
LINK
Computer
Tuborg
ØL

ALL ABOUT COPY PROTECTION

Copy protection hasn't always had a good reputation. Up until a few years ago, it was regarded as an added expense for the software distributor and an aggravation for the legal end-user. At best, you didn't notice it. At worst, it could crash hard disks or simply refuse to run. People hated it on general principle. Still, it was necessary. Software houses lost and are still losing huge sums annually to piracy.

Although many are unaware of the fact, copy protection has quietly gotten over its childhood illnesses. What is more important, it has changed character. Copy protection in the old days was merely a technique for preventing software theft. Today, it is an integral part of the distribution of executable software as well as data, books, encylopedia and other text objects.

To a large degree, this is due to the advent of the CD-ROM which makes it possible to distribute large amount of data at low cost. Other methods of mass distribution such as electronic bulletin boards and world-wide networks such as Internet also play a part and will come to play an even larger part.

The idea now is not to limit the copying of data/programs. On the contrary, the idea is to copy them as widely as possible so that end-users can see them and try them out. However, you want to be in control of access to this data. You want to be able to supply the end-user, via telephone or network, with an access code which allows him to use the program or data on a permanent or time-limited basis. This is the future of copy protection

From a purely technical angle, copy protection is a fascinating subject. Few other topics require such an intimate knowledge of hardware and system software.

DIGITAL AND ANALOG DATA

A floppy disk with digital data on it is fundamentally different from a music cassette containing analog information, though both rely on magnetic media. A copy made of a diskette is exactly as good as, or possibly better than, the original. This is one of the great strengths of digital information. However, for the software manufacturer and distributor, it means that anyone with one of his distribution diskettes can easily make as many copies as he likes, and the quality of the copy will be 100% as good as the original. This fact is the driving force behind copy protection.

Digital information can be roughly categorized as code and data. Of course, both are data in the general sense, but computer programmers differentiate: data that can be run by a processor is called code, while data that is merely information is called data. In many cases, the executable code, so-called EXE files, must be protected. In other instances, pure data files need to be protected, but this requires a somewhat different technique: the data itself is encrypted, while the text retrieval utiltiy, the executable, is copy protected.

WHAT IS COPY PROTECTION AND HOW DOES IT WORK?

Since the binary information can always be copied, the manufacturer must make the operation of his program dependent on the presence of some physical key which cannot be copied. He does this by changing his software so that it cannot run unless some sort of initialization is performed. He then adds some additional program code, called the guard module, which carries out the necessary initialization if and only if it detects the presence of the physical key. This is copy protection in a nutshell:

(1) a modification to the original code to make it dependent on some external action
(2) a guard module to provide the necessary action when it detects the key, and
(3) the key itself.

All copy protection schemes are alike in that they must have the aforementioned three parts in order to work. If any part is missing, the scheme fails. Let's look more closely at these three parts:

Dependency on External Action

The original software must be changed in some way so that it will not run without the action of the guard module. This could consist of merely including calls to the guard module in the software. However, the best way of making the software initialization dependent is to encrypt it. Encryption means taking the code and scrambling it so that it cannot run and is no longer recognizable. The same techniques can be applied to computer programs as to secret messages.

The Guard Module.

This is the code that restores the software to executable form or in some other way initializes the software and allows it to run. It must do this only when the key is present. When the guard module is satisfied that the key is authentic, it initializes the software and executes it. Besides the function of recognizing the key and restoring the software to executable form, the guard module must do its job in complete secrecy. It must be impossible to see what it does, impossible to imitate what it does and impossible to trick it into doing its job when the key is not really present. This is called code security. Unless, the guard module itself is protected in some way, usually by encryption and debug-trapping, the protection can be disabled, and the software made to run without the key.

The Physical Key

This is the actual physical device or object that must be present as proof of ownership and the right to use the protected program. The key can

take on many forms, e.g. a key diskette, a dongle or a "smart card".

MAKING THE SOFTWARE DEPENDENT ON THE GUARD MODULE

There are various ways to do this, though techniques which do not include encryption cannot be considered very safe.

Building in Calls to the Key Check

The software can be made dependent on the guard module by building in calls to the key check in various places. To discourage the discovery and disabling of these calls, they are usually hidden throughout the program. Sometimes an attempt is made to confuse hackers by allowing the calls to jump to other areas in the program instead of simply returning. The problem here is that the programmer can often become as confused as the hacker.

Encryption

Encryption of the main EXE file is one of the best methods, since it protects the software in three ways: First, it makes the program impossible to run. Second, it makes it impossible to see what the program is doing and prevents stealing the know-how employed. Third, it makes it impossible to find and disable calls to the key check.

