systems

Most of the existing E-DRM solutions protect enterprise sensitive information by encrypting the files where the sensitive information resides, and enforce the rights policies by trusted DRM client software. The DRM client authenticates with the license server and receives decryption keys for the file through SSL channel. Based on existing research experience on cryptography, we can have the following assumptions for these E-DRM systems:

- The encryption algorithms are strong enough;
- The authentication mechanism between server and client is normally reliable;
- The encryption keys and rights policies can be exchanged safely.

Under these assumptions, the security strength of an E-DRM system is strongly tied with the extent to which the trusted DRM client can protect itself from being tampered within a potentially malicious environment, because in most cases the protected content bits are physically located at the client machine. A very moderate attacker will always find vulnerabilities in large and complex systems [12]. For a motivated attacker who has been authorized to access the rights-protected information on a computer that is completely under control, it is only a matter of time to break the protection mechanism (if there is any) [8], bypass the rights enforcement checking and steal the data. If a privileged user has been granted rights like exporting unencrypted content or modifying rights policies to a protected file, once she is authorized to access the file content, there is simply no way to control the content's distribution. Moreover, various screen capture approaches are still a large threat to confidential information displayed on the screen, and so far there is no complete solution to prevent it.

Another limitation in many E-DRM systems is that most of the deployment procedure of DRM client requires installing application-specific plug-ins or modifying existing applications to support rights enforcement. On the one hand, these DRM systems may only work with specific applications and content formats. As a result, their deployment may lead to change of user interface and degradation of business efficiency. On the other hand, application plug-ins may be vulnerable to be compromised, bypassed or overridden by other plug-ins of the same application.

In the following section, we will present the design and implementation of a new E-DRM system that enforces stronger protection to enterprise sensitive content by isolating the content bits from the end user's desktop. This solution intends to concentrate most of the computations about

file access and right enforcement on a well-protected server instead of on each individual client machine. Although this approach is not feasible to support mobile users that need to access the protected content off-line, it improves to some extent the protection strength of sensitive information accessed by insiders, whose attacks are more common and dangerous than those attacks from the outside.

4. DISPLAY-ONLY FILE SERVER: DISPLAY WITHOUT FILE CONTENT

4.1 Introduction

Based on existing E-DRM systems, *Display-Only File Server (DOFS)* presents a new solution against information theft, especially the information theft by authorized insiders. DOFS stores enterprise sensitive files on a protected server and prevents bits of the files from physically leaving the server. In other words, once a sensitive file is checked into DOFS server, its digital content can never be directly taken out. Meanwhile, users can still read or write these files through standard applications such as PDF reader or MS Word. The only difference is that applications that operate on sensitive files actually run on the DOFS server and display the execution results back to the end user's computer. Because of the isolation of digital content from end users, the rights management in DOFS architecture can be less complicated than most E-DRM systems, i.e. it could be based mainly on the access control policies of the underlying operating system of the DOFS server. Deployment of DOFS is similar to standard network file server, and is largely transparent to users in that it does not change the *look-and-feel* of those applications that are used to open sensitive files. To support off-line access by mobile users, DOFS also provides an explicit checkout mechanism for users to check out a sensitive file, and turns it into a program that contains an encrypted version of the sensitive file and is itself protected with a tamper-resistant mechanism. Compared with existing E-DRM systems, DOFS exerts a radical control over sensitive files such that even authorized insiders cannot have access to the content bits, let alone leaking them. Being able to prevent authorized users from stealing sensitive corporate information is important because it is estimated that more than one third of the insider attacks that cause information theft are done by managers or people who have worked in the victim companies for more than 5 years [5].

The principle of DOFS design is that the protected file server only outputs the “display image” rather than “bits” of the protected digital content. Although the possibility exists that an attacker can break the communication protocol and encryption mechanism used in remote display to construct the “image” data, this kind of attack is presumably more difficult than, or at least as difficult as attacking most E-DRM client software, such as modifying the code and bypassing the rights enforcement instructions. DOFS also cannot completely prevent screen capture. It offers a limited solution, but so far no complete solution exists to this problem due to various types of analog attacks, e.g. camera photographing or screen emanating radiations [18]. Running trusted client software on a hardware simulator also makes most protections against screen capture ineffective. Overall, DOFS significantly reduces the possibility of leaking confidential information intentionally or by mistake, while requiring minimal changes to user behavior or the existing IT infrastructure.

The idea of DOFS comes from the Secure Mobile Code Execution Service system [27], a commercial system that is used to isolate potentially malicious mobile codes from a user's desktop and monitor their runtime behavior. The remote execution and remote display mechanism in DOFS ensures that even if an inside attacker can take full control of an end user's desktop, she still cannot have direct access to the confidential information within the digital content, because the content

never leaves the DOFS server. In the following subsections, we will describe the DOFS system architecture and the implementation details of a Windows-based commercial system based on the DOFS architecture. We also present the results of an evaluation study on the scalability of the DOFS architecture and an analysis of its weaknesses.

