

A Proposal for an Effective Information Flow Control Model for Sharing and Protecting Sensitive Information

Masato Arai^{†‡} and Hidehiko Tanaka[‡]

[†]Hitachi, Ltd., Systems Development Laboratory

292, Yoshida-cho, Totsuka-ku, Yokohama-shi, 244-0817 Japan

[‡]Institute of Information Security

2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama-shi, 221-0835 Japan

masato.arai.ez@hitachi.com, tanaka@iisec.ac.jp

Abstract

Information leakage has become a serious problem for computer systems that handle a company's sensitive information, such as intellectual properties and manufacturing know-how. The majority of the causes can be attributed to loss or theft of information or worms and viruses. As a countermeasure, forbidding the sharing of information through removable media or the Internet is effective, but it also places restriction on the handling of general information that can be made public. Also, the sandbox model can be used to segregate sensitive information from environments that can easily be infected by worms or viruses; however, even sensitive information is sent as email attachments to various locations within the organization, and this model cannot be applied to business cases where information must be stored and carried out on removable media. In this article, we propose an information flow control model that is suitable for both sharing and protecting sensitive information on computer systems in which general information that can be made public and sensitive information that cannot be exposed outside the company are mixed. In the proposed model, sensitive information are protected from environments that can be easily infected by worms or viruses by segregating the environment for programs that use the Internet and the environment in which programs handling sensitive information are executed, using existing techniques such as the sandbox model. At the same time, by combining automatic file encryption and encrypted file access control, sensitive information can be safely transmitted as encrypted text through removable media or the Internet as the need arises.

Keywords: Information leakage prevention, sandbox, file encryption, access control

1 Introduction

As the use of IT systems and the Internet for social infrastructure such as e-commerce and e-government increases, leaks of sensitive company information and personal private information is becoming a serious problem.

According to a survey by the Japan Network Security Association, the number of incidents involving leaks of personal information has climbed to nearly 1000 cases a year, counting only those reported in newspapers and on Internet news sites (JNSA 2008). The majority of the reasons are given as follows:

- Organization or personal handling errors resulting in loss or misplacement
- Theft of PC or recording media by a third party
- Mistakes in transmission, such as sending e-mail with the wrong address
- Discharge of information due to worm or virus infection

In other words, many incidents of leakage occur even without malicious intent by someone internal to the organization. These occurrences are not limited to personal information, but can be considered to apply across-the-board to sensitive information held by a company. As a result, there is an increase in the number of organizations that strictly restrict the carrying out of information and the use of e-mail attachments. However, if the client PCs operated by general employees are used for multiple purposes, from accessing the Internet to viewing and editing sensitive information, then not only sensitive information but also public information downloaded from the Internet exist on the PCs. Therefore, prohibiting the carrying-out of data or the use of e-mail attachments places restriction on the use of public information by employees.

There are products that block the copy of sensitive files to removable media or as e-mail attachments using copy detection techniques such as one described in article (Brin, Davis and Garcia-Molina 1995). Such techniques do not restrict the use of public data. They calculate the signatures of sensitive files designated beforehand, and place them in a list. By matching the signatures of files accessed by users, they detect the degree of similarity of the files with the sensitive files. Even if the contents of the files are somewhat modified, sensitive files can be detected. However, there is the problem of not being able to detect files correctly if sensitive files are inadvertently not designated as such.

With a digital rights management system, the contents are encrypted, and contents are protected by restricting their use to programs that can decrypt the contents (Ku and Chi 2004). However, the limitation of the use of specialized programs means that the data formats of protectable contents are also limited, which poses a problem to the convenience of such a system.

For methods that restrict themselves to specialized programs, a technique has been proposed that track and control how programs read sensitive information and where the information is output (Kurita, Shioya, Irie, Goshima and Sakai 2007). Under this method, security policies that grant permission to output devices are determined, and data that should be protected are saved with the policy. However, while it is non-problematic to view and edit sensitive information using software whose output on the network can be prohibited, such as a word processor, it becomes problematic when business practices require that the same security policies be applied to e-mail software, and when e-mail are sent to other locations. Considering the sharing of information in this way, security policies can change depending on the purpose of the program.

For example, besides secure OSes (Hallyn and Kearns 2000, Loscocco and Smalley 2001, Bauer 2006), which are equipped with enforced access controls (DOD. 1985) and can switch between access privilege for sensitive information and access privilege for network access (security policy) depending on the execution environment, and can separate the execution environment for programs with different purposes, such as those for using the Internet and those for viewing and editing sensitive information, there are also sandbox models such as WindowBox (Balfanz and Simon 2000) and SVFS (Secure Virtual File System). Both methods allow public information to be received from the Internet, and have the benefit of being able to segregate sensitive information from environments which can be easily infected by worms and viruses. However, even for sensitive information, there are times when such information cannot be segregated but must be attached in emails and sent to various points (different domains) within the organization. Also, there are business cases in which sensitive information must be stored on removable media and carried out.

Conventional methods such as the ones described above face major problems by requiring labor to make a list of sensitive information or by depending on programs to read sensitive information in order to protect sensitive information. Also, restricting the sharing of information on the Internet or removable media poses hindrance to business operations. Furthermore, considering the compatibility of sharing and protecting information, security policies can change depending on the sensitive information and the types of programs. Setting these policies can be troublesome.