It is relatively simple to encrypt a program but difficult to decrypt one without knowledge of the algorithm. If the encryption key (not to be confused with the hardware key) is long enough, the number of computer-hours necessary to crack the encryption makes the job practically impossible. The real problem is hiding the key. Quite unlike the transmission of secret messages where the key is sent separately, the key for software decryption must be found either within the code itself or perhaps within the hardware key, and this makes it accessible to the hacker. Remember too, that even though the routine which does the decryption (the guard module) is itself encrypted, it must begin with un-encrypted code, and this gives the hacker a place to start. He can trace through the decryption code and discover how it is done, or even put a break-point after the decryption, stopping the execution when the code is wide open. Debug-trapping and other more or less successful schemes have been devised to prevent this.

THE GUARD MODULE FUNCTIONS

Key Detection.

This is the part of the code that checks to see if the key is present. It is a requirement that the operation is as low-level as possible (i.e. direct hardware access of I/O ports), thereby preventing any sort of intervention. Note that this can give hardware compatibility problems.

Initialization.

This is the part of the code that does the necessary initialization of the software (decryption, re-location, etc.). Without this initialization, the software will not run.

Code Security

The guard module itself should also be encrypted (not to be confused with encryption of the user program itself). This is done to prevent disassembly of the guard module. If a hacker can understand how the initialization is done, he can throw away the guard module and do the initialization himself.

Debug Trapping

Even though a hacker may not be able to understand the workings of the guard module, if he can debug (trace) his way through it, or place a break-point after it has done its work, he can gain access to the initialized software which he can then save and use as a directly executable file.

The guard module itself can never be totally encrypted. Somewhere, there must be a starting point. This code must be open and accessible, otherwise the CPU can't run it. Some technique must be used to stop the hacker from stepping through the code and finding out how the decryption takes place. One way to do this is to take over the debug and single-step interrupts and use them as part of the decryption. If a hacker then tries to use a debugger, he spoils the routine. The hacker can counter this by designing a debugger which uses another interrupt number. This, of course, requires a degree of programming skill, and even this ploy can be foiled by checking code integrity and thereby preventing the placement of break-points.

VARIOUS KINDS OF PHYSICAL KEYS

The key must be a physical object which is impossible or at least difficult to copy. No matter which type of key is used, the effectiveness of the protection is always dependent on the quality of the code security. If the call to the key can be disabled, then it doesn't matter whether it can be copied or not.

The Key Diskette

This type of protection relies on a specially produced key diskette which has something done to it to make it difficult to copy. Some primitive types of key diskettes employ extra sectors having unusual sizes and ID numbers. Others rely on hidden files. Good key diskettes base their security on the measurement of parameters which are nearly impossible to reproduce with any degree of accuracy. One way of making a key diskette impossible to copy is to physically bore a small hole in the magnetic media. This creates one or more bad sectors having bit patterns which cannot be overwritten. Manufacturers can also produce their key diskettes on special equipment which can do things that a PC cannot, thereby

creating a diskette that can't be copied using ordinary controllers and drives.

The Dongle

Here, the key is an electronic circuit, usually employing a custom chip of some kind, which responds in a particular way to an electrical stimulus. One common type connects to the parallel printer port. When a query is sent to the printer port, the dongle replies according to a complex pattern which is difficult to predict if you don't know the algorithm. Dongles get high scores for non-copyability. One of their weaknesses is that you can run into trouble with several dongles on the same port.

The Hard Disk

Some schemes depend on special marks or deliberately-produced bad sectors on the hard disk. This is a common method of getting away from external keys such as diskettes or dongles, but has the great disadvantage that it can wreak havoc when the hard disk is backed up and restored.

Other Hardware

Other hardware devices can also be used, for example a plug-in board. A program such as an EPROM burner would rarely be copied, since it needs the plug-in board to function, but it can still be a good idea to lock the software to the board to prevent copying of the entire hardware/software package.

A "smart card" consisting of a chip mounted on a plastic card is another example. Of course, this requires a special card reader, but so-called "thin drives" are becoming more popular, making this an interesting possibility.

Even the computer itself can be used as the key. In this case, the protected program is said to be machine-installed.

Personal Characteristics

It is also possible to use human characteristics such as finger prints, voice prints or retinal images, though these require special sensors which are not generally found on personal computers.

PRODUCTION: ADDING THE GUARD MODULE

During production, the three elements of the protection scheme are brought together. This can be carried out, for example, by a separate utility that encrypts the bare EXE file, adds on the guard module and creates the new protected EXE file.