4.2 DOFS System Overview

Corporate confidential information is usually stored in one or more file servers in the form of html, Postscript, PDF and Microsoft Office files. Users can access the files by mounting NFS partitions if they are stored on Unix/Linux file servers or by opening the files from network shares if they are stored on Windows file servers. A user usually creates these files using some editing applications, and then checks them into one of the protected file servers. Other users can later invoke their local applications, either to access these files directly over the network, or to access copies of the files that are duplicated from file servers to their own computers. The access permissions of these files can be determined by their authors or the system administrators.

For a local application to access a file's content, the end user needs to own the read permission to the file and the content bits have to physically come to the user's desktop. Once the end user has a copy of the file, it is rather difficult to prevent information theft from happening. For instance, a user can easily attach the sensitive data in an email and send it to unauthorized users. Even if the current E-DRM systems are deployed, the content bits, although protected through an encryption or obfuscation mechanism, could still be exposed once an attacker bypasses the rights checking instructions in the client software. Given past experiences in the effectiveness of anti-piracy solutions for software packages and computer games, the probability that attackers successfully disable rights checks in DRM software is pretty high, especially when the protected content is worth a great deal of money.

The key idea of DOFS is to prevent bits of sensitive files from reaching the end user's computers once they are checked in, without requiring any changes to the way users interact with them. To achieve this, whenever the user invokes an application on a sensitive file, the application runs on the DOFS server, which then displays the result of execution back to the requesting user's computer. Even though the application runs on the server, the user can still interact with it as if it runs on the local computer, including entering inputs using keyboard and mouse. Because the file server not only stores sensitive files but also runs applications that manipulate these files, and because the only information exported by the file server is bitmaps for display on the requesting client desktop, hence the name "*Display-Only File Server*". The remote display mechanism must ensure that even "*copy and paste*" operations cannot move sensitive file bits out of a DOFS server.

The DOFS architecture consists of a DOFS server component and a DOFS client component, which are equivalent to the content server and client DRM controller in Section 2.2's DRM system. Within an enterprise, there could be one or multiple DOFS servers, and each end user's machine could be a DOFS client. A user machine can mount a partition of a DOFS server just like any normal file server's partition. However, once a file is copied into a DOFS server's partition, the bits of this file can never physically leave the DOFS server, and users can still read or write files stored on a DOFS server in the usual way, subject to the standard authentication and access control mechanism of the underlying operating system. The DOFS architecture supports a "*virtual ticket*" mechanism that allows users to include sensitive files as email attachments without actually copying the bits. Essentially, each such email attachment is a virtual ticket. The receiver of an email attachment that contains a virtual ticket can use the ticket to access the associated file on the corresponding DOFS server. The process of accessing a virtual-ticket email attachment is

completely transparent to the user, if the receiver is authorized to access the corresponding DOFS server. The same mechanism is also applied to the case when users try to copy sensitive files to their local machines, and any duplicated copy is a virtual ticket as well.

Because DOFS clients still have access to display bitmaps, it is important to prevent them from being leaked as well. Therefore, the screen dumping capability of a DOFS client machine is disabled when it is accessing sensitive files. In addition, digital watermarks that contain the user identity are embedded into the bitmaps used in the remote display to deter attackers from copying bitmaps through other means, e.g., taking pictures using cameras. When bitmaps of sensitive files are leaked and detected later, these watermarks can be used as evidence for prosecution in the court of law. These solutions, though cannot offer complete protections, greatly reduces the opportunity for the information on screen to be stolen.

To support mobile users that cannot be connected with a DOFS server, DOFS also allows users to explicitly check out a sensitive file. In such cases, the protection of sensitive files is based on similar techniques used by most existing E-DRM systems. The function of checking out a sensitive file can also be applied to external access by business partners, when they cannot have direct access to the DOFS server.

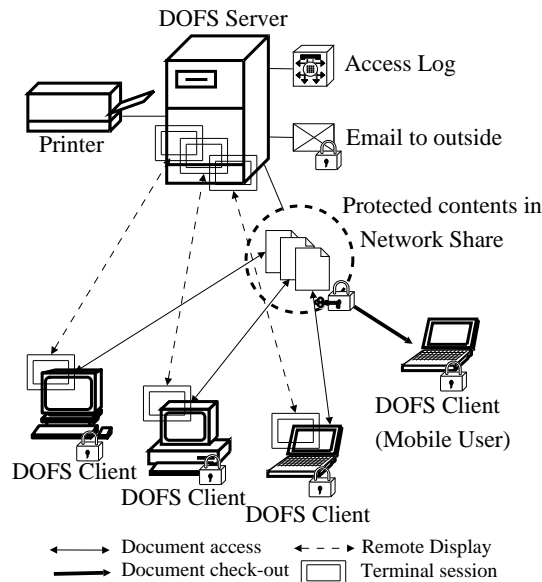


Figure 3. DOFS system architecture

Because all the manipulations of sensitive files are centralized on the DOFS server, critical operations such as checking in, modification, deletion, printing, including as email attachment, checking out, etc. are fully monitored, logged and controlled.

