Software Authorization Systems

Paul A. Suhler, Nader Bagherzadeh, Miroslaw Malek, and Neil Iscoe
University of Texas at Austin

For every piece of business software sold, at least one illegal copy exists. This article describes and classifies software-protection methods.

It has been estimated that for every piece of business software sold for a microcomputer, at least one other copy was obtained illegally.[1] Software vendors and industry associations perceive the software pirate as a major threat to the production of quality software and are making efforts to deter piracy. These efforts are educational (telling people it is wrong and harmful not to pay for software), legal (taking unauthorized distributors and users to court), and technical (implementing programs and devices to prevent unauthorized use).

Software theft environment. Technical protection of software has gained importance as the installed base of microcomputers has increased. Although protection schemes do exist for mini and mainframe computer programs, the distribution method for this software generally requires that license agreements be signed before the software is transferred to the user. Furthermore, mainframe software usually requires a lot of support, updates, and other items that require users to stay in contact with the vendor. Consequently, theft in the mini and mainframe marketplace is not considered a problem that requires a technical solution.

However, the high-volume, mass-market nature of microcomputer software distribution has resulted in a substantial amount of theft.

The first players in the software game were the software vendors, the distributors, and the users. The vendors wrote software and sold it, sometimes through distributors and sometimes directly, to the users. However, pirates soon came on the scene: either companies intent on mass-producing and selling software without permission or users who copy and distribute programs to other (nonpaying) users, either directly or indirectly via network bulletin boards. Figure 1 shows the flow of legal and illegal copies of software.

The motivations for pirates vary. Commercial pirates, who make a profit from

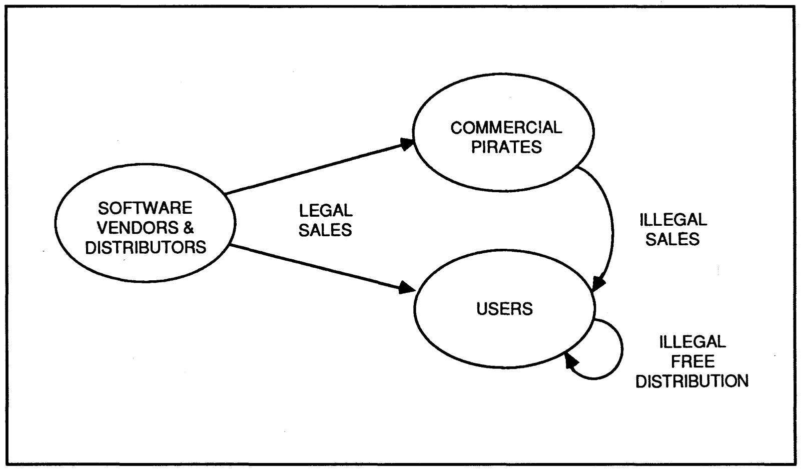

Figure 1. Software distribution.

selling a program without investing in it, are vulnerable to legal action (although enforcement is more difficult in the case of foreign-based pirates). The noncommercial pirate has different motivations, such as getting a program for free or evaluating an expensive piece of software without paying the full price.[2] Because they are so inconspicuous, these pirates have been difficult to detect and stop.

The result has been a self-perpetuating system of theft and high software prices. Piracy reduces the sales of any particular program, so the vendors must maintain high prices to recoup their investment on fewer sales. The higher price is then a greater motivation to obtain a pirated copy, which reduces sales even further.

The problem can reach the point where software developers are less willing to invest the time to produce sophisticated programs because they fear they will never make a profit. The losers are both the vendors who lose profits and the users who are never offered the improved programs.

In response to the need felt by many software vendors to protect themselves from pirates, a new player has entered the game: the protection vendor, who offers technical solutions to prevent unauthorized use. Their systems are sold to software vendors who incorporate them in their products. Some software vendors have their own proprietary protection schemes.

Authorization system goals. The goals of software authorization systems are simple: to prevent the unauthorized copying or execution of software that has been legally sold, or to make the piracy at least as difficult as writing the software from scratch. In doing this, software vendors must view the customer not only as a potential pirate but also as a friendly customer who may purchase more software.

Thus, a software authorization system must meet three criteria: (1) It must be inexpensive and easy to use. (2) It must be compatible with existing unprotected programs and with programs protected by other authorization systems. (3) It must be easy for the software vendor to incorporate in his production and distribution system.

Software protection history. A report[3] written by the Software Protection Committee of the Association of Data Processing Service Organizations (ADAPSO) presents a fairly comprehensive history of personal computer software protection efforts.

The report says that an application program for Apple computers, VisiCalc, was the first software to be protected against unauthorized copying. Since then, there have been several variations on this scheme, which we will discuss in detail.