In this paper, we propose an information flow control model that is compatible for both sharing and protecting sensitive information, which is a challenge faced by conventional methods. Our model presupposes the mixing on the company computer system both general information that can be exposed to the general public and sensitive information that cannot be exposed outside the company. Sharing information here means the delivery of files, for example through removable media, e-mail, or the Internet, in an organization having various locations with different domains. Furthermore, we explain a method of implementation that can easily deploy the proposed model in existing client PCs.

In Section 2, we describe measures against the leakage of information. In Section 3, we give an overall picture of

the proposed information flow control model. In Section 4, we describe its implementation, and in Section 5, we present its use scenario. Finally, in Section 6 we describe the study results of the proposed method's effectiveness.

2 Measures against Information Leakage

To lower the risk of information leakage that may arise from loss, misplacement, theft, erroneous transmission, worms and viruses, etc. and yet not hinder the sharing of necessary information in business operations, we propose an information flow control model that fulfills the requirements below.

- Sensitive information that cannot be exposed to outside parties are differentiated from public general information. No restrictions are given at all to the use of general information, whereas sensitive information can be protected even without the use of specialized programs.

- Sensitive information can be safely shared between relevant parties, even with the use of removable media or the Internet as needed.

- Access privilege (security policy) for the twin tasks of sharing and protecting sensitive information can be easily configured.

First, information is broadly divided into two groups: information used for business and information that can be made public. Information used for business are all considered to be sensitive information. To protect sensitive information without using specialized programs such as DRM, there are techniques to monitor and control the behaviors of programs (file access, network access, etc.). However, access settings for the control of these programs are not easy to configure. Thus, instead of giving access privilege individually to programs, we decided to build and segregate program execution environments based on the type of information (sensitive and general), and to grant privilege based on the execution environment. By doing this, common access privilege can be given to the same programs in an execution environment. Segregating the execution environment is effective in protecting sensitive information from environments in which infection by worms or viruses can easily occur. Also, to share sensitive information safely, information is always encrypted using a key shared among relevant parties. If the user has privilege, the information can be shared on removable media and the Internet and decrypted for viewing and editing.

3 Summary of Information Flow Control Model

We build the information flow control model, which fulfills the requirements of the security guidelines given in Section 2 above, by combining encryption of sensitive information, segregation of the program execution environment for handling sensitive information, and control mechanisms for file and network access.

3.1 Encryption of Sensitive Information

As shown in Figure 1, under the proposed information flow control model, all sensitive files are saved in an encrypted state, whereas general files are saved as plain text, so that sensitive information is concealed from all

except the relevant parties even when the sensitive information is carried out or stored in shared folders. Any type of removable media for file storage can be used. User authentication is carried out before sensitive files are viewed or edited. If the user has privilege for handling sensitive information, he or she is allowed to use the key needed to decrypt the file. Also, to prevent illegal decryption of sensitive information that slipped outside the company, user authentication and use of the key is limited to within the company only.

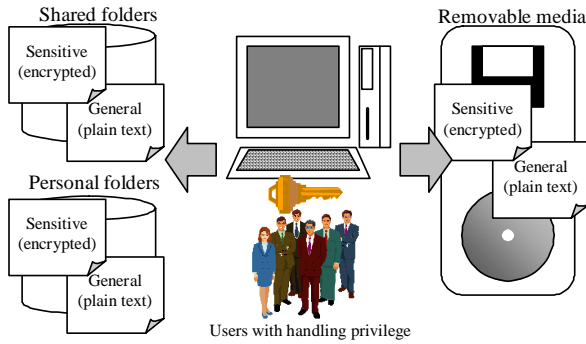


Figure 1: File formats for storage

The role of the file encryption described above is to absolutely conceal sensitive information to users who do not have handling privilege (key). The following actions by users with handling privilege (key) are opposed:

- (1) Decrypted sensitive information is saved as-is as plain text and carried out.
- (2) Decrypted sensitive information is sent as-is as plain text over the Internet.

These events may occur even without malicious intent by the user. There is especially the fear that event (2) may happen because of new kinds of worms or viruses that are undetected by anti-virus software. Thus in our information flow control model, the program execution environments are segregated depending on their purpose, and file and network access are controlled, in order to prevent the leakage of sensitive information as plain text.

3.2 Segregation of Program Execution Environments

In the proposed model, program execution environments are separated into two types, Type A and Type B.

Type A: Execution environment of programs that view/edit sensitive files, and programs that use the company network and access shared file servers, etc.

Type B: Execution environment of programs that use the Internet to send e-mail to various locations within the company, and programs that obtain information from outside the company online.

Also, along with setting control for file access and network access that occurs in each execution environment, by controlling the transfer of data that crosses over the execution environments, we prevent the leak of sensitive information as plain text due to worms, viruses, or erroneous operations. Below, we explain the file access and network access given to each execution environment, as well as the control of data transfer between programs.

3.2.1 File Access Control

As shown in Table 1, read and write privileges are assigned for programs that handle sensitive information (Type A) and for programs that handle the Internet and general information (Type B). From the table, we see that Type A programs, which are given the privilege for encrypting sensitive files, are not given the privilege for writing general files. Also, as described in Section 3.1 below, data written as new files by Type A programs are forcibly encrypted regardless of the type of media, and saved as sensitive files. This resolves problems such as sensitive information saved as plain text by Type A programs due to erroneous operation by the user. As for Type B programs, because they are not given privilege to decrypt sensitive files, the risk of leaking sensitive information as plain text due to virus infection or user error is reduced. Now, as for the reason why Type B programs are given permission to read sensitive files (encrypted as-is), this is so that sensitive files can be transmitted as-is in their encrypted state using the Internet. We explain in further detail in Section 3.2.2.