Figure 3 illustrates the system architecture of DOFS, which consists of a DOFS server and four DOFS clients (including a mobile client). The DOFS clients within the enterprise access sensitive files stored in the network share of the DOFS server. These requests are intercepted by the client shell, and redirected to the corresponding application running on the server, whose screen display is sent back to the client over a *terminal session*. There are several alternatives to implement the remote display capability, including Microsoft Windows Terminal Server, Linux server running Wine or Crossover Office and VNC, and Windows server running multi-user VNC or X Windows [27]. The current DOFS server's implementation is based on Windows Terminal Services (WTS),

and works on Microsoft Windows 2000 (advanced) server, whereas the DOFS client works on Windows NT 4.0 and above. WTS adopts RSA RC4 to encrypt the communication data, including the display bitmaps between server and clients. The existence of DOFS is completely transparent to standard file operations, so users cannot feel any difference when accessing sensitive files through Windows Explorer or command shell, regardless of whether these files are from a Windows network share or a mapped network drive.

4.3 System Implementation

The DOFS server is responsible for filtering file access, controlling system service and executing session applications, which are satisfied by three separate components as shown in Figure 4: A file system filter driver, DOFS Service Control and Execution Manager. When a sensitive file is read, only an encrypted version of its path name on the server is returned. This is called the file's "pseudo content", which can be regarded as a type of metadata for the file. The file system filter driver intercepts the I/O requests for reading a sensitive file, encodes the file's path name on the DOFS server, and returns it in the output buffer instead of the file's real content bits. When a DOFS client receives a request for accessing a file that contains an encrypted path name, it starts a Windows Terminal Server (WTS) session on the DOFS server and sends the encrypted file path to the Execution Manager running in each new established terminal session. The Execution Manager is responsible for starting appropriate server applications to open the target file, and the DOFS Service Control is used to communicate with the Execution Manager and the file system filter driver. The three components are described in the following subsections from 4.3.1 to 4.3.3.

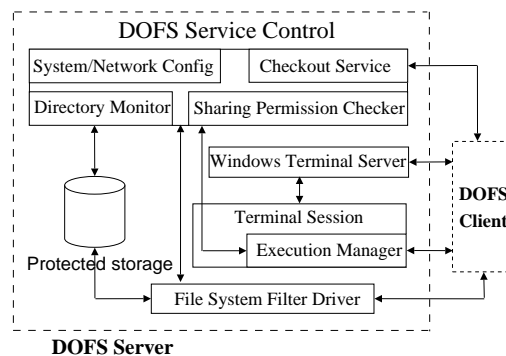


Figure 4. Software Components of DOFS Server

4.3.1 File Access Filtering

File sharing on Microsoft Windows OS is implemented through remote File System Driver (FSD), which contains a redirector FSD executing on a client computer that communicates with a server FSD on the file server, as shown in Figure 5. The redirector FSD intercepts Win32 file I/O requests directed at files residing on the file server, translates them into Common Internet File System (CIFS) protocol commands and transmits them to the server FSD, which listens for commands from a network connection, and translates them back into I/O requests that in turn are issued to local FSD [6].

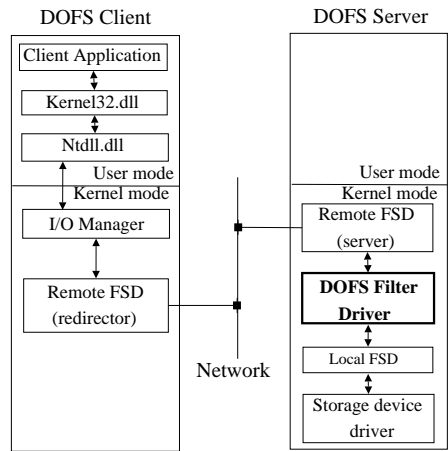


Figure 5. Remote FSD and DOFS Filter Driver

Windows device drivers are implemented as a series of layers, which form the layered device object stack. A file I/O request coming from a user-mode application or a kernel-mode driver will be translated into an I/O Request Packet (IRP), which is first serviced by the topmost device driver. This driver can either complete this IPR directly or pass the IRP to the lower-level driver for further processing. Windows provides well-defined interface for attaching a filter driver above an existing functional driver like a local FSD. Therefore, DOFS can easily attach a filter driver above the local FSD corresponding to the disk volume device where the protected file directory resides. The DOFS filter driver is based on the “FileMon” program built by Mark Russinovich and Bryce Cogswell from Sysinternals [16].

The DOFS filter driver sits immediately below the server FSD, shown in Figure 5. It filters all IRPs coming from the server FSD that read and create files, and passes down IRPs from DOFS server’s local system to the lower driver without any modifications. Hence, the filter driver does not affect applications running directly on the DOFS server.

