RC 12401 (#55642) 12/18/86
Computer Science 34 pages

ABYSS: A Basic Yorktown Security System
PC Software Asset Protection Concepts

Dec. 18, 1986

Vincent J. Cina Jr. (CINA at YKTVMH)
Steve R. White (SRWHITE at YKTVMH)
Liam Comerford (CMRFRD at YKTVMH)

IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, New York 10598

IBM Research Report Number RC 12401

RC 12401 (#55642) 12/18/86
Computer Science 34 pages

ABYSS: A Basic Yorktown Security System PC Software Asset Protection Concepts

Vincent J. Cina Jr., Steve R. White, Liam Comerford

IBM Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Abstract

This paper describes the technical concepts of ABYSS (A Basic Yorktown Security System) as it relates to the problem of personal computer (PC) software asset protection. The current PC software environment is described as it relates to the protection of software assets. The basic concepts of the ABYSS system are then presented. This is followed by a discussion of new PC software marketing options that are made possible by the system.

Preface

The ABYSS system can span a range of capabilities, from the low end, applicable to small workstations (such as the IBM PC) to the high end, applicable to configurations containing large IBM mainframe processors and communications networks. This paper describes ABYSS concepts only as they relate to the IBM PC family of products.

ABYSS is a hardware and software system. Any ABYSS implementation on an IBM PC requires both hardware and software changes or additions to the personal computer system. A prototype ABYSS system is being developed by the Research Division of IBM.

The ABYSS system also provides solutions to related security problems, such as remote terminal access security and local area network (LAN) resource control. This paper does not discuss these aspects of the ABYSS system. The details of the cryptographic system or the cryptographic key management algorithms used in this system are also beyond the scope of this paper.

A glossary is provided in the appendix to describe new terms or terms that have a new usage in this paper.

CONTENTS

Introduction

Much has been written about the problem of personal computer software piracy. This paper discusses why previous attempts at a technical solution to this problem have failed and develops the concepts of a system that can prevent software piracy.

Organization

The current personal computer environment is described as it relates to the problems that are addressed by the software asset protection system. The goals of the system are then described. An overview of the the major components and new concepts is presented in preparation for a full discussion later in the paper.

Later sections describe the personal computer software asset protection architecture and the major components and concepts used in the system. New marketing techniques are made possible with the personal computer software asset protection system. These new techniques are surveyed in later sections.

"Appendix A. Usage scenarios" on page 26 in the appendix contains examples of how this system might be used.

Scope

The purpose of this paper is to introduce the concepts of the PC software asset protection system and illustrate how the system may be used.

Any implementation examples used in this paper are just used to help illustrate the concept being discussed. It should not be assumed that any implementation examples presented in this paper represent the best implementation alternative. Research is being conducted on specific implementation prototypes and the new ideas that have been formulated during the development of the system.

The research work that has produced the concepts and architecture for the personal computer software asset protection system that is described in this paper is called *A Basic Yorktown Security System* or ***ABYSS***. This paper essentially describes the concepts that are needed for a fuller understanding of the ABYSS system. The terms "ABYSS system" and "PC software asset protection system" are used interchangeably in this paper.

Current PC environment

Part of the success of the IBM personal computer family stems from the fact the IBM PC is designed to be an "open system". The term "open system" refers to the design characteristic of the IBM PC family that exposes all hardware and software, and associated interfaces, to the end user. These interfaces are described by IBM and enable anyone to examine or modify any information in the IBM PC.[1] The ability for end users, third party vendors, and other IBM divisions to modify or extend the hardware and software in an IBM PC has resulted in a wealth of hardware and software products for the PC. These products have contributed greatly to the success of the IBM PC product line.

Software Piracy Problems

Since the IBM PC is an open system, there is no way to prevent anyone from examining or modifying any part of the system. This has greatly contributed to the problem of "software piracy". The following situations are common in today's environment:

- An end user purchases a game from the computer store. The user tries the game and likes it. A number of duplicate disks are created and given to friends. Both the software developer and the software distributor lose revenue when illegal copies are distributed.

[1] Even read only memory (ROM) in the PC can be modified or replaced because all hardware interfaces are exposed.

- The software developer uses a "copy protection" mechanism to (try to) prevent the end user from making backup copies of the software. This can create any of the following problems.
  - The distribution diskette becomes unusable due to a physical defect, the user can no longer execute the program.
  - The copy protection mechanism does not allow the user to execute the program from a hard disk.
  - The copy protection mechanism allows installation on a hard disk but the original distribution diskette must be present for the program to execute. If the original distribution diskette becomes unusable, the user cannot execute the program. A backup of the hard disk does not effectively back up the installed (copy protected) program.
  - The copy protection mechanism allows installation on a hard disk and there is now no way for the user to execute that program on another machine, such as a backup machine.
- Copy protection systems have been produced that are sold to software developers to protect their software assets.
- Enhanced copy products (hardware and software) have been produced and sold to end users to defeat the copy protection systems.

A vicious cycle exists. New copy protection systems are developed, then new copy programs that defeat the new copy protection systems are developed. In fact a new sub-industry has developed with a set of vendors developing new copy protection systems and a set of vendors developing newer copy programs, which are able to defeat the newest copy protection scheme.

With current technology there is no end in sight for this problem. With the current open PC system it will always (theoretically) be possible to defeat whatever copy protection mechanism can be developed. The fact that the system is open prevents an effective solution to the software piracy problem.

Goals

A set of objectives or goals have been established for a solution to the PC software asset protection problem. This section describes these goals and objectives.

Impact

Any solution to the PC software asset protection problem must have minimal impact to the software developer, the software distributor, and the end user. For example, current copy protection mechanisms are a great inconvenience to the end user, these types of solutions are not acceptable. Software developers should be able to produce protected software without much more effort than they are using today with copy protection mechanisms.

Cost

The cost to the end user must be minimal. The purchase of a secure software asset protection system (hardware and software) may provide little direct economic benefit to an individual end user[2]. Most of the direct economic benefit goes to the software developer and distributor. The cost should be low enough to the software developer or distributor so that they do not need to pass on a large cost increase to the end user. Eventually, the end user should benefit economically from an effective PC software asset protection system. Since unauthorized execution of a software package is prevented, software developer and distributor revenues may be higher through greater sales volume. Thus the end user cost for an individual software package may even decrease.

---

[2] Direct economic benefit is possible to the end user if the software developer takes advantage of new license agreements that are made possible with this system. An economic benefit from the ability to return software for a refund or rent software is now possible.

Performance

The execution speed of the protected application should not be greatly reduced by the software asset protection system. A slight performance degradation may be acceptable, but the user should not perceive poor performance or inconsistent response times due to the software asset protection system.

Distribution methods

The term "distribution method" refers to the way the software developer gets the software to the distributor and the way the distributor gets the package to the end user. A PC software asset protection system should not adversely affect any current or future distribution channel.

A common distribution channel in use today is the local neighborhood computer store. Software products are kept on the shelf or on display in the computer store. An end user buys the package off the shelf, brings it home, installs it and runs it. A future distribution channel may involve the broadcasting of software from a satellite. The PC software asset protection system should complement any existing or planned distribution methods, and possibly enable new distribution or marketing methods.

Flexible terms and conditions

The following relationships exist between personal computers and end users:

- An end user exclusively uses only one PC.
- One PC is used by many different people at different times.
- One user uses many personal computers[3].
- The PC being used may be a part of a local area network (LAN) with services to access shared data and programs.
- The PC may be connected to a host mainframe system to obtain services or data that reside on the host.

In addition to these relationships both the hardware and software may be owned by a large enterprise, an organization within an enterprise, or an individual.

These factors contribute to the software piracy problem because existing license agreements do not make good provisions for them. The lack of an effective PC software asset protection system limits the types of license agreements that might be developed to permit more flexible terms and conditions to encourage the legal use of these relationships.