Sometimes, the guard module is not a separate module, but consists of scattered calls to the key check which are built into the EXE file. In this case, the programmer of the original software includes the protection as part of his source code.

Internal Protection

The guard module can be programmed into the source code. In this case, the call which verifies the key is put in by the original programmer. He can try to confuse the enemy by placing calls in many areas of the code. If this is done without sophisticated encryption techniques, it is just a matter of hard work for the hacker to find the calls and disable them. The programmer can, of course, work out an encryption method, but often he does not want to be bothered nor does he have the necessary expertise. So the protection is usually added on afterwards by some kind of commercially available package.

Add-On Protection

The guard module can be added to the executable file after it has been compiled. This has the advantage that the programmer need not concern himself with copy protection when building his program, but can concentrate on making the best possible software. The protection is added on afterwards by a professional protection package. Many distributors do not have access to the source code and can therefore use only this type of protection.

Link-In Protection

It is also possible to use a combination of the above approaches. The protection manufacturer can supply the software house with object modules which can then be linked together with the package. In this way, the programmer need not design the protection, but has the flexibility of being able to put in as many key checks as he likes, wherever he wants them. However, this technique still requires access to the source code and object modules.

BREAKING THE PROTECTION.

There are many ways of cracking a protection scheme, and the results have varying degrees of consequence for the distributor.

Reverse Engineering

By disassembling the program and finding out how it works, another programmer might use the principles involved to write his own program without actually copying the program itself. This process is time consuming and difficult.

Creating a "Cracked Copy"

If the protection can be disabled, the software will run without the key. The idea is to either peel off the guard module or find some other way to defeat it, so that the protection check is never made, but the initialization is made anyway.

Copying the key

If this can be done easily, it is almost as good as creating a cracked version. Dongles and diskettes with physical holes are extremely difficult to copy; for all practical purposes they cannot be copied. Key diskettes made from

normal diskettes can or cannot be copied depending on the degree of sophistication of the diskette and of the equipment attempting to copy it. Commercially available copying boards can defeat many protection schemes. For the serious pirate, there is the synchronized bit copier which moves every bit directly from one diskette to another, using electronically synchronized drives. This can copy all but the very best key diskettes.

Assume, for example, that the key is a diskette that can be copied using a board costing less than $200. The end-user would have to buy and install the board and learn to use the accompanying software. This gives some degree of protection - enough to discourage the casual copier, but not a dedicated crook. The distributor must evaluate his needs and choose a system that fits.

Fooling the Protection

A memory-resident program can be installed which makes the guard module "think" the key is present. If, for example, the guard module checks the key diskette or dongle by way of the system BIOS, a filter can be set up to watch the interrupt and intervene when the key check is made, feeding false input to the guard module and simulating the expected signals.

HOW MUCH PROTECTION IS NEEDED?

Any scheme can be defeated by the use of expensive data analyzers, computer time and hard work. Therefore, all protection must be evaluated on the basis of how much time and money a pirate is willing to invest in order to crack the protection.

There's little sense in heavily protecting a program that can be written from scratch for less than the cost of breaking it. One might think that an expensive program would need better protection than a cheap one. It ain't necessarily so. The expensive program might only be of interest to reputable customers who wouldn't dream of cheating. It also depends on the part of the world in which the software is to be distributed. Some countries have no legal basis for prosecuting pirates. In any case, you have to know your market.

Remember, too, that some enthusiastic young hackers break programs just as a challenge. They are not really interested in selling pirate copies in large quantities, although a distributor can risk finding his program on an electronic bulletin board.

WHAT DOES THE PROTECTION COST

If the program being distributed is expensive, then the cost of the individual protection is not a determining factor. Dongles or diskettes with physical holes can be used in this case. If the price per protection is important, then a better choice would be one of the systems which lets you produce your own key diskettes. Remember that the best key is useless unless the quality of the code security is equally high.

OTHER WAYS OF COMBATTING PIRACY

There are other things a manufacturer can do to discourage unlawful copying without actually using copy protection.

The Manual

He can make his software dependent on the manual. Manuals are more expensive and harder to copy than diskettes. One could argue that this is a bad idea, because the software is poorly designed if it cannot be understood and used without reading the manual. Some manufacturers have their programs ask for a piece of information on a particular page in the manual. The page number changes each time the program is run.

Support

Constant improvement and updating of the software discourages copying, or rather encourages the purchase of legal versions, since pirates are stuck with the version they have.

Good telephone support does the same, since owners of pirate copies can't call in for help.