Object of Access	Type A Programs	Type B Programs
Sensitive Files	Read (Decrypt)	Read
	Write (Encrypt)	
General Files	Read	Read
		Write
New Files	Read (Decrypt)	Read
	Write (Encrypt)	Write

Read (Decrypt): Read while decrypting
Write (Encrypt): Write while encrypting

Table 1: File Access Privileges

3.2.2 Network Access Control

If, for the sake of argument, a Type A program with the privilege to decrypt and read sensitive files is equipped with communication functions, the problem of sensitive information being sent as-is as plain text through the Internet arises. At such times, the method of encrypting the data to be transmitted can be considered to solution. However, again encrypting sensitive files that have been temporarily decrypted for reading in order to transmit them raises problems in efficiency.

Object of Access	Type A Programs	Type B Programs
Internet	Deny	Allow
Intranet	Allow	Deny

Table 2: Network Access Privileges

As shown in Table 2, our proposed model addresses this problem by giving only Type B programs permission to use the Internet. For example, if e-mail software is defined as a Type B program, then as shown in Table 1, because sensitive files can be read as encrypted data, they can be attached as-is and sent as encrypted data. With this

scheme, the problem of sensitive information leaking through the network and the problem of overhead involved in re-encrypting data can be resolved.

3.2.3 Control of Data Transfer between Programs

In this section, we discuss measures to address the problem of unauthorized data transfer that may occur between Type A and Type B programs.

(1) Data Transfer from User Operations

Data transfers that accompany user operations include those due to drag-and-drop and copying/cutting to the clipboard. Drag-and-drop uses data transfer between multiple APs (applications) that appear on the same screen. Therefore, when Type A and Type B programs share the same screen, and sensitive information decrypted by a Type A program is sent to a Type B program, there is the danger of sensitive information leaking out as plain text. To address this problem, in our information flow control model, shells that display AP user interfaces (desktop) are generated each for Type A use and Type B use. User interfaces for programs of differing types are not displayed on the same screen. With this scheme, the desktop for sensitive use and the desktop for general use can be used by switching between them to meet the level of the information being handled. Figure 2 shows a representation of these desktops. On the desktops for sensitive use and general use, there are icons to switch between them, and by selecting the appropriate icon, the user can switch between the desktop he or she needs.

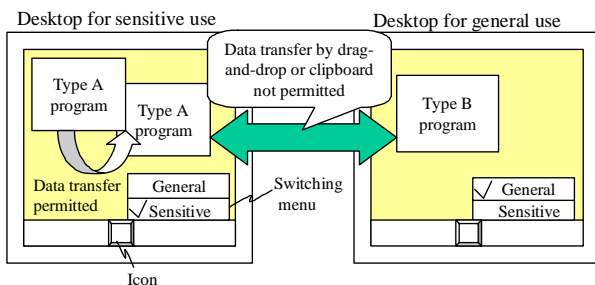


Figure 2: Image of multi-desktop

As for the clipboard, it is a shared data space where text and graphic data selected by the user are saved temporarily for other programs to copy. If the clipboard is abused, sensitive information is sent from a Type A program to a Type B program as-is as plain text. Thus in our information flow control model, only the sharing of the clipboard between the same type of programs is permitted. Sharing of the clipboard between programs of differing types is controlled so that sensitive information cannot be transmitted as plain text.

(2) Data Transfer Using Inter-Process Communication

Inter-Process Communication (IPC) here means, for example, data transfer between processes using named pipes. It differs from data transfer (1) described above in that no user operation is involved. If IPC is carried out between programs of differing types, sensitive information decrypted by a Type A program may be leaked out through a Type B program. Thus in our information control model, IPC is permitted between the same type of programs.

Between programs of differing types, IPC is controlled so that sensitive information cannot be transmitted as plain text.

4 Method of Implementation

We describe the method of implementation of the information flow control model described above.

4.1 Functional Requirements and Implementation Guidelines

We describe the functional requirements that should satisfy the functional requirements of the proposed model.

Requirement 1: Capability to create multiple program execution environments

Requirement 2: Capability to control data transfer that crosses over to another execution environment through IPC processes or the clipboard

Requirement 3: Capability to carry out network access according to the security policy shown in Table 2

Requirement 4: Capability to carry out file access according to the security policy shown in Table 2

Requirement 5: Capability to manage the decryption key for files so that it is not possible to legally obtain it outside the company

Below, we describe what kind of methods to use to fulfill the requirements described above. For requirement 1, in the same manner as WindowBox (Balfanz and Simon 2000), we use the method to build multiple program execution environments by desktop using Windows® OS as the base. For requirement 2, similar to WindowBox, we restrict IPC crossing over to another desktop. Also, for data transfer through the clipboard, we prevent it as described in Section 4.3.2. For requirement 3, we satisfy it using the same method as WindowBox. For requirement 4, in a different way from WindowBox, we fulfill it by implementing a file access control module and file encryption module as described in Section 4.3.1. For requirement 5, we use a group encryption system (GCS) (Ito, Susaki, Arai, Koizumi and Takaragi 1998, Arai, Kaji, Ito, Tezuka and Sasaki 1999) and install the group key generating server in a domain that cannot be accessed outside the company.