The goal is to permit flexible terms and conditions and be able to enforce them.

Ability to publish

The details of a PC software asset protection system should be publishable without compromising the security of the system.[4] This will enable the system to become a standard in the industry. Software developers and distributors will be able to examine the systems details, know how it works, and have trust in its security and usability.

Security

One guideline for the level of security needed in a software asset protection system is that it should take more effort (money, time, or resource) to defeat the system than it would take to independently develop the protected resource.[5] Unfortunately, while this guideline is necessary, it is not sufficient to describe the goals of an effective software asset protection system. Some pirates may not care how much time and effort it takes to break the security of a system, it is the challenge that is important.

---

3 Those personal computers may or may not be used by other users, and may or may not be portable.

4 An example of such a system is the Data Encryption Standard: the details are published and the system is secure.

5 For an extreme example: You can put a mud pie in the vault at Fort Knox, and it will be very secure, but, since anyone with some water and dirt can make a mud pie, placing one in a vault does not increase the security of mud pies.

For example, a student at a large university might spend all weekend, in a university lab, using university equipment, to break a security system, then publish sensitive information in the campus newspaper, all just for the "fun of it".

Components

The ABYSS system introduces some new concepts that are mentioned in this introduction and covered in more detail later in the paper.

PC software asset protection architecture

An architecture has been developed to describe the software asset protection system at a level independent of any implementation. The architecture contains a set of services which constitute the operations of the software asset protection system. The concepts described in this section are discussed in detail in [Whit86].

Protected processor

As stated earlier, with an open system, software piracy cannot be prevented effectively. The ABYSS architecture defines a ***protected processor*** which is a "closed" portion of the system. Since part of the system is now closed, it is now possible to build an effective software asset protection system to prevent piracy. The protected processor consists of both hardware and software and is both logically and physically secure.

The logical security of the system prevents any software in the system from accessing and exposing secret information. Cryptographic techniques are used as part of the logical security system. When protected information is outside the protection of the physical security system it is encrypted.

The physical security system prevents the access and exposure of secret information by physical means. For example, the physical security system is designed to prevent the use of hardware probes to access secret information within the protected processor. [Wein86]

Software partitioning

In order for a software package to be protected some of it must execute in the protected processor, where it cannot be accessed by any other hardware or software. The software must be split into a "protected" part and an "unprotected" part. This is called software partitioning. The unprotected part of the software resides in personal computer memory and executes in the personal computer's main processor (such as the Intel 80286 processor in the IBM PC/AT). The protected part of the software executes in the protected processor and therefore cannot be examined or modified. When the protected part of the software product is not in the protected processor it is encrypted. This prevents a pirate from obtaining any useful information about the protected part of the software. Both the protected and unprotected parts of the software must be present in the system for the application execute.

The way the software package is partitioned into a protected and unprotected part is important to the logical security of the system. The protected part of the software must be chosen in such a way that its contents cannot be determined by examining either the unprotected part of the software or the interaction between the protected and unprotected parts.

The right to execute

Today, the possession of a software distribution diskette, or a duplicate copy (backup) of the distribution diskette is all that is needed to execute a program. The software asset protection architecture separates the possession of the software from the ***right to execute*** that software. Having a copy of the software package does not give one the right to execute the package.

The control of the right to execute a software product protected by the software asset protection system is a very important part of the system. The "right to execute" is kept in the protected processor where it cannot be directly examined or changed, except as specified by the software vendor.

The authorization process

In order for the software asset protection system to control the "right to execute" for a particular software product, it must be able to control the number of times that "right to execute" is installed in a protected processor. The architecture describes a general process, called the ***authorization process***, which can control the number of times a supervisor service is done. The authorization process can be performed by trusted humans, trusted protected processors, or by a hardware implementation of a "token"

For now, suffice it to say that the authorization process can securely control the number of execution rights for a program. For example, the number of end users given the "right to execute" for a software product just purchased in a computer store can be limited to one.

Distribution methods

Any full discussion of the PC software asset protection problem must include a discussion of distribution methods. The architecture has separated possession of the software from possession of the "right to execute" that software. The software asset protection system is concerned with both the distribution of the software product and distribution (and control) of the "right to execute" that software.

Some security systems fit some distribution methods nicely and do not work well with others. We analyze current and possible future distribution methods and categorized their essential characteristics. The PC software asset protection system described here works with all known distribution methods.

Application data store

One of the goals of the PC software asset protection system is to support flexible terms and conditions. This requires persistent storage in the protected processor, associated with protected software products. There are a number of other facilities that applications might want to provide that require saving some application data securely between executions of that application. Today this is usually accomplished by storing that data in a file. Such a file can be examined or changed by the end user, so sensitive information is not secure when stored in this manner. The "protected processor" provides a secure place to store such information, called the ***application data store***. Since the application data store is in the protected processor it can not be examined or changed by any hardware or software, except the owning application.

The application data store can hold data that is used to enforce the flexible terms and conditions in the license agreement and any other sensitive facility the application wishes to implement.

Authorization process

The "authorization process" is a very important concept in the PC software asset protection system. This section describes the ***authorization process***, the ***token***, and the ***token descriptor*** which can be used by the authorization process. The authorization process is described before a discussion of the PC software asset protection architecture because it *is* used extensively within the architecture.

What it does

In general, an authorization process is a mechanism which limits the number of times an action or process can take place. An example of a process which one might want to limit is the loading of the "right to execute" a software product into a protected processor. Authorization processes are used by the software asset protection system to control execution rights. An authorization process must be highly resistant to forgery. The entire process must be capable of being observed without significantly increasing the chance of forgery. If the authorization process could be forged, the software asset protection system would lose control of the number of execution rights.

The form it takes

An authorization process does not necessarily require any special hardware. For example, your employer may only allow you to ask for one day off with pay. A note may be made (mentally or physically) when the request is granted. This effectively limits your days off to one.

An authorization process may also use a special piece of hardware, called a ***token*** which is used in the limiting function. An example of a "token" is a theater ticket. You purchase a theater ticket good for one entry into a show. When the ticket is presented at the door you are admitted and the ticket (token) is destroyed. The ticket must have the property that it is difficult to duplicate or forge.

The software asset protection system uses authorization processes in many of its operations. Implementations may or may not require a token (special hardware). Some implementation scenarios use a central service bureau to keep track of when things are done. Contact with the service bureau might be through personal contact, telephone conversation, or data communications (PC to service bureau computer). Any such communication has to be protected against duplication or forgery and service bureau records must be kept secure.

Electronic tokens

The ABYSS system introduces a novel authorization process, embodied in an electronic token. A sample implementation is a small chip that can be inserted into a token reader slot in the protected processor. A token can also be implemented in a small protected processor which is the size of a common credit card, called a "smart card". The protected processor connected to the personal computer would contain a reader for the credit card sized protected processor. In either implementation token information is accessed by the protected processor through the reader[6]. Electronic tokens are described more fully in [Stro85].

Regardless of implementation, the token information is discharged (logically or physically) from the token when the subject function is performed (in much the same way that the ticket taker at the theater door destroys your ticket upon admittance to the theater)[7]. The token itself must be logically and physically secure. Information contained within the token is used by the authorization process to identify the resource and the specific function to be performed. For example, if the function being limited is entry into the theater, a "token" contains information (unlike a normal theater ticket) that identifies the person (resource) that is allowed entry. If the token information could be completely

---

[6] A token could also be implemented in a large mainframe host system and communicate with the protected processor via telecommunications.

[7] This example assumes the authorization process using the token is to limit the number of actions to one. Tokens can be produced to limit the number of actions to any (reasonable) number. In the PC software asset protection system the limit of one is most often used.

examined, the token could be forged. The token could not detect that this has occurred and the authorization process could not perform its limiting function.