To prevent a sensitive file’s content from flowing out of the DOFS server, IRPs associated with the major function code of *IRP_MJ_READ* must be intercepted. Instead of passing the IRP down to read the file’s real content, the DOFS filter driver constructs a private buffer containing a unique DOFS label, the server IP and the file’s path name. The buffer’s data is then encrypted using RC4 and copied to the beginning of the IRP’s output buffer. Therefore, the response to an *IRP_MJ_READ* IPR is an encrypted path name padded with a sufficient number of zero bytes such that the total length is the same as the target file’s size. Sometimes the target file size is so small that the IRP’s output buffer is not large enough to hold the encrypted file path name. In such case, the DOFS filter driver also intercepts a *FastIO* routine when clients query the file size information and thus fakes the size as well.

The other IRPs that the DOFS filter driver intercepts are the ones associated with *IRP_MJ_CREATE* code, which are used to potentially overwrite existing files. The filter driver renames the original file if it exists by appending an extension of “.dofs” to the file name, and passes the IPR downwards. This renaming operation is important to prevent a protected file from being accidentally overwritten by its own pseudo content, in order to maintain the data integrity on the DOFS server. We will describe its detail in the next section.

4.3.2 DOFS Service and Access Control

DOFS Service Control is an NT service application that is running in the *LocalSystem* account. It constantly monitors the directories that keep sensitive files for file creation and retrieves the

sharing permissions for a particular file. It also checks the number of active terminal sessions from the same client IP and ensures that a DOFS client cannot hold too many terminal sessions simultaneously. Moreover, it is responsible for loading the DOFS filter driver and communicating with it through I/O control APIs.

Because a DOFS client only has access to the pseudo content of a sensitive file, it is important for the DOFS server to distinguish between pseudo content and real content when a file is written. For example, a user may copy a file A from the DOFS server through Windows Explorer and save it as file B in a local directory. Later he accidentally overwrites A with B, which contains nothing but a pseudo content of A.

DOFS's directory monitor constantly watches the sensitive file partitions and receives notifications whenever a new file is created there. The file name is returned asynchronously by associating the target directory's handle with an I/O completion port. When a new file is added, the directory monitor spawns a worker thread to check the file content for the DOFS label. If the label does not exist, the directory monitor validates the file content and leaves it alone; otherwise, the file must contain the pseudo content of a sensitive file on the DOFS server, and the directory monitor can extract that file's path and restore the real content. In the case when the user attempts to overwrite the real file with its own pseudo content, the DOFS filter driver always renames the original file first by appending a ".dofs" extension to its file name, so the overwriting does not really happen and the file content can still be restored from the renamed backup version.

DOFS's utilizes the discretionary access control mechanism in Windows Access Control Models and NTLM authentication to manage the accesses to the sensitive files. This reduces the implementation complexity of maintaining its own rights policies database for inside users. However, DOFS's application execution model creates one problem: Sharing permissions of sensitive files are never checked when they are accessed by server application processes because these processes are considered local processes on the DOFS server. To resolve this issue, DOFS implements a sharing permissions checker, which receives a file's path name from the Execution Manager, identifies the corresponding network share on the DOFS server, and retrieves the effective access rights for the target file. If the resulting sharing permission is read-only, DOFS makes a copy of the file, and asks the Execution Manager to act on the new copy so that the original file will not be contaminated. Meanwhile, enforcement of local user permissions for the sensitive files stored on the DOFS server is done by WTS.

4.3.3 Session Application Execution

The Execution Manager starts within each terminal session and exchanges messages with DOFS client. This communication is based on a virtual channel, which is a session-oriented transmission protocol [28]. The WTS is configured in a way that the Execution Manager is the only program that can initiate other application processes within a terminal session on the DOFS server, and the terminal session automatically ends after the Execution Manager and all of its child processes terminate. No application processes in a terminal session are allowed to access the remote network shares on other machines.

Upon receiving a request, the Execution Manger decrypts it to retrieve the target file's path name, checks with DOFS Service Control for the file's sharing permissions, and starts a proper application process for this file. It then monitors the sessions' state periodically until there are no running child processes or visible windows. It is also responsible for receiving updates of client clipboard data and then uses a hidden window to set the same data into the server clipboard of the corresponding terminal session. Section 4.3.5 describes the issue of clipboard mapping in detail.

4.3.4 Client-Side Control

The main components of a DOFS client include a shell extension, the Client Service Control and a document handler, as shown in Figure 6. The shell extension is a COM object that implements *IShellExecuteHook* interface to intercept the *ShellExecute* family of Win32 API calls. These API calls are used to activate a proper application instance to open a file from the Windows shell. Therefore, when a user opens a file, DOFS client can always step in and assume control as a DRM controller.