In Sections 4.2 and 4.3 below, we explain the implementation of areas in our guideline above that differ from WindowBox.

4.2 Encryption of Sensitive Information

To deal with the problems of loss, misplacement, theft, wrong transmission, and worms and viruses, it is desirable to encrypt and save sensitive information regardless of the storage location. It would also be desirable to have technology that makes it possible to share the decryption key between users who have privilege to handle sensitive information. An example of a simple way to share such keys is the Group Cipher System (GCS).

4.2.1 Outline of Group Cipher System

As shown in Figure 3, in a Group Cipher System (GCS), information is encrypted using a Group Key generated from the Decryption Control List (DCL) and the master key unique to the system. The DCL is a disclosure list of user and group names. It can flexibly specify a

combination of affiliation and position. When used to generate the Group Key, the DCL is added to the header of the encrypted information. During decryption, the privilege-checking program reads the DCL from the header of the encrypted information, and confirms whether or not the ID of the user attempting the decryption is included in the DCL. At this time, if the user ID is included, the Group Key generating program generates the Group Key; if the user ID is not included, the Group Key is not generated. The Group Key generated during decryption is the same Group Key used for encryption.

Here, for the Group Cipher System (GCS), there is the method of using IC (Integrated Circuit) cards to generate the Group Key and the method of using servers. For the IC card method, the master key and ID information, the privilege-checking program, and Group Key generating program are stored in an anti-tamper area of the IC card, so alteration of these data is prevented. For the server method, alteration is prevented by physically protecting the server. Also, according to the Group Key generating method (Ito, Susaki, Arai, Koizumi and Takaragi 1998), since the generated Group Key changes if the DCL changes, if the DCL attached to the header of the encrypted file is illegally altered, the Group Key needed to decrypt the file cannot be obtained.

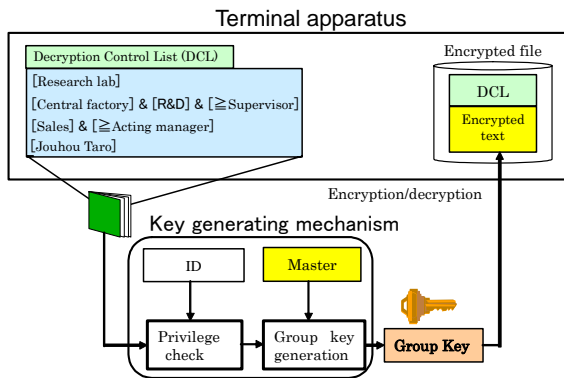


Figure 3: Outline of Group Cipher System

4.2.2 Decryption Control List of Sensitive Files

For a GCS user, authentication is carried out using the user identifier and password. At the same time, the user identifier is associated with ID information, as shown in Table 3.

No.	Category	Data	Code
1	Name	Jouhou Taro	
2	Date of Birth	1960/11/07	19601107
3	Sex	Male	M
4	Location	Systems development office	301
5	Department	Product planning	3
6	Position	Supervisor	9

Table 3: User ID information

Using the user identifier and password, authentication is established against the key generating mechanism, and Group Key generation becomes possible. The DCL, which is used during the Group Key generation, is expressed in conditional statements in concatenated form. For example, if sharing is desired for “position” (category value of row 6) greater or equal to “supervisor” (code value = 9), the DCL is specified as 6C>=9 (C is the initial for Code; the statement here specifies looking for the code value for Category row 6 greater or equal to 9.) Also, if all positions are given a code value or 1 or greater, specifying DCL to be 6C>0 allows the generation of the Group Key that can be shared by all employees within the organization. For example, if a sensitive file is encrypted and shared as information confidential to anyone outside the company, one should use a Group Key with this setting to encrypt the file. Thus, specifying the DCL at the time of Group Key generation by the GCS is necessary. However, it has the benefit of not requiring keys to be decided and shared among relevant parties beforehand.

4.2.3 Installation of Group Key Generation Server

Many file encryption products use the user’s personal password as the key for file decryption. However, in general humans tend to use passwords that are easy to remember (difficult to forget), and so the passwords are easy to guess. By contrast, the Group Key is generated by the DCL (described above) and the master key, so it distinguishes itself from the password used for user authentication (personal authentication). In other words, compared to passwords, the Group Key has the benefit of being difficult to guess. Even if a password is stolen by a third party outside the company, the Group Key cannot be handed over, and so the risk of information leakage is reduced.

As described above, the GCS has two methods of generating the Group Key: using IC cards (Ito, Susaki, Arai, Koizumi and Takaragi 1998) and using servers (Arai, Kaji, Ito, Tezuka and Sasaki 1999). However, considering the problem of inappropriate disclosure due to password guessing, there are also cases when the IC cards cannot be secured against the possibilities of loss, misplacement, and theft. Thus we use the server method to deal with this problem, and place the server in a domain that cannot be accessed from outside the company (for example, on the intranet). Even if passwords are stolen, the Group Key cannot be illegally obtained from outside the company.