Token communications

The communication between the token and the protected processor does not have to be secure. Such communications can be completely observed without sacrificing the integrity of the authorization process or exposing token information. Observation of this communication does not allow the would-be pirate to forge or duplicate the token. This allows the protected processor to interact with the token using any communications path, for example, token reader slot in the protected processor or telephone based data communications.

Electronic token identification

The electronic token contains the information to ensure that the token process performs an action once (or any limited number of times). In the PC software asset protection system tokens are used to authorize the "right to execute" for an application to be loaded into a protected processor. The token is bound to a specific application by what is called a ***token descriptor***. The "token descriptor" is encrypted data that describes the contents of the physical token. The token descriptor is encrypted with a cryptographic key that is associated with the protected resource.

Whenever a token is produced to protect a sensitive resource a "token descriptor" is also created. Both the sensitive resource and the token descriptor are encrypted using the same cryptographic key. This binds the token to the sensitive resource. The token information is discharged when the operation that is being limited by the authorization process is requested.

Summary

Authorization processes can securely limit the number of times an action can take place (such as the ability to transfer a "right to execute"). Tokens can be used to authorize the function being requested.

PC software asset protection system architecture

In order to specify, in an implementation-independent way, the functions that a PC software asset protection system must perform, an architecture has been developed. The architecture calls these functions *supervisor services*.

Supervisor Services

The architecture defines a number of services, some required for any software asset protection system implementation and some optional[8]. The important services that are needed for a complete solution to the PC software asset protection problem are discussed in this section.

Load right to execute

As stated earlier, one of the most important jobs of a PC software asset protection system is to manage or control the "right to execute" for a particular application. This service manages the "right to execute" for a software product after it is purchased or transferred.

Distribution set

When a software package is purchased it comes in what the architecture calls a *distribution set*. To the architecture the distribution set is a logical construct: it has no physical form. We will use a common real-life example of a distribution set in this section for ease of understanding. You can think of the distribution set as the physical package that you take off of the computer store shelf. This package usually contains diskettes, documentation, and possibly a token. The diskettes contain the protected part of the software package (encrypted), the unprotected part of the software, and the "right to execute" the software package in the form of an application key (encrypted). A token may be included to authorize a "load right to execute" service.

Right to execute

The "right to execute" an application needs to be stored in a protected processor where it cannot be inspected or modified. The "right to execute" takes the form of a cryptographic key, called the *application key*. When the application key is not in the protected processor it must be encrypted. The application key is supplied by the software developer and is normally known (in the plain text form) only by the software developer.

The "load right to execute" service moves, in a secure manner, the "right to execute" from the distribution set to the protected processor. Normally this service is only performed once, when the software product is initially purchased and installed on a personal computer. The license agreement for the package may stipulate that the user may only use the software on one personal computer. The authorization process is used to control the number of times the "load right to execute" service can be performed.

While in a protected processor, the "right to execute" is the application decryption key that is used to decrypt the protected part of the software when the "load and execute application" service is executed.

A simple example, using current distribution methods, is used to illustrate the "load right to execute" service.

Example

When you purchase the new program product in your neighborhood computer store the distribution set contains the "right to execute" that product. The right to execute a software package must be loaded into the "protected processor" before the application can be executed. The authorization to

---

[8] Other services are also needed for an effective software asset protection system. These services exist in other systems and are not detailed in this section of the paper. They include: encryption and decryption services, application program transfer of control services, and random number generation services.

move the "right to execute" into the protected processor is also contained in the distribution set[9]. The license agreement states that this package can be used on only one system.

You take the distribution set home to your personal computer (an IBM PC/XT, with a hard disk) and follow the installation instructions in the documentation. You place the distribution diskette in the floppy disk drive and type **INSTALL**[10]. The install program uses the "load right to execute" service to move the right to execute from the distribution set into the protected processor[11]. You may make as many backup copies of the distribution diskette (at any time) as you wish.

Conservation of right to execute

The term "conservation of right to execute" has to do with the logical security of the system. A system that conserves the right to execute does not allow the "right to execute" an application to proliferate to unauthorized users. Much care was put into the development of the cryptographic key management algorithms of this system to ensure the conservation of the right to execute.

The distribution set contains the "right to execute" (or application key) of the subject software package. The end user is allowed to make any number of backup copies of the distribution diskette. This does not proliferate the "right to execute" to unauthorized users for the following reasons.

- While on the distribution diskette the application key is encrypted. It can only be decrypted by a valid protected processor, which is physically and logically secure.
- The application key is decrypted in the protected processor and stored in the protected processor by the "load right to execute" service. This service must be authorized to execute by the authorization process, which can limit the number of times the service can be done (the limit is usually set to one). The token is associated with the application by the fact that the token descriptor and the protected part of the application are encrypted using the same cryptographic key.

Therefore, having the original distribution diskette is of no use to the would-be software pirate. The pirate can have many copies of the distribution set diskettes. Such copies are useless because the "right to execute" is only meaningful when it resides in a protected processor. The authorization process completely controls the number of times the "right to execute" can be placed in protected processors.

Load and execute application

After the software has been purchased and installed on the system the end user wants to execute the application many times. The "load and execute application" service is used when the application is executed. The "load and execute application" service will be the most frequently used software asset protection service.

Application loading

The software product has two parts: an unprotected part, and a protected part that is encrypted by the software developer with the application key.

When the application is started, the unprotected part of the application is loaded into the personal computer processor. The "load and execute application" service is used to load the protected part of the application into the protected processor. If the "right to execute" the application has been loaded into the protected processor (using the "load right to execute" service at application installation time) the protected processor can operate on the protected part of the software.

---

[9] The authorization to load the "right to execute" is contained in a "token".

[10] Depending on the implementation of the authorization process you may have to insert a token into a slot in the ABYSS processor when (and only when) the program is initially installed.

[11] The authorization process is used internally to make sure the service is performed only once for this distribution set or any copies of the distribution set.

Decryption

The "right to execute" the application exists in the protected processor in the form of an application decryption key. This key is used to decrypt the protected part of the application inside the protected processor.

Application execution

Now that the unprotected part of the application is in the personal computer and the protected part of the application is in the protected processor in plain text (not encrypted) application execution can begin. Application execution begins on the personal computer which periodically makes calls to the protected part of the application executing in the logically and physically secure protected processor.

The application must be partitioned in such a way that both the protected part and the unprotected part must be present to successfully execute the application.

Application security

The protected part of the application is always encrypted when it is not in the logically and physically secure protected processor. The content of the protected part of the application is not accessible to the would-be pirate. Essentially, the protected part of the application executes in the "closed" part of the system.

The "right to execute" the application must be present in the protected processor in order for the protected part of the application to be decrypted and made executable. Since the system is built to control the "right to execute", the application can only be executed when authorized.

Create backup

The problem of backup, in case of failure, changes with the PC software asset protection system. This section describes why making backup copies of the software is no longer a problem. Creating a backup of the protected processor, which contains execution rights and application data, is addressed.

Software backup

With the PC software asset protection system as many backups of the software can be made as necessary. Any medium can be used, such as:

- Diskette to diskette copy
- Hard disk to hard disk copy
- Hard disk to diskette backup
- Diskette to host copy or archive
- Hard disk to host copy or archive

In addition the software can be stored on a PC network file server system or mainframe host system and be accessible to many users.

There are no restrictions placed on the ability to back up the software, and this includes the protected part of the software (encrypted), and the unprotected part of the software.

Protected processor backup

The "create backup" service addresses the need to make a backup copy of the application keys (and application data store) that are contained in the protected processor.

Whenever the end user changes the information in the protected processor a backup should be generated. This is to protect against the possibility of a hardware failure in the protected processor, and thus loss of the "right to execute" the protected applications[12]. This is a simple operation for the end user. The "BACKUP" command is issued and a *backup set* is created with all the important infor-

---

[12] The probability of a hardware failure in the physically secure protected processor is expected to be small as compared to the probability of a hardware failure in the personal computer system itself.

mation that is kept in the key store and application data store of the protected processor. The "backup set" includes an authorization process, which can authorize the installation of the backed up "rights to execute" on a replacement protected processor. The "backup set" can be implemented on a diskette and smart card pair. Along with the application keys and application data store, the unique serial number for the protected processor is also stored on the backup set. The "BACKUP" command uses the "create backup" service to copy this data to the backup set.