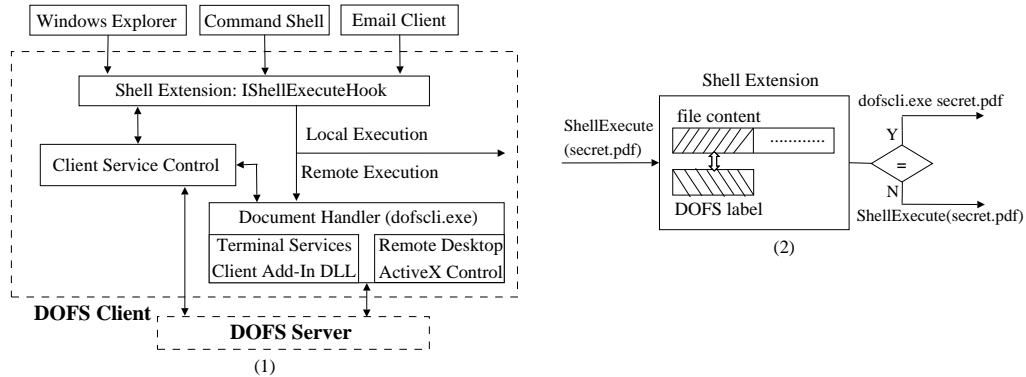


Figure 6. (1) Software Components of DOFS Client; (2) Execution flow of protected files

By decrypting the bytes in the file’s header and comparing the resulting string with a DOFS label, the shell extension can determine whether the file contains pseudo content (metadata) that points to a protected file on the DOFS server. If it does, the DOFS document handler is invoked with the file’s path name as the input parameter. Otherwise, the file is simply opened locally. Once a DOFS document handler is invoked, it decodes the DOFS server’s IP from the file’s content header, obtains the user account from the Client Service Control, and connects to the WTS by a Remote Desktop ActiveX Control. This ActiveX Control is used to establish connections and exchange simple messages with WTS. To exchange file information, a client add-in DLL is registered to provide full functionality of a virtual channel. The DLL obtains the document’s encrypted file path from shared memory and sends it to the Execution Manager.

4.3.5 Clipboard Mapping

The default configuration of WTS supports clipboard mapping between terminal server and terminal client, so users can copy data from a server application and paste it into a client application. Copy and paste is undesirable because it provides a channel through which bits of sensitive files can be leaked. Therefore, DOFS server disables the WTS clipboard mapping and implements its own one-way clipboard data transfer from client to server.

The DOFS client window monitors the changes in client clipboard content by adding itself to the clipboard viewer chain and handling specific window messages. When it receives the message of *WM_DRAWCLIPBOARD*, the client add-in DLL is notified through an event object to read the new clipboard content and to send it to the Execution Manager, which then sets the server clipboard data accordingly.

One drawback of the current implementation is that the clipboard may contain multiple objects in different clipboard formats, and it is difficult to enumerate all the different data structures to copy data from the memory. As a result, currently DOFS only allows users to copy and paste pure text from the client to the server application.

4.3.6 Preventing Screen Capture

Screen capture is the main approach that an attacker can use to steal sensitive file content in the DOFS architecture. It can be done via the *Print Screen* key or through specific screen capture tools. In the first case, the bitmap of the desktop screen or the current active window will be dumped into the clipboard immediately after *Print Screen* is pressed. To prevent users from getting the bitmap on a DOFS client, the DOFS Client Service Control installs a keystroke message filter procedure, which resides in a DLL and intercepts system-wide keyboard event, into a hook chain. Whenever a message identified by the virtual-key code of *VK_SNAPSHOT* is to be processed, the clipboard content is cleared to remove the screen dump bitmap.

To stop screen capture tools is more complicated. Instead of going through clipboard, these tools typically rely on the bit-block transfer functions to read the screen bitmap indirectly from the frame buffer. To intercept and disable the related bit-block transfer functions, either from user-mode GDI (Graphics Device Interface) library or kernel-mode DDI (Device Driver Interface) interface, the following conditions should be satisfied: (1) the rectangle to be copied overlaps with the DOFS client window; (2) the source object refers to the desktop screen, which eventually corresponds to the frame buffer controlled by the video driver. This approach against screen capture is a more general solution than simply controlling the *Print Screen* key, but it also needs to be made tamper-resistant so as to protect itself from being disabled.

Essentially, solutions against information theft by screen captures have to consider two fundamental issues. The first issue is whether it is possible to identify and intercept all the interfaces through which the underlying frame buffer can be accessed. For the user level interfaces, we already know that there are various methods for capturing the screen, including Window GDI, DirectX and Windows Media API [13], but it is also possible that attackers may find other lower level interfaces to read the frame buffer. The second issue is how to prevent the installed protection driver from being compromised, bypassed or overridden by third party drivers. This requires controlling the driver installation/un-installation process and monitoring related driver stacks and service functions' entry points. Indeed, there are no complete solutions to the screen capture problem, because any trusted DRM controller that prevents screen capture can be executed on an OS running on a hardware simulator, making the original protection mechanism ineffective.