4.3 Segregation of Program Execution Environments

To protect sensitive information by segregating the execution environments of programs handling sensitive information and programs using general information and the Internet, one can use technologies such as secure OSes (Hallyn and Kearns 2000, Loscocco and Smalley 2001, Bauer 2006) and sandbox techniques such as WindowBox and SVFS (Zhao, Borders and Prakash 2005). For our implementation on client PCs, we chose the sandbox

model, which can be implemented on Windows¹ OS, over secure OSes mentioned above, which are based on Linux², because Windows has a higher share of users. We give a summary of our implementation below.

WindowBox, one of the sandbox models, generates multiple desktops (shells) on Windows[®] OS. For example, a program execution environment for handling sensitive information and another for using the Internet can be established by desktop and segregated. As described in Section 3.2.3, based on the level of information being handled, the user switches between the desktop for sensitive use and the desktop for general use. Another example, SVFS (Zhao, Borders and Prakash 2005), executes multiple guest OSes (e.g. Windows[®] OS) on VMM (Virtual Machine Monitor), and segregates the execution environment for programs using the Internet and another for programs handling sensitive information by guest OS. Another similar type of commercial solution is Data Trans Guardian³ (HitachiSoft News Releases 2007).

There are other types of sandbox models, but the sandbox used here is defined as a model that has the capabilities for creating multiple program execution environments, for controlling file and network access from programs based on the execution environment, and for controlling data transfer that occurs between the programs in the different execution environments. Figure 4 shows a diagram of the implementation of these functions. In this example, resource segregation and assignment are carried by the OS or VMM, and file and network access and data transfer between programs are controlled by a reference monitor based on security policies. In our paper, we presuppose that the defense mechanism to protect against alterations to the reference monitor and security policies is provided by the sandbox itself.

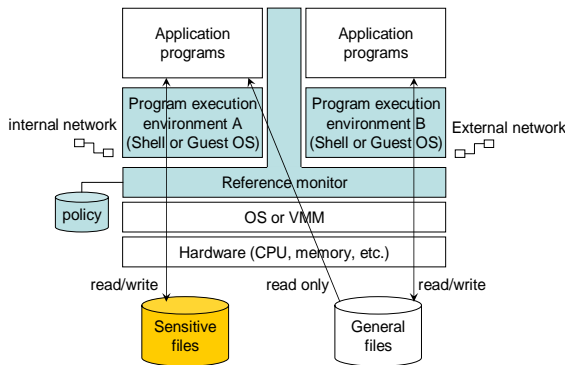


Figure 4: Conceptual diagram of sandbox

The model proposed in Section 3 shares the same network access control as WindowBox and SVFS. However, it differs from them in encrypting and storing sensitive files, and in how to give access privilege to these sensitive files (security policy). For example, WindowBox and SVFS absolutely prohibits the reading of sensitive

information by Type B programs (see Figure 4). This is because they assume that the sharing of sensitive information occurs through a company's private network (intranet). The proposed model allows the reading of encrypted sensitive information as-is. We allowed this action so that sensitive information can be exchanged using the Internet or removable media even by those who do not have privilege to handle sensitive information. For such occasions we allowed the safe sharing of information as encrypted files. Also, because WindowBox generates multiple desktops on one generic OS to segregate program execution environments, when it comes to OSes that allows sharing of the clipboard across desktops, it is possible to transfer data illegally between Type A programs and Type B programs. How to control this is not described.

For requirements 1 to 3 raised in Section 4.1 above, it is possible implement them using the VMM sandbox method such as SVFS and secure OSes. One is not limited to these choices; if the above requirements are fulfilled, they essentially unchanged regardless of which OS is used for implementation. However, authors suggest using the method that segregates program execution environments by desktop, such as WindowBox. This is because of the ease of deploying this model in client PCs. For example, for offices using PCs with Windows[®] OS, which already has a large share, implementing a VMM sandbox like SVFS or secure OSes as the base requires reinstalling the OS on each PC. However, with the proposed method, only modules such as one for launching the new desktops (for secure use and general use), for controlling file access, and for file encryption need to be installed on an already deployed OS. Setup is easier, as there is no need to re-install the OS.

4.3.1 File Access Control

As shown in Figure 5, file access is completely monitored by the file access control module through the file system. The module distinguishes between the types of program (A or B) originating the access request and between the types of the file being requested (sensitive or general). It then assigns access privilege as given in Table 1 above.

Judging the type of program is made possible by knowing from which desktop the program is launched. Specifically, judging from the user ID (privilege) assigned by the program is acceptable, or looking up a table that records from which desktop a program is launched may also be carried. Also, a program may be launched at the same time the OS is started, in which case it does not belong to either desktop. A class of programs called Windows[®] Services fall under this category. In the proposed implementation method, these Services are all treated as Type B programs, and are assigned file access privilege according to Table 1. Writing and decrypting sensitive files are prohibited.

As for the types of files to determine access, encrypted sensitive files have a special code embedded in their header. Access is determined by whether the appropriate code is present. Also, following the access privileges given above, when file access determined to be legal encrypts or decrypts for writing and reading, the file access control module calls the file encryption module. The file encryption module encrypts and decrypts sensitive files

¹ Windows is a registered trademark of Microsoft Corporation in the United States and other countries

² Linux is a registered trademark of Linus Torvalds

³ Data Trans Guardian is a trademark of Hitachi Software Engineering Co., Ltd.

using the distributed Group Key based on the privilege from the Group Key generating server.