The data that is written to the backup set is encrypted with a special cryptographic key. This key is associated with the authorization process, to ensure that the backed up "rights to execute" can be installed on only a single replacement protected processor. The information in the backup set (but not the authorization process) can be duplicated any number of times by any conventional system.

Install backup

If the protected processor fails, the end user obtains another protected processor. The user can now the "install backup" service to load the backup information from the backup set into the working protected processor. The user can immediately start using the set of protected applications that were available on the now damaged protected processor. The "create backup" service uses the authorization process to ensure only the last backup level can be loaded with the "install backup" service.

Backup control

At this point, the end user has made a backup of the information in the protected processor and has installed that backup on another protected processor. If we just stop here we have introduced an integrity exposure into the system. What if the user made a backup of his protected processor with a targeted backup protected processor that his friend owned. The backup can now be given to the friend, who uses the special form of the "load right to execute" service to load the rights to execute onto his system. Now both users have the right to execute the programs. The software asset protection system has lost control of the "right to execute" for a set of applications, its primary job.

This exposure can be prevented. When a protected processor backup is made with the "create backup" service the serial number of the original protected processor is placed on the backup set in such a way that it cannot be changed. When the "install backup" service is used to install the rights to execute from a protected processor backup, a time limit is placed on the restored "rights to execute"[13]. This limits the amount of time the backup protected processor is usable for those rights to execute. For example, a time limit of two months might be used. After that time period the rights to execute in the backup protected processor that came from the backup set of the faulty processor will be made inactive. These "rights to execute" may be reactivated, or may be kept from becoming inactive in the first place, by using the "retain backup" service.

The PC software asset protection system has other options to protect against temporary unauthorized use of the protected processor backup set. Individual application developers can prevent the "right to execute" for their product from being placed on the protected processor backup set[14]. The "install backup" service can only be executed once for any given protected processor.

Retain backup

The life of a restored set of execution rights is limited when they are installed. The "retain backup" service removes that limitation. The conditions under which "retain backup" can be authorized are as restrictive and secure as possible.

Only some centralized agency[15] can authorize the execution of the "retain backup" service. The end user must present the damaged protected processor in order for the agency to execute the "retain backup" service[16]. A number of checks are made, which might include:

---

13 Protected processors contain a real time clock.

14 In this case the software developer would supply another backup mechanism.

15 An example of such an agency might be a central service bureau, the protected processor manufacturer, or the computer store.

- The protected processor must be damaged. The only reason for the central agency to remove the time limitation is if the original protected processor, which originally contained the execution rights, is no longer usable.
- The internal serial number of the backup must match the serial number of the protected processor[17].
- Centralized records must show that no "retain backup" services have been completed previously using this serial number.

If all checks are passed, the agent may perform the "retain backup" service or authorize the end user to perform it. In either case, the serial number of the original protected processor is recorded in order to prevent the "retain backup" service from being performed again for the same backup set.

Perspectives on backup services

Two things should be kept in mind when considering the way the PC software asset protection system provides for protected processor failure.

1. Actual failures of protected processors should be very infrequent. It is expected that protected processor failure frequency will be much less than the failure rate of the rest of the hardware system. Some reasons for this are that the protected processor package itself is very durable due to the physical security requirements and the number of components within the protected processor is relatively small.
2. Misuse of the backup facilities yield poor results for the pirate. The "right to execute" for a set of applications can be given to one (and only one) unauthorized user for a short finite time and the original user loses the ability to back up his system.

Even though a central agency is involved for the "install backup" service, the number of times an average user would have to use such facilities should be quite small.

Transfer right to execute

The ability to transfer ownership (the "right to execute") of a software product must be possible without loss of control of that "right to execute". The "transfer right to execute" service in the PC software asset protection system provides that ability.

How is it done

When a "right to execute" moves from one protected processor to another the "right to execute" must be removed from the original protected processor.

The transfer of a "right to execute" can be accomplished securely in two ways, through a protected processor to protected processor communication or by creating a "transfer set". A "transfer set" contains essentially the same information as the original distribution set.

In both cases the "transfer right to execute" service will remove the "right to execute" from the original protected processor. This can be assured because the program that implements the "transfer right to execute" service is part of the protected processor and is therefore logically and physically secure. A potential pirate cannot alter the way the service works.

Communications between protected processors

An implementation using the protected processor to protected processor communication method must establish a communications connection between the two protected processors. This communications channel need not be physically secure. The protected processors can establish a logically secure

---

16 The agency must have a policy as to what to do if the end user claims the protected processor is lost, and thus cannot be presented for the serial number verification check. This is no different than the situation today if an end user says the software product he just purchased 15 minutes ago was hit by lightning and disintegrated on the way home. What the agency does in this case is a matter of *policy*.

17 The agent uses the internal serial number of the protected processor if it is accessible, or an external serial number on the protected processor.

communication session between themselves using cryptographic techniques. This allows anyone to listen to the conversation between the two protected processors but does not allow any one to effectively alter or forge the conversation. See "Protected processor to protected processor communications" on page 23 for more information on this type of interaction.

Creating a transfer set

The method that creates a transfer set essentially reverses the "load right to execute" service process. This only requires the creation of a diskette with the protected portion of the software (encrypted), the "right to execute" (in the form of an encrypted application key), and the unprotected portion of the software. In addition a token must be created that authorizes the "load right to execute" service to be performed only once.

Both methods have advantages and disadvantages. They can both be used in the same system, thus obtaining the advantages of both.

Summary

The "transfer right to execute" service securely transfers the ownership of a software product from one end user to another (or back to the distributor for a refund). The techniques used effectively prevent a would-be pirate from obtaining an unauthorized "right to execute"

Access application data store

There is a need to keep certain information about an application in a secure place where it can be stored between executions of the application. Certain types of information, related to the software asset protection problem need to be stored to be able to enforce flexible terms and conditions in license agreements (see "New terms and conditions in license agreements" on page 20). Applications may also want to save other types of application related information in a secure place. Today applications usually store such information on an external storage medium like disk or tape. Unfortunately such information can be easily examined, changed, or destroyed.

The protected processor contains a section of persistent memory where application related information can be securely stored; this is called the ***application data store***. The application data store is a logically and physically secure storage area within the protected processor. Each application is assigned a section of this storage area. Applications can only access their assigned section of the application data storage area. The "access application data store" service is used by applications to access (read and write) their section of the application data store. The "access application data store" service is implemented by trusted programs within the protected processor. Only the protected part of the application, executing in the protected processor, can use the "access application data store" service.

Using the "access application data store" service, applications can now securely save any information they desire.

Protected processor

Effective PC software asset protection requires that some portion of the system be closed (not available for inspection or modification by the end user). The "protected processor" is the closed part of the system, and is both logically and physically secure. The logical security of the system is discussed first followed by information on the physical security of the protected processor.

Logical security

The logical security of the protected processor ensures that information within the protected processor cannot be examined, changed, or erased, except by authorized users in a controlled manner. Application programs and data are loaded into the protected processor by a ***supervisor process*** within the protected processor.

Supervisor process

The "supervisor process" is a processor state and a set of ***supervisor programs*** that execute in that processor state. The supervisor programs implement the PC software asset protection services. They reside in read-only memory (ROM) within the physically secure protected processor. The routines that implement a supervisor process are supplied by the protected processor manufacturer. The state the protected processor is in when supervisor programs are executing may allow access (read and write) to all the memory in the protected processor. Application code does not execute when the protected processor is in this state.