4.3.7 Digital Watermarking

One typical application of digital watermarking is to monitor and trace the distribution of copyrighted materials, where some hidden information representing the user's identity is embedded into the protected content and is difficult to destroy or eliminate. DOFS applies the same technique to the bitmaps sent to the end user's computer to protect them from any analog copy attacks, such as taking a picture of the display through a camera. One possible method is to intercept the function calls related to windows drawing of DOFS client, and embeds the user's identity as watermarks to the pixel values before updating the frame buffer. The other more secure implementation is to hook Windows Terminal Services and embed watermarks when the bitmap is sent from server to client.

Applying digital watermarking provides limited protection because a motivated attacker can always find a way to eliminate the watermarks, e.g. using OCR software.

4.3.8 Off-line Access

For off-line access to protected content, or external access that cannot connect to DOFS servers, the file content must be encrypted and distributed to end user's computer. The exported sensitive content must be able to protect itself from being saved into an unencrypted form, copy-and-paste text into unprotected applications, or unauthorized printing and screen capture.

The functionality of exporting sensitive files in a controlled way is provided by the Checkout Service, a content packager on the DOFS server and is based on the tamper-resistant file protection technique. Basically, when users check out a protected file from the DOFS server, what is really exported by the Checkout Service is a package program containing the encrypted file content and a code module that performs run-time file decryption and rights enforcement. The content rights can be expressed in standard REL such as XrML. This package program starts an application process associated with the protected file type when uses open it, and redirects its file access to the embedded file content decrypted at run time. It also loads the screen capture controller, which can be a user DLL or a kernel driver. In general, this is essentially a virtualization technique in that the executable content package provides a virtual execution environment surrounding its application process. A similar approach has been applied to Thinstall (<http://thinstall.com/>), which offers "no installation" distribution and execution of application software. In addition, we will apply a set of obfuscation techniques to the package program to protect it from reverse engineering by attackers. The off-line access support is an important functionality for DOFS to become a commercial system, but it still does not provide stronger protections than existing E-DRM systems.

4.4 Evaluation

4.4.1 Scalability

The main concern about DOFS's scalability arises from the fact that applications operating on protected files are required to run on one or several DOFS servers using WTS. In this subsection, we will examine the startup latency of individual applications and the CPU/memory consumption on the DOFS server. To be sure, WTS actually supports server clustering to improve overall throughput and fault tolerance. We used a DELL desktop with an Intel Pentium 4 2.4GHz CPU, 768MB PC2700 DDR memory as the DOFS server, and an Acer desktop with an Intel Pentium 3 650MHz CPU, 384 MB PC133 SDRAM memory as the DOFS client. Both the server and the client computers run the MS Windows 2000 server OS.

We tested different types of documents using applications in Microsoft Office Suite and the result shows that when the available physical memory on the DOFS server is more than 20MB, the difference in startup latency between local execution and DOFS-mode execution is only about one second on average. This value is totally acceptable compared with the normal startup latency in Windows environment. As the number of connected terminal sessions increases, e.g. increasing to 30 or 40 sessions, the startup latency remains almost constant. Only when the available physical memory on the DOFS server drops below 20MB, the latency starts to increase significantly, i.e. 5 seconds or longer, because of excessive memory swapping.

The CPU usage on the DOFS server is also related to its memory consumption. When the server's available physical memory is more than 20MB, its CPU usage increases to a peak value between 50% and 90% at the time when a terminal session starts or terminates, but quickly decreases to a lower average value. However, once the available physical memory becomes smaller than 20MB, the CPU usage remains at a high peak value and that is the time when the startup latency starts to deteriorate.

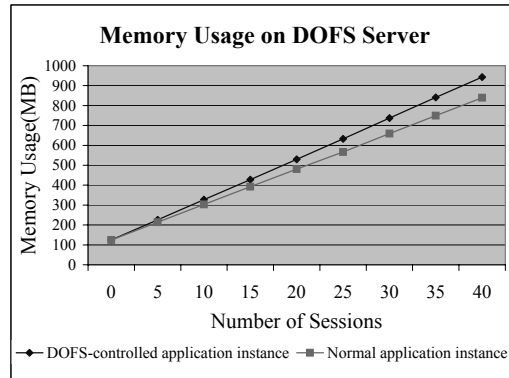


Figure 7. Memory usage on DOFS server

The memory usage on the DOFS server increases linearly when the number of connected terminal sessions increases, as shown in Figure 7. In this test, we created a series of terminal sessions, each running a Winword instance that opens a word document, and the memory consumption on the DOFS server increases at a rate of about 20MB per session. To isolate the contribution of the DOFS server, we removed all the DOFS components and opened the same Word document in terminal sessions again, and the increment rate for the memory usage became about 18MB per session. This shows that the DOFS server components consume only around 2MB per session. The Winword application in this case consumes 8MB because it contains many images. This means that there is always a fixed overhead of about 10MB associated with each terminal session even if it does not run any applications and Terminal Services is still a costly solution as the remote display scheme in terms of memory consumption.