The file access control module and file encryption module are naturally different in kind from the file access control provided by general OS. However, by using filter drivers (Filter Driver 2004), additional functions may be added to the file system to make implementation possible.

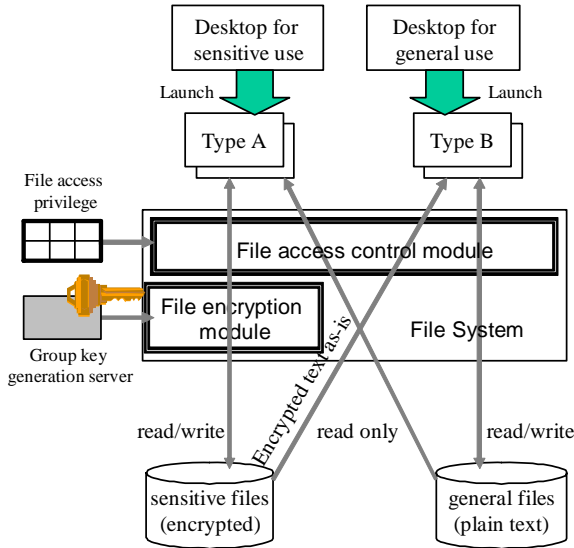


Figure 5: File Access Control

4.3.2 Control of Data Transfer through Clipboard

We describe the implementation method of the clipboard control module that prevents the sharing of the clipboard across desktops. The clipboard control module is executed immediately after the OS is launched. It is implemented as an active program until the OS shuts down. Also, every time the desktop is generated by user login, a memory space for temporarily storing the clipboard data is created in the memory. When the desktop is switched by the user, the clipboard data is moved to the temporary memory space mentioned above, and the data used by destination desktop is shifted to the clipboard (Figure 6). This does not limit the use of the clipboard within the same desktop. Please note that in order to protect sensitive data stored as plain text in the temporary memory space from being read by Type B programs, the memory space has a sandbox role. Its strength depends on the specification of the sandbox being used.

The process described above does not prohibit the sharing of clipboards between Type A and Type B, nor does it prohibit IPC as WindowBox does. It allows communication between the environments while preventing the discharge of sensitive information. One reason for being able to do so is that it regularly monitors access to the clipboard and IPC, and forces encryption of written data and communication data from Type A programs using the Group Key. This is similar to Type B programs not being able to read sensitive information as plain text despite given read privilege of sensitive files (see Section 3.2.1).

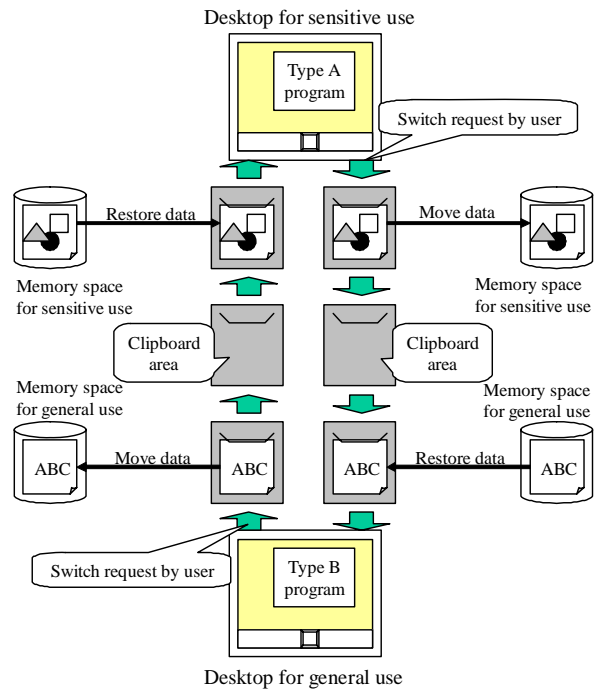


Figure 6: Information flow control through clipboard

5 Use Scenario

We explain the use scenario of an information system with the proposed information flow control model applied.

5.1 Example of System Configuration

The components of the smallest proposed system are the Group Key generating server and a client PC that divides the program execution environment into two using the sandbox technique (Balfanz and Simon 2000), as well as network equipment such as a router. Within the two types of program execution environments provided by the client PC, in the program execution environment for sensitive use (Type A), programs for accessing the company Group Key generating server and the file sharing server are installed besides programs for viewing and editing sensitive information. For the program execution environment for using the Internet (Type B), programs such as a browser and e-mail software are installed. Also, before sharing information, all work files except for files with only public information are considered to be sensitive files, and are encrypted and stored in the file sharing server or client PC. The DCL used for the Group Key generation is set as "6C>0" (sharing for all positions) (described in Section 4.2.2) for its default value. However, it is possible to change the sharing parameter based on the sensitivity level of the information. Please note that file encryption by the Group Cipher System (GCS) is not carried out only manually. We propose an automatic encryption function by which copying files to a folder set by the DCL automatically encrypts all those files. By using the setup described above, if a client PC (including a mobile PC) is lost or misplaced or stolen by a third party, because sensitive information is ordinarily stored as encrypted files and the Group Key generating server that creates the decryption key is installed in a location that cannot be accessed outside the company, the risk of disclosing

sensitive information to someone outside the company is reduced.