Providing legal users with newsletters, bulletin board access, and other services also encourages people to pay for the program.

Display of User Name

Equipping the software with an encoded user name which is displayed at run time is a good technique. While it's easy to make a copy and give it to another user, it is embarrassing to run a program with someone else's name on it. This type of copy protection actually comes under the heading of "copy discouragement"; it has the great advantage, though, of not creating pesky hardware compatibility problems. If the original program runs on a particular machine, the "protected" program will run too.

About the author.

Roger Mester was educated in the USA at Rensselaer Polytechnic Institute and Stanford University. He has worked for General Electric, Lockheed, Radiometer, and GNT. He was one of the two founders of Link Computer - now Link Data Security - in 1982.

Document DialSem version 1.00

LINK Data Security A/S

05/12/94

DialCops Sales Information:

1. General

DialCops is a licensing system allowing a very convenient and safe distribution of scrambled (encrypted or disabled) data or programs. Parts of the data can be open (enabled) as an appetizer or it can stay open until a specified date.

DialCops is unique in that it restricts the allowed use to a single machine or group of users by uniquely identifying the inner signature of the computers.

Based on the Cop's Copylock security products, DialCops has an outstanding level of codesecurity and field tested encryption algorithms.

1.1 Definition of terms:

MachineID: A unique identification of the installation - either the machine reference measurement or the serial number (if a key-diskette version). For the Network version the super-user MachineID will be used all over.

DataSet: An individually accessed block of data - for instance one book or one program.

DataSetID: A unique identification of the DataSet - for instance, the ISBN number for books.

AccessCode: The code that can open up (enable) a DataSet.

ApprovedList: This list contains all the approved DataSetIDs and their corresponding AccessCodes.

Publisher: The organization which distributes the DataSets and owns the protection software.

SubPublisher: An organization which distributes DataSets via the Publisher and which is entitled to generate and sell AccessCodes to their DataSets.

1.2 About this paper

This paper outlines the principles in DialCops dial-up access control of individual DataSets, primarily targeted at CD-Rom distribution.

Two different approaches are covered - one for companies that want to control their ApprovedList and integrate it into their own application and one for companies that want a pretty much standard solution. They can use our skeleton DLL with its built-in ApprovedList handling.

1.3 Overall description:

The security routines mentioned here constitute the security kernel in a system, where a Publisher can control who get access to what where. For program protection no changes have to be made. For data protection a set of routines must be integrated into the data retrieval system, and a small database must be maintained with information about which machines can access the various DataSets. The database can be maintained by the end-user remotely

controlled by the Publisher over the telephone. It can also be distributed in a finished form, at least to start up with.

Normally DataSets are closed (disabled) until a machine-specific AccessCode is bought, but certain DataSets can be permanently open on any machine or they can be configured to demand a key-diskette before being enabled.

All DataSets must be encrypted before distribution. The distribution key can be chosen to be individual for each DataSet.

The end-user will report a MachineID and a DataSetID over the telephone and receive an AccessCode. This AccessCode contains a decryption key identical to the key used to perform encryption of the DataSet.

The system can be enhanced to include time expiry as part of the Access-Code. This AccessCode will not be valid after a date that was chosen when the AccessCode was created. The combination of time expiry and choosing the DataSets to be enabled on any machine is a powerful demo feature.

Security related functions should be placed in a .DLL file together with dummy entries matching the routines in section 2. The compiled DLL must then be protected with DIALCOPS.

Inside the DLL the functions OpenCode and CloseCode can be used to mark up to 32 selected areas as being closed. They will then be encrypted by the DialCops protection utility, and they will be opened up when execution arrives at OpenCode. Debuggers will be thrown off at the same time. When the CloseCode function is reached, the whole area is closed again. Code security will thus be very high, effectively keeping these areas as secret.

2. The protection module.

The DLL code segment containing the initialization entry point will be totally encrypted (besides OpenCode/CloseCode areas being scrambled).

During initialization of the DLL, this segment will be decrypted - but only if the protection is found to be OK. At the same time a machine measurement is made and it is compared to the MachineID stored in the .DLL.

If the DLL is run for the first time, the MachineID is stored and the protection will return OK.

If these two values are nearly identical (within the tolerances), the .DLL is decrypted and other functions can then be called.

2.1 NetCops protection module:

If the .DLL is network protected, a small DOS program protected by NetCops is called. It returns a machine measurement from the super-users machine. This is compared in the DLL to the stored MachineID, or, if not yet installed, it is stored as the MachineID.