The main purpose of the supervisor programs is to control access to information in the protected processor. Requests for access to data and services in the protected processor are made by issuing PC software asset protection services. The code that implements these services (the supervisor programs) cannot be examined or altered by the end user.

These supervisor programs control the loading of the protected parts of applications, the loading of cryptographic keys, access to cryptographic keys, and access to the application data store.

The key store

Cryptographic keys are stored in an area of physically and logically secure memory within the protected processor. Some keys, called ***supervisor keys*** are placed in the protected processor by the manufacturer of the protected processor. Other keys, called ***application keys*** are placed in the protected processor by a supervisor program when certain ABYSS services are executed.

Supervisor keys

There are two reasons supervisor keys are needed in the protected processor:

1. The protected processor must be uniquely identifiable in order to support targeted distribution methods (see "Distribution methods" on page 18), secure backup, and transfer of execution rights.
2. In order to transmit secret information between any two parties, both parties must share some common information. The common information shared by all protected processors is the other type of supervisor key contained within the protected processor. This is called a "common supervisor key".

Application keys

Application keys are contained within the logically and physically secure protected processor. The keys are loaded into the protected processor by the "load right to execute" service. When in the key

store, an application key represents the "right to execute" for the application. Outside the protected processor, the application keys are encrypted by supervisor keys[18].

Application data store

The application data store is an area of logically and physically secure memory within the protected processor which contains sensitive data related to individual protected applications. Access to the application data store is controlled by supervisor programs.

The application data store is divided into sections, one section for each application. Each application is allowed access only to the section of the application data store assigned to that application. There are two types of information kept in the application data store.

The supervisor programs keep information related to a standard set of terms and conditions that are supported by the ABYSS system in the application data store. At a minimum the application data store must hold the expiration date associated with the "right to execute", if any. The "load right to execute" service places an expiration date in the application data store if the source of the "right to execute" is a protected processor backup set.

The other type of information kept in the application data store is defined by the application. An application can access (read or write) its section of the application data store using the "access application data store" service. Data placed in the application data store by an application is retained between executions of the application.

There are many possible uses an application can make of the application data store. For example, an application can keep track of the number of times it is executed, and thus implement usage billing.

Even though the amount of application data store memory available to an application is limited, any amount of information can be protected by using only a small amount of space in the application data store. This is best illustrated with an example. Suppose an application needs to store a large array between executions. Also suppose that it is extremely important that the next time the application executes, the array produced from the previous execution be loaded. Without the PC software asset protection system the best that could be done would be to encrypt the array so it could not be changed. There is no way to ensure the array being loaded was the last one created. A copy of the array from many executions earlier could be used without detection. With a logically and physically secure protected processor the following technique can be used to ensure the correct array is loaded. The protected part of the application, executing in the protected processor, obtains a random number from the protected processor. The random number is added to the array to be stored and is also stored in the application data store by using the "access application data store" service. The array and random number is then encrypted (in the protected processor) and passed outside the protected processor for external storage. The next execution of the application decrypts the array and verifies it is the correct version by using the random number in its application data store. That random number never leaves the physically and logically secure protected processor (without being encrypted) and thus is highly resistant to forgery. This technique is just as secure as placing all the data in the application data store except that there is no protection against the data being erased.

In summary, the application data store can be used by a protected application to store any amount of information securely between executions of the application.

Application process

For an application to be protected by the PC software asset protection system it must be partitioned into a protected part and an unprotected part. The protected part of the application executes in the protected processor in what is called the "application process". The protected part of the application is loaded into the "application process" by the "load and execute application" service. The way the protected part of the application interacts with the unprotected part of the application depends on the software partitioning method used to split the application. The application process is a processor state

[18] A supervisor program is supplied in the protected processor to allow an application developer to pass the application key into the protected processor and have it encrypted by a supervisor key.

in which there is no direct access (read or write) to the key store, application data store, or supervisor process.

The key store contains rights to execute (application keys) of all protected applications. The protected part of the application has no need to access the key store. If the protected part of an application had access to the key store it could move supervisor and application keys (for other applications) outside the protected processor, thus completely compromising the security and integrity of the system.

The protected part of the application does need access to the application data store. This is accomplished in a controlled way by using the "access application data store" service. The supervisor process that implements the "access application data store" service limits access to the section of the application data store assigned to that application.

The protected part of an application does not need access to the supervisor process. The supervisor process has access to all the memory within the protected processor and if an application could change a supervisor process the integrity of the system would be lost.

Random number generator

Many of the cryptographic techniques and other logical security schemes used in the PC software asset protection require the ability to generate a good random number[19].

Real time clock

One effective implementation of a good random number generator uses a real time clock[20]. The low order bits of a high resolution clock make good input into a random number generation algorithm. The real time clock must be contained within the physically secure protected processor so its operation and contents cannot be disturbed.

The real time clock is also necessary to enforce some of the time dependent terms and conditions that this system supports.

Encryption-decryption processing

The protected processor provides the encryption and decryption services used in the PC software asset protection system. The following types of information are encrypted or decrypted in the system.

- The protected part of the application
- The application key
- The token descriptor
- Data transmitted over a secure communications channel
- The key store
- The application data store

In addition to these types of data, applications may wish to encrypt data for reasons related to application processing which have little to do with the control of execution rights.

The protected processor provides encryption and decryption services which are accessible from supervisor programs and applications.

Physical security

It has been said that it is impossible to provide perfect physical security. While this may be true, it is also true that it is not necessary to provide "perfect" physical security to implement an effective PC software asset protection system (see "Security" on page 3). The level of physical security provided in the protected processor is an important implementation question.

---

[19] The ability to establish a secure communications channel and verify the contents of a token are examples of the need for random number generation.

[20] A high resolution counter could also be used in place of a real time clock.

Research is being done into new and innovative physical security measures which we believe provide more than adequate physical security. Some of the new physical security techniques employ active detection mechanisms which destroy all sensitive information within the protected processor when a physical intrusion is detected. The level of security provided is designed to not only thwart the hardware hacker but also the university student who has access to the most sophisticated laboratory equipment.

Summary

The protected processor is both physically and logically secure. The components within the protected processor are the "supervisor process" which controls access to the "key store" and "application data store" and the "application process" which executes the protected part of the application. The protected processor also provides services to generate random numbers, and cryptographic services.

Distribution methods

One of the goals of the PC software asset protection system is to support current and future distribution methods. This section discusses the way the PC software asset protection system views distribution methods. Distribution methods are generically classified. The classification method employed enables possible implementations to be judged as to their level of support for the differing distribution methods.

Separation of distributed parts

The PC software asset protection system separates the possession of a software product, or access to a software product, from the "right to execute" the software product. This separation is a fundamental concept and control of the "right to execute" is the primary responsibility of the PC software asset protection system.

Types of distribution methods

This section describes three types of distribution methods. Each distribution method takes into account how the software product is distributed and how the "right to execute" is distributed. The way we have classified distribution methods is not the only way to think about this problem though we have found it very useful in determining how flexible various implementation alternatives are in relation to distribution methods.

There are many ways in which things can be distributed today and the future will open new, exciting, distribution capabilities. In any distribution method the item being distributed is "targeted" to a potential customer. If the item is being distributed for an individual (system or user) we call it a ***targeted distribution***. A targeted distribution requires the item be tailored (individualized) to the purchaser. If the item is distributed for any potential user or system we call it an ***untargeted distribution***. With an untargeted distribution any distribution set can be purchased by any user.

To arrive at our classification scheme we combined the two items we are distributing (the software product and the "right to execute") with the "targeted" and "untargeted" distribution techniques to form a set of different distribution methods. The following distribution methods have been identified:

1. Untargeted software product, untargeted "right to execute"
2. Untargeted software product, targeted "right to execute"
3. Targeted software product, targeted "right to execute"

The combination "Targeted software product, untargeted right to execute" is not considered to be a useful combination and is not discussed. The following sections discuss the useful distribution methods.