5.2 Transmission and Receipt of Sensitive Files

Editing a sensitive file and its flow from transmission to receipt is shown in Figure 7. The editing of a sensitive file takes place in the program execution environment for sensitive use (Type A), and is saved in encrypted state. When the sensitive file is attached to an e-mail, the program execution environment is switched to the one for using the Internet (Type B), the e-mail software is launched, and the e-mail is sent after the user enters the address, subject, body, and attaches the previously edited sensitive file. At this time, the attached sensitive file is read in as-is in its encrypted state and sent. For the party receiving the e-mail, he or she uses the e-mail software in the Type B program execution environment to receive the e-mail, and saves the attached encrypted file in an arbitrary folder. At this time, because the received sensitive file is encrypted, it is saved as an encrypted file. When decrypting the file for viewing, the user switches to the Type A program execution environment, and uses a program for viewing to open the encrypted file. At this time, the file encryption module shown in Figure 5 sends the DCL added to the encrypted file to the Group Key generating server and requests a Group Key for decryption. The Group Key generating server authenticates the user and checks privilege. Only when the user is included in the DCL is the Group Key generated and returned. This allows the sharing of the decrypted received file only by users included in the DCL (having privilege). If, for example, a sensitive file is mistakenly sent outside the company, because the Group Key server cannot be accessed from outside the company, the risk of information leakage due to erroneously sent mail can be reduced.

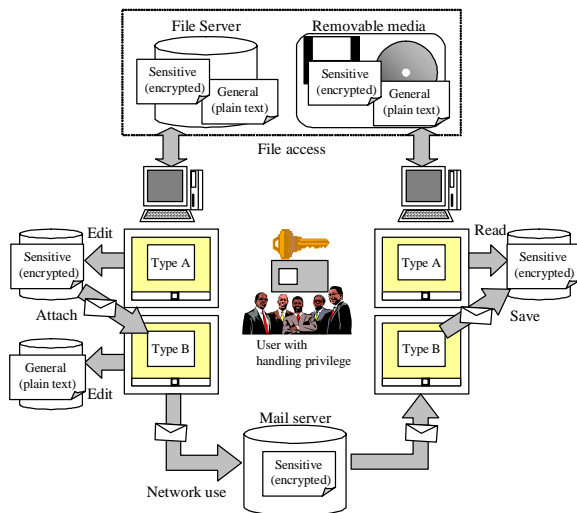


Figure 7: Transmission and Receipt of Sensitive Files

5.3 Sharing of Files with Removable Media

In the proposed model, a file saved from the Type A program execution environment is saved in an encrypted state regardless of the type of storage media. Therefore if the sensitive file is stored in a removable media such as a USB memory stick, it is automatically encrypted and

saved. Meanwhile, if a sensitive file is copied to a removable medium from the Type B program execution environment, the sensitive file is read in its encrypted state as-is and written to the new file in its encrypted state as-is according to the file access privilege given in Table 1, so the copied file is also encrypted. In other words, sensitive files saved in removable storage media are always stored as encrypted files regardless of the program execution environment. Thus, even when the removable storage media is lost or misplaced or stolen by a third party, because the sensitive information is always saved as encrypted files and the Group Key generating server which produces the decryption key is installed in a location that cannot be accessed from outside the company, the risk of disclosure of the sensitive information to someone outside the company is reduced.

For someone who receives a sensitive file, he or she directly opens the file from the removable storage media for viewing, or copies the file to the client PC. Either way, a user with privilege can only open the file from the Type A program execution environment. Also, for copying sensitive files from the removable storage media to the client PC, it can be from either Type A or Type B environment, but the copied file is always in its encrypted state. Technically speaking, when using a Type A program to copy a sensitive file, the file is decrypted and read, and then encrypted and written as a new file. When using a Type B program to copy a sensitive file, the file is read and written as-is in its encrypted state without undergoing decryption.

6 Study of Effectiveness

According to the information flow control model and its implementation described above, because all files used for business are considered and saved as sensitive files, work to create a list of signatures calculated from the sensitive information, as is done by fingerprint technology (Brin, Davis and Garcia-Molina 1995), is unnecessary. Also, dedicated programs to read sensitive information is necessary, and compared to DRM systems, it is a scheme that is less reliant on programs. Also, by using the Group Key to encrypt sensitive information, even if the key is not decided among the relevant parties beforehand, the sensitive information can be safely transmitted through the Internet or removable media. Furthermore, the access privilege (security policy) is fixed for each program execution environment, so it is unnecessary to change them.

When it comes to the advantages for everyday work, for offices that use two PCs, one for the Internet and one for the internal network, tasks can now be carried out on one PC, so convenience is greatly increased. For offices that use one PC for a variety of purposes, users are asked to switch desktops according to the task they are performing. However, if switching can be made a one-click operation, it can be considered to be the same as switching between windows and is thus acceptable to users. Also, for encrypting sensitive files, because the encryption is automatic with the default setting of the DCL as "confidential to outsiders" (6C>0), no special operation by the user is required. However, to control the sharing parameter within the company for information, some work

is needed to specify the appropriate DCL for target files and folders.