3. The programmers approach

Companies that wants to handle their own ApprovedList will only use a few basic functions.

If the verification fails, they should show the DataSetID from their own ApprovedList, and the returned MachineID. Then their customer could call up the Publisher, and buy an AccessCode.

Then they simply store the AccessCode in their own ApprovedList, and from now the DataSet will be enabled on this machine.

The descrambling routine or the display routine used later on should at least partially reside in the DLL and make calls to the basic functions to ensure that the DataSet was correctly verified and to get the decryption key.

All the .DLL functions/procedures can be encapsulated by an OpenCode and CloseCode procedure for maximum code security .

4. The general approach

Using the built in functions no handling of the ApprovedList is necessary.

When an attempt is made to open a DataSetID either it succeeds or the returned MachineID can be reported back to the Publisher.

The user then enters the received AccessCode, and calls a function that puts this entry into the ApprovedList.

When data needs to be descrambled, a function is called. Internal mechanics ensures that only valid Data can be descrambled.

If the developer wishes to improve the security, he can use the Opencode/CloseCode functions provided by LINK.

5. Unprotected DataSets

Most Publishers would like to include books/DataSets that are already open for access. To allow this we have defined a broadcast MachineID value that will create AccessCodes that can be used on any machine.

Using the general algorithm an access code can be calculated for any given DataSetID (ISBN number) using a reserved wildcard MachineID.

In this way a book can be made generally available by broadcasting only one common access code. It is still impossible for users to open other DataSets, since the access code is dependent on the DataSetID.

6. Time expiry

An interesting feature in DialCops is time expiry. When an AccessCode is made by the Publisher (or SubPublisher), a certain expiry date can be chosen. One month before this date, it is possible to warn the user.

Expiry will always take place at the beginning of a new month. Thus if the warning comes for instance in May, expiry will take place 1st of June.

It is not possible to fool this system just by faking the PC system time - not even at the CMOS level.

Time expiry is an efficient sales and demonstration tool. Combined with an AccessCode that matches any machine, a time expiry protected demo version can be distributed on, for instance, a CD-ROM.

Please notice that this demo facility only requires 1 (one) AccessCode.
Demo versions can thus be sent out free of royalties.

7. The AccessCode generator

In order for the Publisher (or SubPublisher) to generate AccessCodes, LINK delivers a codegenerator program. This program is a DOS program which can run in a Windows Dosbox. It can easily be interfaced to existing administrative programs. Basically the end-user inputs, MachineID and DataSetID are combined with a decryption key (typically looked up in a table from the DataSetID) and an optional expiry date. The output of the generator is an 8-digit alphanumerical AccessCode which is saved in a file on the harddisk and shown on the screen.

The codegeneratorer also makes log files for two different purposes:

1) In-house activity reports and SubPublisher/Publisher relations.

2) A specification used for royalty payment to LINK or to the Publisher.
The DialCops system is well suited for voice response systems, since only digits are used from the end-user.

Once every six months the AccessCode generator must be renewed from LINK and the log contents will be the basis for this.

8. SubPublisher Setup

A very common setup involves a Publisher, who on behalf of several SubPublishers is responsible for making a common CD-ROM. The Publisher can be in charge of distribution of AccessCodes but he can also choose to allow SubPublishers to make AccessCodes to their individual DataSets. Since the DataSets are encrypted with a chosen key, the SubPublishers can be given an AccessCode generator with a table of allowed DataSetID's and their corresponding encryption keys.

It is also possible to let the SubPublisher be responsible for encrypting the DataSet, in which case he will be the only one to know the key.

The SubPublishers AccessCode generator must be renewed every six months, and to do this he must report to the Publisher an extract from the log created by the SubPublishers AccessCode generator. The Publisher's AccessCode generator is capable of issuing such a renewal code, at the same time incorporating the reported figures into the Publishers main log.

9. Pricing:

The starting fee for DialCops is $ 1950. On top of this comes a royalty per AccessCode obtained. This fee is $ 2.00 per AccessCode. In volume (more than 2,000 AccessCodes) this fee is reduced and for very high volumes (more than 10,000 AccessCodes) it stabilizes at $1.00. The count is cumulative from day one.

A minimum count will be agreed to

The royalty fee is paid every 6 month based on the previous periods count.

LINK Data Security A/S
Vesterbrogade 51
1620 Copenhagen, Denmark

Tel + 45 3123 2350
Fax + 45 3123 8448
BBS + 45 3123 2384

LINK Computer

Vesterbrogade 51
1620 Copenhagen V
Denmark
Telephone: +45 123 2350
Telefax: +45 123 8448