As for the network bandwidth, because much of the bandwidth consumed is used to support updating and refreshing the client's graphical display, the bandwidth consumption could be small enough for applications that do not need significant graphical refreshing. In the capacity and scaling test performed by Groupe Bull and NEC engineers, the total bandwidth consumption for 40 knowledge workers is only about 85,000 bytes/second [31], which is pretty small and affordable in an enterprise environment.

4.4.2 Attack Analysis

In this subsection, we analyze the weakness of the DOFS architecture and its current Windows implementation.

Bypass client control: Some client components such as the shell extension and the clipboard monitor are mainly used to automate the access to sensitive files on the DOFS server. Bypassing these client controls by overriding the shell extension or breaking the clipboard viewer chain may cause the client to lose its functionality, but does not help to obtain any confidential information because the DOFS filter driver filters the file content on the server side. However, other client component such as the screen capture controller requires stronger protection to ensure the integrity

of the DOFS architecture. Including a tamper-resistant mechanism into DOFS client implementation or performing control interception at a lower level can help to resolve this problem.

Compromise rights enforcement for checked-out files: This kind of attack mainly comes from experienced attackers. Its purpose is to break the protection that safeguards sensitive files exported to mobile users. Once an attacker reverse-engineers the protection technique, she will be able to steal the content of any sensitive files that is checked out of a DOFS server. This problem is fundamentally related to the strength of tamper-resistant mechanisms embedded to the checked-out content package.

Execute malicious content: A user may unintentionally check into the DOFS server a file containing malicious content like macro viruses. Once the malicious code is triggered, the DOFS server may be damaged or contaminated. One solution against this attack is to install anti-virus software products in the enterprise environment, and to deploy behavior-blocking products in the DOFS server. Anti-virus products can identify and remove known viruses, and well-designed behavior-blocking products can stop malicious executables from damaging the DOFS server.

Trojan horse: Trojan horse refers to some malicious code that is hidden inside a data file or program, and can be executed under some specific conditions. Instead of damaging the DOFS server, the Trojan horse may establish a network connection with a malicious client process and transfers the protected content to the client computer, similar to the *covert channels* concepts in MLS systems. One possible way to resolve this problem is to install a simple firewall on the DOFS server that only opens the necessary TCP ports for DOFS server operation.

Denial of service: If a malicious user keeps opening sensitive files stored on the DOFS server, he may create too many terminal session, consume most resources on the DOFS server and lower the whole system's performance. To prevent this kind of denial-of-service attack, the WTS on a DOFS server is configured to set an automatic timeout interval for each connected session, and the total number of terminal sessions that are allowed from a single client IP is also limited to a small number.

Attack on Terminal Services: The current DOFS implementation utilizes MS Windows Terminal Services to support server execution and remote display, so attack on the terminal services can also cause sensitive information to leave DOFS server. A well-known vulnerability is that the RDP (Remote Desktop Protocol) implementation didn't verify the server's identity when setting up the encryption keys for the RDP session, which may result in the man-in-the-middle (MITM) attack [10]. In this way an attacker can construct the session key and use it to restore the transmitted display bitmaps. A more secure RDP implementation may help to resolve this problem.

Other attacks: It is almost impossible to build a system that can completely prevent all information theft attacks. For example, an authorized user can always remember the information of a sensitive file after viewing it, types this information into the computer at home and sends the result to someone else. This kind of information leakage always exists and probably cannot be prevented by any technical solution, including DOFS.

5. CONCLUSION AND FUTURE WORK

Enterprise digital rights management helps organizations to protect their confidential information and intellectual property from information theft, which can lead to significant financial loss and increased legal fees. E-DRM systems secure digital content by encryption or isolation to prevent users from directly accessing the content, and enforce access and usage rights by trusted DRM

client software that authenticates with DRM license or policy servers to get the decryption keys and rights policies. One fundamental issue with E-DRM systems is how to effectively protect the digital content from unauthorized access when it is distributed to end user's desktop computers, including how strong is the protection to the trusted DRM client and how comprehensive is the prevention of screen captures. Another issue is how secure and transparent is the integration of E-DRM systems with existing enterprise IT environment and desktop applications.

The key contribution of the DOFS architecture is that it prevents users, authorized or not, from having access to the physical bits of sensitive files, and thus provides a different, presumably stronger protection against information theft committed by corporate insiders than most existing E-DRM solutions in both research and commercial enterprise world. In addition, DOFS provides this protection without affecting enterprise users' normal working practice. Finally, DOFS is completely compatible with existing IT infrastructures and thus significantly reduces the deployment and administration cost in large enterprises. For mobile users without access to the DOFS server, DOFS also supports off-line access and provides particular content package programs to enforce rights policies. This off-line functionality is at least as strong as existing E-DRM solutions.