Untargeted software product, untargeted right to execute

This type of distribution method is closest to the way PC software is distributed today. There are many software packages on the shelf in the neighborhood computer store. Each package of a particular product is identical and contains both the software and the (implied) "right to execute" the software. Neither the software nor the "right to execute" are tailored to any individual system or user[21]. Anyone can purchase and use the product.

The PC software asset protection system supports this type of distribution. The authorization to load the "right to execute" can be represented by a token that is contained within the package[22]. The package can be purchased off the shelf by any potential user.

---

21 Some software products contain unique serial numbers on the diskettes. These unique serial numbers do not make these packages "targeted". They do not effectively represent a "right to execute" and they do not affect program execution.

22 Other implementations, using other types of "authorization processes" are possible.

Untargeted software product, targeted right to execute

This distribution method is not commonly used today for marketing PC based software. With this distribution method software is readily available in identical packages[23]. Some contact with a service bureau is usually required to successfully execute the software. This service bureau contact represents the transfer of the "right to execute" to the purchaser.

The PC software asset protection system can support this type of distribution method without the use of a service bureau. Protected processor to protected processor communications can be established (in the computer store or over telephone lines) to transfer the "right to execute". A token can also be used.

Targeted software product, targeted right to execute

This distribution method is not commonly used today for marketing PC-based software. In this case both the software package and the "right to execute" are tailored to the individual purchaser. This is the most secure distribution method. It is also the most expensive distribution method. The software and the "right to execute" are useful only to the individual to which it was tailored.

The PC software asset protection system supports this type of distribution method. A unique serial number is used to identify the targeted system. It is thought that this distribution method will only be used for fairly expensive, specialized software. The maintenance of the "right to execute" can be done without the use of a "common supervisor key".

Future distribution techniques

The following distribution technique is supported by the PC software asset protection system with good usability characteristics. This technique is a possible distribution implementation and is presented here to illustrate some of the possibilities in the system.

A software development company can distribute its entire line of software on optical disk to all potential users for the cost of the media. The "right to execute" for any particular software product can be sold separately. This optical disk can also be used to demonstrate the software before the full license is purchased. Satellite broadcast, FM broadcast, public data network access, local area network access, or host mainframe access can be substituted for the optical disk media.

Summary

The PC software asset protection system supports both current and future distribution methods with minimal negative impact on the software developer, distributor, and end user. Some cumbersome distribution techniques are made easier (by eliminating the need for a central service bureau) and new distribution techniques are made possible with protection of software assets.

---

[23] This includes broadcast software and software available from a file server on a local area network or mainframe host system.

New terms and conditions in license agreements

One of the exciting benefits that is realized with the PC software asset protection system is the ability to support new, flexible terms and conditions in the license agreements for software. The PC software asset protection system not only supports new license agreements. It also provides a secure way to enforce terms and conditions in license agreements.

This section discusses some of the new types of terms and agreements that are possible with the PC software asset protection system. The actual usage of these new terms and conditions depends on the exact nature of the PC software asset protection system implementation and in the use of the application data store in the protected processor.

Unlimited software backup

The ability to make unlimited copies of software products is a direct consequence of the base architecture. Any implementation supports this capability. The ability to make unlimited backup copies of software comes directly from the fundamental concept that access to (or possession of) the software is separate from the "right to execute" the software. There are no restrictions on the ability to make backup copies of the software, though copies of the "right to execute" are strictly controlled by the system.

Limited lifetime software

If the implementation has a real time clock a time limit can be placed on the "right to execute" for a software product. For example, an application can be sold to work only one year, or only until a specific date. The application data store is used to hold the expiration date.

Allowable execution periods

The application can be built to execute only during certain time periods. For example, only on Saturdays, Sundays, or holidays will the games on this local area network be enabled for execution, or only during the last week of the year can this year-end financial report be generated. This requires a real time clock and application processing using the application data store.

Demonstration software

A software author may wish to freely distribute a demonstration copy of a new software product. Today this is accomplished by putting limited function in the demonstration copy. With the PC software asset protection system that has a real time clock the full functioning package can be freely distributed. The "right to execute" the full functioning demonstration copy can expire in a short time period (for example, ten days) or on a specific date. This is a specific use of the limited lifetime software technique described above. If the end user likes the demonstration copy of the program an unlimited "right to execute" can be purchased for full price.

Software guarantee

Many software packages are sold today "as is", without guarantee that the software fulfills the intended need for the end user. Distributors are usually unwilling to take software back because there is no assurance the end user does not still have a usable copy. The PC software asset protection system controls and conserves the "right to execute" a software package. A software return policy, with full refund, can now be established if the end user not only returns the software but also the "right to execute".

This new capability has a large positive economic benefit to the end user. End users typically have many software packages sitting on their shelf, full price paid, that are not used because the software did not operate as desired.

Distributors or software developers could also offer partial refunds for software returned after some period of time.

Rental software

Personal computer software could be rented, in much the same way that video cassettes are rented today. The software asset protection system is used to control execution rights and the application data store contains rental license information.

Test level software

Application developers may wish to distribute an early release of a new product to a select set of users to help test the new system. When the minor errors that are found are corrected the official product version is released to the general public. All the users that participated in the early test program now have copies of the program that are almost the same as the official product version. They have little incentive to purchase the official version. With the PC software asset protection system the ability to execute the test level software could expire after the date when the product level is available. Early product testers would then have to purchase the official product level in order to continue to execute the program. The number of potential testers can be increased, increasing the quality of the shipped software.

Usage pricing

Applications could keep track of when the software is executed. This is done by keeping a counter in the application data store. The application could also keep track of what important functions were being used within the application. The end user could then be billed for the usage of the application or important functions within the application.

Site license

All the personal computers within a site can be authorized to execute a particular software product. A single copy of the software could be kept on a file server on a local area network or host mainframe connection. Today many software developers find it difficult to make site license agreements because there are no guarantees that copies are not being used outside the legitimate systems. With a PC software asset protection system such assurances can be made.

Lending library

An enterprise may want to purchase a number of rights to execute a particular program and lend the rights out on a first come first served basis. For example, ten rights to execute for a specific program may be purchased. The program may be controlled by a file server on a local area network. The file server would allow authorized borrowers to "check out" a "copy" of the program for temporary usage. The file server keeps track, in the application data store, of the number of "copies" currently on loan. The file server would ensure no more than ten copies are in use at any one time.

Conclusions

The PC software asset protection system can enforce any type of license term or condition that can be programmed. Maximum flexibility is permitted. Combinations of the above examples may be implemented.

The implementation of any of these terms and conditions normally uses the real time clock or the application data store that is within the protected processor, both of which are physically and logically secure.

The ability to enforce a rich set of terms and conditions is one of the major benefits of the PC software asset protection system.

Software partitioning

The PC software asset protection system protects products by executing part of the software in the logically and physically secure "protected processor". The software product or application must be split or partitioned by the developer. This split produces two parts, the part that executes in the personal computer processor (called the *unprotected part*) and the part that executes in the protected processor (called the *protected part*). This section discusses the requirements for effective software partitioning.

Basic requirement

The protected part of the application executes in the "application process" in the protected processor. While in the protected processor the protected part of the application cannot be examined or changed by the end user. The protected part of the application is encrypted when it is not in the protected processor. The application developer must decide how to partition the application into a protected part and an unprotected part. The basic requirement for the protected part of the application is that it be difficult to reconstruct and it must perform important and necessary function. The would be pirate knows all about the unprotected part of the application and the interface between the protected part and the unprotected part of the application.

If the function of the protected part of the application could be easily guessed the pirate could create the function in the PC and thus create a copy of the application without the protected part. This would effectively defeat the protection for that application.

Investigations

In much the same way that work continues in the general field of application development technologies, research is continuing into the effectiveness of different partitioning methods. When analyzing a partitioning method the security afforded by the method needs to be determined. Partitioning methods with good security characteristics have been identified. Security is not the only parameter to be examined for a successful system. Performance, usability and cost are also important aspects to be investigated. Both analytical and experimental techniques are being used in these investigations.