In 1980, Business Professional Industrial used a hardware device to protect its accounting package for the Apple II. This scheme required the user to open the computer and insert the security-key device in the game-paddle port.

Although it provided sophisticated protection, this scheme prevented the user from using the paddles without removing the device. The problem proved to be more than a mere inconvenience, as reports came in of pins bent by repeated insertion and removal of the protection device. In early 1982, the company discontinued the system and eventually resorted to disk-based protection methods.

Software authorization has three criteria: to be inexpensive, compatible with other systems, and easy to implement.

Sensor-Based Systems also used a hardware-based method, for their Metafile product on a Vector Graphics computer, that required the installation of a PROM chip in the system. Again, this functioned well but was an inconvenience. For the IBM PC version of Metafile, the company used the PROM and Xedex's Baby Blue Card.

A major drawback was the lack of portability, so in August 1982, Sensor-Based Systems introduced an 8086-based version of Metafile for the IBM PC. This version used a special protection device that was attached to the RS-232 port.

These examples illustrate the evolution of systems for what has become known as software authorization: preventing unauthorized copying and unauthorized execution of programs. The systems have required special devices: either special program disks or extra hardware.

Authorization approaches

To protect software from unauthorized use, there are three basic authorization techniques and several ways of implementing those techniques. The techniques are based on general principles: copy protection, validation, and encryption. The implementations are based on hardware devices: floppy disks, special microprocessors, and devices attached to the system bus or an I/O port. More than one technique can be employed in a single software authorization system, while seldom will more than one implementation method be used.[4]

Authorization techniques. Authorization can rely on either of two basic principles: making it impossible for an unauthorized person to obtain a copy of a program or preventing unauthorized copies from executing on any but the licensed system. The former is done with various techniques called copy protection and the latter is done either by encrypting the program or by checking authorization during execution (validation). Each technique has inherent strengths and weaknesses, as summarized in Table 1.

Copy protection. Copy protection means making the disk on which the program is provided (the master disk) uncopyable by the usual microcomputer operating system utilities. This can be done by inserting dummy file segments having erroneous CRCs or invalid control codes in the program file. These segments, while not normally read in the course of loading and executing the program, will be read by the operating system copy utility, which will not copy the intentionally placed "bad" information. When run, the copied version of the program will look for the missing information and abort execution.

Some versions permit one copy to be made: the copy process modifies the original disk so that it cannot be copied again. VisiCalc for the Apple computer was one of the first commercial programs to be distributed on copy-protected disks.

Software vendors using copy protection

typically provide main and backup disks with their programs and then offer free replacement of worn disks. Copy protection requires no extra devices or operations and does not noticeably slow down execution of the program. It has the advantage of usually being transparent to the user, although this depends on the implementation.

Unfortunately, copy protection is relatively easy to evade. Instead of using the standard copy routine, a pirate can use a bit copier, a program that copies a disk bit for bit without interpreting CRCs, protection bits, or other information for its meaning. Another way around the protection is to execute the program under a debugger, interrupt execution, and copy the memory image of the program to an unprotected disk.

The desire for a quick backup method has prompted the creation of several commercially available programs that copy these protected disks. Such programs are advertised for making only legal backup copies, but they can be abused.

Validation. Protected software can check the user's right to execute it. Validation systems usually feature blocks of code embedded in the program to locate a unique key in the system. If the key is not found, the program is assumed to be on an unlicensed system and execution is aborted.

The key can be a number stored on the program disk (in a location not normally copied), in a read-only register in a hardware device attached to the system bus or an I/O port, or in a detached device that the user must query in response to a program prompt. In a disk-based system, the key can also be data in sectors or tracks written in an unusual location or format. Absence of the unusual format indicates an unauthorized copy.

If the key is located on the disk, the validation is usually invisible to the user, who may or may not notice a degradation in performance. Unfortunately for the vendor, this is no more secure than copy protection, because it can be evaded by the same methods. If the key is in a device, it is much more difficult to copy, but the device presents an inconvenience to the user, who must attach it and who will lose the use of the port for I/O if the device is not transparent. Finally, if the user must manually query a detached key device and enter the resulting key, he will have to tolerate interruptions to his processing.

A variation on validation is used with programs distributed on ROMs that try to determine if the program has been copied into RAM. The validation code uses hardware-dependent operations to determine if the program is being executed from ROM or RAM, and aborts in case of the latter. This method is uncommon, as relatively little software is distributed on ROMs.

In general, validation systems can be defeated by removing the validation code from the program, by duplicating the key, or by simulating its presence. This has led to a war of wits between vendors and pirates.

Using a debugger and a disassembler, pirates can remove validation code or modify it to never abort. But vendors can place validation code in several places in the program, so if the pirate misses even one routine, the program will abort.