At this point, we are improving the current DOFS implementation by providing more client-to-server clipboard formats, by implementing a more robust and comprehensive mechanism to prevent screen capture, by embedding digital watermarks into display bitmaps, by supporting off-line access and rights enforcement, and by protecting DOFS client components with tamper-resistant mechanisms. Moreover, DOFS will also support more detailed content rights expressed by XrML to allow more fine-grained usage control. Although WTS is the best choice among all the remote display projects so far in term of performance overhead and usability, it requires expensive licensing charge and should be replaced with some other alternatives, such as Wine on Linux platform. Finally, the DOFS architecture can also be applied to the protection of sensitive information stored in enterprise database. In this case the DB client will be located on the DOFS server so users have no way to export any sensitive information from the database to their local computers.

6. REFERENCES

- [1] Andrew Conry-Murray. DRM: A Question of Balance. *Network Magazine*. December 2003.
- [2] Art Baker and Jerry Lozano. *The Windows 2000 Device Driver Book, A Guide For Programmers*. Second Edition. 2001 Prentice Hall PTR.
- [3] Bill Rosenblatt, Bill Trippe and Stephen Mooney. *Digital Rights Management: Business and Technology*. M&T Books. 2002
- [4] Brian A. LaMacchia. *Key Challenges in DRM: An Industry Perspective*. Proceedings of 2002 ACM Workshop on Digital Rights Management. November 2002.
- [5] Computer Security Institute (CSI) and the FBI, 2003 Computer Crime and Security Survey. <http://www.security.fsu.edu/docs/FBI2003.pdf>
- [6] David A. Solomon and Mark E. Russinovich. *Inside Microsoft Windows 2000*. Third Edition. Microsoft Press.
- [7] *Driving the Standard for Interoperability in Digital Rights*. MPEG REL Software Development Kit for Java User's Guide. ContentGuard Inc.

- [8] E. John Sebes and Mark Stamp. *Solvable Problems in Enterprise Digital Rights Management*. January 2004. <http://home.earthlink.net/~mstamp1/papers/DRMsebes.pdf>
- [9] Ellen Messmer. Security tools target inside jobs. *Network World*. April 2004.
- [10] Erik Forsberg. *Man in the Middle-attack against Microsoft Terminal Services*. Cendio System AB. April 2003.
- [11] *eXtensible rights Markup Language (XrML) 2.0 Specification*. ContentGuard Inc.
- [12] Joan Feigenbaum, Michael J. Freedman and others. *Privacy Engineering for Digital Rights Management Systems*. Proceedings of 2001 ACM Workshop on Digital Rights Management. November 2001.
- [13] KrishnaPG. *Various methods for capturing the screen*. The CODE PROJECT. <http://www.codeproject.com/dialog/screencap.asp>
- [14] *Liquid Machines and Microsoft RMS: End-to-end Rights Management for the Enterprise*. White paper. Liquid Machines, Inc. Feb.2004.
- [15] *Liquid Machines Technical Overview*. White paper. Liquid Machines, Inc. May 2003.
- [16] Mark Russinovich and Bryce Cogswell. *Filemon for Windows*. <http://www.sysinternals.com/ntw2k/source/filemon.shtml>
- [17] Mark Stefik. *Letting Loose the Light: Igniting Commerce in Electronic Publication*. Internet Dreams: Archetypes, Myths, and Metaphors. MIT Press. 1996.
- [18] Markus G. Kuhn and Ross J. Anderson. *Soft Tempest: Hidden Data Transmission Using Electromagnetic Emanations*. University of Cambridge.
- [19] *Microsoft Windows Rights Management Services for Windows Server 2003 – Helping Organizations Safeguard Digital Information from Unauthorized Use*. White paper. Microsoft Corporation. October 2003.
- [20] *Page Recall: The Key to Document Protection*. White paper. Authentica Inc.
- [21] Renato Iannella. *Open Digital Rights Language (ORDL)*. IPR Systems Pty Ltd. 2002.
- [22] Rick Smith. *The Challenge of Multilevel Security*. Cryptosmith LLC. October 2003.
- [23] Ross J.Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley Computer Publishing. 2001.
- [24] Stefan K. and Fabien A. *Information Hiding Techniques for Steganography and Digital Watermarking*. Artech House. 2000.
- [25] *Technical Overview of Windows Rights Management Services for Windows Server 2003*. White paper. Microsoft Corporation. November 2003.
- [26] *Trends in Proprietary Information Loss*. Survey Report. September 2002.
- [27] Tzi-cker Chiueh, Lap-chung Lam and etc. *Secure Mobile Code Execution Service*. Technical report, March 2004.
- [28] *Using and Understanding APIs for Terminal Server*. White Paper. 1997 Microsoft Corporation.
- [29] Victor DeMarines. *Content Security for the Enterprise*. White paper. Authentica Inc. April 2002.
- [30] Walter Oney. *Programming the Microsoft Windows Driver Model*. Second Edition. Microsoft Press.

- [31] *Windows 2000 Terminal Services Capacity and Scaling*. White paper. 2000 Microsoft Corporation.
- [32] Yang Yu and Tzi-cker Chiueh. *Display-Only File Server: A Solution against Information Theft Due to Insider Attack*. To appear in Proceedings of 2004 ACM Workshop on Digital Rights Management. October 2004.