A range of effectiveness using various partitioning methods has been demonstrated. The security of some methods is easy to break, using automated techniques. Other techniques provide an extremely high level of security. Specific protected processor implementations are being investigated to determine their effect on system performance, application performance, application security, and the program development process.

Summary

There are probably as many ways to partition software as there are ways to create the original software. Just as work continues today in ways to improve the application development process, we expect research to continue in ways to improve software partitioning. Security, usability, and performance need to be maximized while cost is minimized.

Reasonably secure software partitioning techniques have been identified and are being used today.

Protected processor to protected processor communications

Protected processors can communicate directly with each other. Such communication enables sensitive information to be transferred between protected processors.

Secure communications

All protected processors contain a secure common supervisor key. The common supervisor key is used by the protected processors to verify the identity of the protected processor they are communicating with. A protected processor will not pass sensitive information to another protected processor unless its identity has been verified. This prevents the would-be pirate from making a protected processor that divulges secret information.

The random number generator is used in generating a secure communications session key. All transmissions between the protected processors are encrypted using the session key to allow maximum flexibility in transmission medium. Protected processors can communicate over a public telephone network without any loss of security.

Transfer of rights

The protected processor holds the "right to execute" and the license agreement terms and conditions for each protected application. Secure communications between two protected processors can be used to move rights between protected processors. The supervisor programs within the protected processors make sure the "right to execute" for any application is conserved and that terms and conditions in license agreements are not violated.

Summary

Protected processors can communicate securely over an open channel, safely transmitting sensitive information. Such communication can be completely observed without loss of security.

Technical summary

These are the main concepts of the PC software asset protection system:

- The IBM PC is an open system. It is not possible to protect information in an open system. The ***protected processor*** is an addition to the personal computer system that provides a "closed" portion of the system where secure information is kept and managed.
- Part of the function of the application to be protected must execute in the "protected processor", this is called the ***protected part*** of the application. The rest of the application, called the ***unprotected part***, executes in the personal computer.
- The possession of, or access to, software is separated from the ***right to execute*** that software.

The main job of the PC software asset protection system is to manage the "right to execute" for protected applications in accordance with the terms and conditions specified in the license agreement. Cryptographic techniques are used to protect sensitive information when it is not inside (and thus protected by) the "protected processor".

Control of execution rights is accomplished without adversely affecting the end user, software distributor, software developer, system performance, or application performance.

Current software distribution methods are supported and new distribution methods and license agreement terms and conditions are made possible.

Implementations can be designed to meet a broad range of security requirements. The system is secure enough to be completely published and has the potential to become a security standard.

The ABYSS system provides the technical means for solving the PC software asset protection problem.

References

[Stro86] Bill Strohm, Liam Comerford, Steve R. White, "ABYSS Tokens," IBM Research Report Number RC 12402 (Dec. 18, 1986)

[Wein86] Steve H. Weingart, "Physical Security for the μABYSS System," submitted to *1987 IEEE Symp. on Security and Privacy* (also IBM Research Report Number RC 12344 (Nov. 24, 1986))

[Whit86] Steve R. White, Liam Comerford, "ABYSS: A Trusted Architecture for Software Protection," submitted to *1987 IEEE Symp. on Security and Privacy* (also IBM Research Report Number RC 12343 (Nov. 24, 1986))

Appendix A. Usage scenarios

Scenarios are presented here to help clarify how the PC software asset protection system might be used. Each scenario contains a set of assumptions, usually about implementation, distribution technique, or license agreement. Then, the actions that make up the scenario are presented.

Computer store purchase

This scenario is closest to the way people buy software today.

Assumptions

The PC software asset protection system uses a "token" for one-time authorization of rights to execute. The distribution method used is untargeted software and untargeted right to execute distribution. The end user has a PC with a hard disk, diskette drive, and an attached protected processor.

Actions

1. Alice goes to her neighborhood computer store to buy a software product off of the shelf.
2. Alice pays for the product and brings it home.
3. Alice opens the package. The package contains a distribution diskette, a token, and documentation.
4. Alice makes a backup copy of the distribution diskette.
5. Alice, following the installation instructions, inserts the distribution diskette, the token, and types INSTALL.
6. When Alice wants to execute the program, she just types the program name.

Broadcast software

This scenario illustrates a possible future distribution technique.

Assumptions

A large company distributes its entire software product line each day by satellite. A toll free telephone number is supplied to obtain the right to execute. The toll free number allows a computer connection for ordering software. The software vendor's central computer has protected processor functions. The software is paid for by credit card. The end user's credit is good.

Actions

1. Bob obtains the latest issue of all the software vendors products, via satellite receiver or cable TV.
2. Bob wants to purchase one of the products.
3. Bob calls the toll free number and connects his personal computer to the software vendor's computer.
4. Bob enters the product name and his credit card number
5. The protected processor attached to Bob's personal computer establishes a secure protected processor to protected processor communications channel to the vendor's computer.
6. The software vendor's computer checks Bob's credit.
7. One "right to execute" is transferred to Bob's protected processor.
8. Protected processor to protected processor communications is terminated.
9. Bob can now execute the program.

Transfer ownership

This scenario illustrates how the ownership of the product might be transferred.

Assumptions

The license agreement allows the software product to be transferred. Both the current and new owner have protected processors. Protected processor to protected processor communications is used to make the transfer (a token could also be used).

Actions

1. Alice brings her protected processor to Bob's house to receive ownership of the product.
2. Alice's protected processor is connected to Bob's protected processor.
3. Secure protected processor to protected processor communications is established.
4. Bob instructs his protected processor to transfer the execution rights for the program to Alice's protected processor.
5. The "right to execute" is transferred to Alice's protected processor and removed from Bob's protected processor.
6. Bob gives Alice a copy of the software.
7. Alice can now execute the software, Bob cannot.

Office and home use

This scenario illustrates how a business software package can be used in the office and at home.

Assumptions

A portable ABYSS processor is used. The license makes provision for use of the software in the office and at home, but not at the same time. The software is installed at work. The protected processor contains the "right to execute" for the software product.

Actions

1. Bob makes a copy of the software and brings it home.
2. When Bob goes home at night, he brings his protected processor with him (it is the size of a small pocket calculator).
3. At home Bob attaches his protected processor to his personal computer and executes the program.
4. The next morning Bob goes to work with his protected processor.
5. The protected processor is attached to his work computer and he executes the program.

Protected processor hardware failure

This scenario illustrates what happens if a protected processor breaks.

Assumptions

There are two scenarios presented here. The first assumes the user has a backup protected processor when the one being used fails. The second scenario assumes the user does not have a backup protected processor. In each case the protected processor manufacturer allows a protected processor backup to be used for sixty days Protected processors are sold by the local computer store. The software developers allow their software execution rights to be backed up by the protected processor "create backup" service. A "smart card" is used for the authorization process.

Actions with a backup

These actions assume Alice owns a second protected processor that can be used as a backup.

1. Alice creates a "backup set" onto diskette and "smart card". This backup set contains all the information needed to backup her primary protected processor. The "create backup" service is used to create the backup set. The backup set is targeted to her second protected processor.
2. The primary protected processor fails.
3. Alice uses the "install backup" service to load execution rights into her backup protected processor. The information in the "smart card" is verified and discharged. These execution rights have a time limit of sixty days.
4. Alice can immediately execute her programs using her backup protected processor.
5. Sometime within the next twenty days Alice must visit the local computer store and bring her failed protected processor, the backup set, and her working protected processor.
6. The computer store verifies the protected processor has failed and that the backup set is for that protected processor.
7. The computer store establishes a secure protected processor to protected processor communication between the store's protected processor and Alice's working protected processor. The time limit of sixty days is removed from Alice's working protected processor.
8. The computer store records the service, including the unique serial number of the failed protected processor, and will notify the protected processor manufacturer of the failure.
9. Alice can obtain another protected processor to act as a potential future backup. The failed protected processor may have been under a guarantee from the manufacturer allowing Alice to receive a replacement for free.