Validation code can be written in a Byzantine manner, with its instructions scattered through the program and connected with goto statements, so it will be more difficult to recognize than a cleanly written routine.

Multiple validation routines can also be made not to function every time the program is executed, so a pirated program with some validation routines removed may function for a while and then suddenly fail, causing frustration among the pirate and his customers.

Recent microprocessors with memory protection that requires accessing I/O devices through system calls have been a boon to the pirate, who now must look only at the exception vector to locate the routines communicating with the key device. (This has been named the 286 problem by vendors trying to protect programs for systems such as the IBM PC AT, which use the Intel iAPX 286.) Pirates with access to test equipment can learn the key by monitoring data transfers between the CPU and key device.

Systems with a processor in the key device can have the same algorithm executing in the validation code and the device to generate not just one, but a sequence of keys, which is more difficult to understand. Thus, the battle continues, with vendors trying to push the price of defeating their systems beyond the skill, patience, and resources of most pirates.

Table 1.
Comparision of authorization techniques.

Technique	Advantages	Disadvantages
Copy protection	No additional hardware Can be totally transparent	Can be defeated with bit-copy program or debugger
Validation	Medium security Relatively inexpensive	Can be defeated by patching out validation code Usually requires additional hardware Requires execution time overhead
Encryption	Potentially highest security	Usually requires additional hardware Can require long execution times May require complex distribution system for keys

Encryption. The program can be encrypted so that before being executed it must first be decrypted according to a unique key available only on the licensed system. In this scheme, the buyer provides an encryption key (possibly based on his system's serial number) to the vendor when ordering a program. The vendor in turn produces a version of the program that can be decrypted only according to the key unique to the purchaser's system.

Decryption can be done at load time by the CPU or by a device attached to the bus or an I/O port. In these cases, the key is usually handled as it is in validation systems. The decryption can also be performed at runtime by a cryptoprocessor that replaces the computer's standard microprocessor and decrypts instructions internally during the instruction fetch phase. Only in systems using a cryptoprocessor does the plaintext code never appear in a form readable by the user.

The basic problem with encryption is that the more secure the encryption algorithm (the longer it takes a pirate to break the code without having the key), the more processing is required to perform the decryption even with the key. This can degrade the instruction execution rate significantly.

Furthermore, encryption systems can be defeated in three ways: by determining the key and decrypting the program, by using a debugger to copy the program after it has been decrypted, or by copying the vendor or distributor's master, nonencrypted copy of the program.

Using a hardware device for decryption means the key can be hidden from the user. However, if the same key is used for encryption and decryption, as in the Data Encryption Standard,[5] the key must be available for encoding programs as they are purchased — and is thus susceptible to theft and use in unauthorized decryption.

By using an algorithm such as that of Rivest, Shamir, and Adleman,[6] in which the decryption key cannot be determined from the encryption key, a user can know his encryption key to order new software without knowing his decryption key. However, key distribution can be an administrative problem for the software vendor.