On the security front, one problem that arises is the receipt of a suspicious file by a program for using the Internet, which is then read by a program for handling sensitive information, resulting in a virus infection. Such a malicious program does the following for its attack:

(a) Force shutdown of the file access control module or disable protective functions using uninstall

(b) Intercept decrypted sensitive information at the Type A side by installing a malicious program

Action (a) can be countered by requesting administrator privilege and password entry during force shutdown and uninstall. For (b), the countermeasures differ for programs running under user mode and for programs running under kernel mode. For programs running under user mode, the attack does not directly lead to information leakage, because even if the malicious program reads sensitive information from the Type A side, Type A programs are not given permission to use the Internet. For programs running under kernel mode, it is not possible to gain control using the file access control module, and it is not possible to intercept decrypted sensitive information on the Type A side and transmit to the Internet using a program on the Type B side. For installation to the kernel mode, because administrator privilege is required, under the proposed model administrator privilege is given neither to Type A nor to Type B, so illegal installation can be prevented. The only problem left is the possibility of the attack maliciously exploiting the vulnerability of the OS to obtain administrator privilege and perform an installation. Also, the proposed model does not make provisions for attacks on the information integrity or availability. Therefore, repairing the OS' vulnerability and dealing with information integrity and availability are not problems handled by the proposed model. For these problems, there is a need to devise a plan that is completely separate from the proposed model, whose purpose is to prevent leakage of information.

7 Conclusion

In this paper, we proposed an information flow control model that is compatible for both sharing sensitive information and protecting sensitive information, and we also proposed a method of implementation that can be easily deployed on existing client PCs. We then described its effective points and problem points. With the proposed model, even if a removable medium or mobile PC is lost or misplaced or stolen by a third party, the risk of disclosing sensitive information to someone outside the company is reduced. Similarly, the proposed model is effective against the problem of sensitive files being passed to outside parties due to mis-sent e-mail or worms and viruses. Also, because the proposed model seeks to solve the problem of the leakage of sensitive information shared in the company, it divides information into two categories, General and Sensitive. However, three or more categories may be applied. For example, if information shared in the organization (Sensitive) is further classified as Top Secret and Secret to control access, it is possible to create the number of desktops based on the access levels (including General), and differentiate the Group Key (Top Secret and Secret) used to encrypt information to change the DCL.

In the future, we plan to investigate new use methods of GCS that makes it possible to share sensitive information safely with relevant persons outside the company.

Finally, the content reported in this paper is considered strictly as an example of a countermeasure on the IT security front. To deal with information leakage, a wide range of measures including physical countermeasures and user education is required.

8 References

- JNSA (2008), 2006 Information Security Incident Survey Report [Abstract] ver.1.0, NPO Japan Network Security Association, <http://www.jnsa.org/en/reports/incident.html>, Accessed 26 August 2008.
- Brin, S., Davis, J. and Garcia-Molina, H. (1995): Copy detection mechanisms for digital documents, *Proceedings of the 1995 ACM SIGMOD international conference on Management of data, San Jose, CA, USA, May 22-25, 1995*, ACM, pp.398-409.
- Ku, W. and Chi, C.-H. (2004): Survey on the Technological Aspects of Digital Rights Management, *Proceedings of the 7th International Conference, ISC 2004, Palo Alto, CA, USA, September 27-29, 2004*, Springer Berlin / Heidelberg, pp.391-403.
- Kurita, H., Shioya, R., Irie, H., Goshima, M. and Sakai, S. (2007): Dynamic information flow control for Preventing Information Leakage, *Proceedings of the IPSJ SIG Technical Report, HOKKE-2007, Hokkaido, Japan, March 1-2, 2007, 2007-ARC-172*, IPSJ Press, pp. 227-232.
- Hallyn, S. and Kearns, P. (2000): Domain Type Enforcement for Linux, *Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, Georgia, 2000*, USENIX Association, pp.15-15.
- Loscocco, P. and Smalley, S. (2001): Integrating Flexible Support for Security Policies into the Linux Operating System, *Proceedings of the FREENIX Track: 2001 USENIX Annual Technical Conference*, USENIX Association, pp.29-42.
- Bauer, M. (2006): Paranoid Penguin: An Introduction to Novell AppArmor, *Linux Journal*, Vol.2006, No.148, p.13.
- DOD. (1985). *Department of Defense trusted computer system evaluation criteria*. Department of Defense Standard DoD 5200.28-STD.
- Balfanz, D. and Simon, D.(2000): WindowBox: A Simple Security Model for the Connected Desktop, *Proceedings of the 4th USENIX Windows Systems Symposium, Seattle, Washington, Aug. 2000*, USENIX Association, pp.37-48.
- Ito, H., Susaki, S., Arai, M., Koizumi, M. and Takaragi, K. (1998): Group Cipher System for Intranet Security, *Trans. IEICE*, Vol.E81-A, No.1, pp.28-34.
- Arai, M., Kaji, T., Ito, H., Tezuka, S. and Sasaki, R. (1999): Group Cipher System for Enterprise Information System, *Trans. IPS. Japan*, Vol.40, No.12, pp. 4378-4387.
- Zhao, X., Borders, K. and Prakash, A. (2005): Towards protecting sensitive files in a compromised system,

Proceedings of the Third IEEE International Security in Storage Workshop (SISW'05), IEEE Computer Society, pages 21-28.

HitachiSoft News Releases (2007): Hitachi Software Engineering Co., Ltd., viewed 25 December, 2007, <http://hitachisoft.jp/english/news/news173.html>. Accessed 7 November 2008.

Filter Driver (2004): Filter Driver Development Guide 1.0a, Microsoft Corporation. <http://www.microsoft.com/en/us/default.aspx>. Accessed 26 August 2008.