Actions without a backup

These actions assume that Alice does not own a second protected processor that can be used as a backup.

1. Alice creates a "backup set" onto diskette and "smart card". The backup set is not targeted to any protected processor.
2. The primary protected processor fails.
3. Alice must go to the computer store with her failed protected processor and backup set.
4. The computer store verifies the protected processor has failed and that the backup set is for that protected processor. The "smart card" information is verified and discharged.
5. The computer store provides a new protected processor (possibly under guarantee).
6. The computer store establishes a secure protected processor to protected processor communication between the store's protected processor and Alice's new protected processor. The execution rights are transferred from the backup set to the new protected processor without a time limit.
7. The computer store records the service, including the unique serial number of the failed protected processor and will notify the protected processor manufacturer of the failure.
8. Alice may obtain another protected processor to act as a potential future backup.

Demonstration software

This scenario shows how demonstration software can be used as a marketing technique.

Assumptions

A software distributor distributes a full function demonstration copy of a new product as part of a sales campaign. Potential customers may use the demonstration package for one week without restriction. Demonstration packages are available at the local computer store. The new software

package is very user friendly, containing on-line help, tutorials, and machine readable documentation. The package is to sell for about $500.

Actions

1. Bob goes to the computer store and obtains a demonstration package. This computer store requires he leave a $5 deposit for the diskette.
2. Bob takes the demonstration package home. The package contains a token and a diskette.
3. Bob makes a backup copy of the diskette.
4. Bob inserts the diskette and the token into the system and types **INSTALL**.
5. Bob can now use the full function of the new product for one week. At the end of the one week the execution rights for the product expire.
6. Bob can purchase a full right to execute for the product at any time.
7. Bob need not return the demonstration package to the computer store unless he wants his $5 deposit back.

Software guarantee

This scenario shows a marketing technique that may be used to guarantee software.

Assumptions

The local computer store had a policy that all sales were final. The only returns that would be permitted were returns of packages that were not opened. Now that there is an effective PC software asset protection system, the policy has changed. Anyone who returns all materials, including the "right to execute" within one month of purchase can get a full refund[24]. A transfer set is used to transfer the execution rights. Protected processor to protected processor communications could have also been used.

Actions

1. Alice purchases the software from the local computer store.
2. Alice installs the software on her system (see "Computer store purchase" on page 26).
3. After using the software in her own environment, on her own data, Alice decides she does not want the software.
4. Alice creates a transfer set for the software package. The transfer set contains the "right to execute". The right to execute no longer exists in Alice's protected processor.
5. Alice brings the package and transfer set back to the computer store.
6. The computer store verifies the "right to execute" on the transfer set and gives Alice a refund.

Rental software

This scenario shows how software can be rented.

Assumptions

The computer store wishes to rent software in much the same way video cassettes are rented.

Actions

1. Bob goes to the computer software rental store to rent, for one month, a word processor to do a term paper for school.
2. The computer store prepares a package with a right to execute that is limited for one month.

[24] If manuals and diskettes are returned in other than perfect condition, less than a "full" refund may be given.

3. Bob takes the software home and uses it to do his term paper.
4. Bob returns the software to the computer store. The right to execute does not have to be returned because it expires at the end of one month.

Portable rights

This scenario shows how a user can carry her execution rights with her wherever she goes.

Assumptions

The software license allows the software to be executed on any personal computer but only one at a time. All personal computers have a protected processor built in and a "smart card" reader. The "smart card" contains the protected processor (without the ability to execute applications) and is used to carry execution rights.

Actions

1. Alice purchases the software at her local computer store. The "right to execute" is loaded into her portable protected processor (smart card).
2. Alice goes to any personal computer, inserts her credit card sized protected processor into the smart card reader and a copy of the distribution diskette into the diskette drive.
3. Alice executes the program.

A pirating attempt

Assumptions

A potential pirate owns a protected processor and wishes to obtain cryptographic keys from the protected processor.

Actions

1. The pirate attempts to physically enter the protected processor.
2. The protected processor tamper detection circuitry detects the intrusion.
3. The tamper detection circuitry removes power from the volatile memory that contains the key store and the application data store.
4. The pirate cannot get any useful information from the protected processor. All execution rights and cryptographic information are lost.

Appendix B. Glossary.

This glossary defines terms and acronyms that may be new or used in a special way in this paper.

ABYSS: Stands for *A Basic Yorktown Security System*. An IBM Research Division project to address the problem of protecting software assets.

Application data store: A logically and physically secure portion of memory used by applications and the software asset protection system to store information between executions of the application. The application data store is contained within the protected processor.

Application key: A cryptographic key supplied by the software developer that is used to encrypt the protected part of the software. This application key also becomes the "right to execute" for the software package.

Authorization process: The authorization process can securely limit the number of times an action takes place. The authorization process can use a "token".

Backup set: A set of information on any media (usually a diskette and token) which is used to back up (hold a copy of) sensitive information from within the protected processor in case the protected processor becomes unusable. Sensitive information is encrypted in the "backup set". Backup sets are created and used only by trusted protected processors. An authorization process authorizes future backups or restoration of the backup set.

Copy protection: Any mechanism designed to limit the number of functional copies of a software product. Usually the number of copies is limited to zero or one.

Distribution set: The original package that is obtained when a software product is initially purchased by an end user. This package usually contains items such as distribution diskettes and documentation. For products protected by the PC software asset protection system the distribution set contains the unprotected software, protected software (encrypted), "right to execute" or application key (encrypted), and possibly a token.

Open system: A system in which all interfaces, hardware and software specifications are logically and physically accessible and, therefore any hardware or software component of the system can be examined, changed or extended by the end user.

PC: Personal Computer - any member of the IBM personal computer product line or compatible system.

Protected processor: A logically and physically secure processor. The protected processor constitutes the closed portion of the computer system in which sensitive data and programs reside. Its memory holds cryptographic keys, sensitive application data, the protected part of application programs and supervisor programs. The secure processor executes supervisor programs and the protected parts of applications.

Right to execute: The authorization for an end user to execute an application or software package.

ROM: Read only memory. A type of computer memory that normally contains permanent programs or data. This memory cannot be modified by programs executing in the computer system.

Service: A function or service that is part of the PC software asset protection architecture. Also referred to as a "supervisor service".

Software piracy: The illegal distribution of licensed software. This is commonly accomplished by the "software pirate" making a copy of the distribution diskette (or backup diskette) and giving or selling it to unauthorized users.

Supervisor key: A cryptographic key placed in the protected processor by the manufacturer of the protected processor. Supervisor keys are used to uniquely identify the protected processor, establish a secure channel for communication with the outside world, and encrypt sensitive information.

Supervisor process: The protected processor state in which supervisor programs are executed. All of the memory in the protected processor is accessible when the protected processor is in the "supervisor process" state.

Supervisor program: A trusted program that executes in the supervisor process in the protected processor. The supervisor program is usually supplied by the protected processor manufacturer. PC software asset protection services are implemented by supervisor programs.

Token: The "authorization process" can be implemented with the use of a physical device that attaches to the protected processor when the authorization process takes place. The token contains the authorization for the service that is being executed and controlled by the authorization process. Like a theater ticket which is destroyed when it is used to enter the theater, it cannot be used again.

Token descriptor: A description of the contents of a token. The token descriptor identifies the sensitive resource that the token is protecting. The token descriptor is used to verify the validity of a token and to make sure the action that is being authorized is acting on the correct resource.

Transfer set: A package that is created when ownership of a software product is to be transferred. The transfer set contains the same type of information that is contained in the original distribution set.

Index

T

U