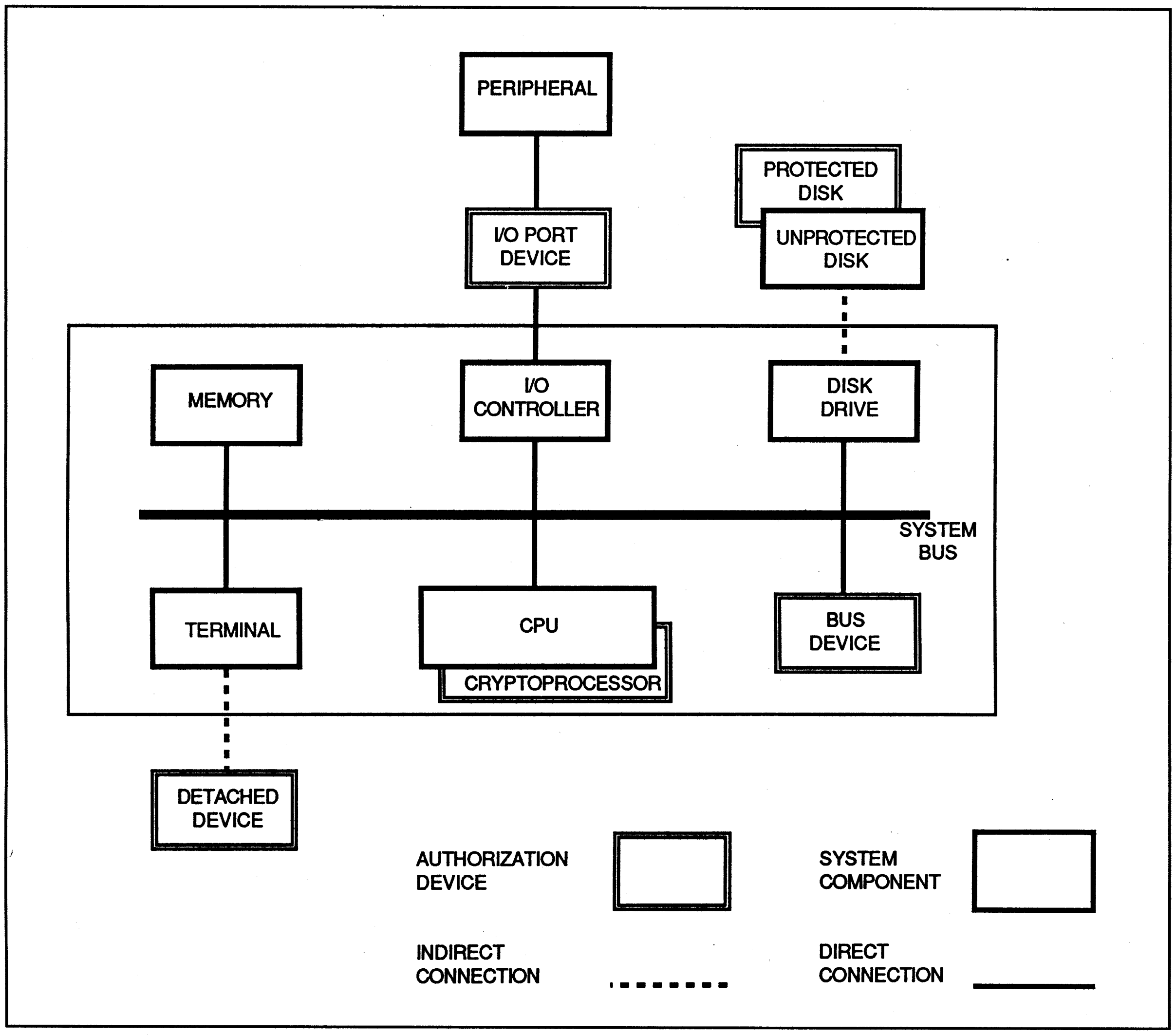

Figure 2. Authorization device attachment. (Not all devices would be attached simultaneously.)

Implementation techniques. Implementation techniques are categorized according to the hardware involved in the authorization process. There are five main categories: disks, detached devices, I/O port devices, internal bus devices, and cryptoprocessors. Each device can contain a unique key for validation or decryption of the protected program, and some can produce sequences of keys or perform the decryption. Figure 2 shows how these types of devices are attached to a typical personal computer. Table 2 gives a brief comparison.

Floppy disks. Floppy disks are now the standard means of distributing microcomputer software. In copy protection schemes, it is the difficulty of copying the disk itself that is intended to deter unauthorized use. In validation schemes, the disk contains the validation key. Such systems are essentially no more difficult to use than are copy-protected programs. However, they are the easiest to defeat, usually by using a bit copier. Nevertheless, due to their ease of use, they are also the most common protection means in use today.

A recent article in *IEEE Spectrum*[7] presents details of various disk-based schemes. Typically they involve the use of unusual data formats, such as extra tracks per disk, extra sectors per track, precisely defined sector alignment, tracks that follow a spiral (rather than circular) path, extra-wide tracks, and even tracks containing weak bits that are sometimes read as ones and sometimes as zeroes. Some of these require the user's computer to have a disk drive with nonstandard capabilities, such as stepping in half-track increments or to extra sectors, and therefore have not been applicable to all systems.

Disk-based authorization systems present a problem when the program is used on a computer with a hard disk or a RAM (virtual) disk. It is unreasonable to expect the user to insert the floppy disk for each program he wants to run, even though the program itself is on the hard disk.

Copying a key from the floppy to the hard disk and obtaining authorization from the new copy may not be possible — and would certainly be insecure. And it would be unreasonable for the floppy to be made unreadable after the program is first copied to the hard disk: the floppy is necessary as a backup in case of a hard disk crash.

The four remaining implementation techniques involve hardware devices and are seldom combined with one another; however, each can be used with copy-protected disks.

Detached devices. As mentioned with validation schemes, the validation key can reside in a device not physically attached to the computer. As Figure 2 shows, the detached device connection is an indirect one, via the keyboard and screen.

Such devices typically are about the size of a pocket calculator and have a small keyboard and a one-line alphanumeric display. The validation routine in the protected program presents a query code on the screen to the user, who types the code into the device, reads the key it displays, and enters that into the computer, which continues execution if the key is correct.

The validation routine and device sometimes simultaneously execute an algorithm to generate new query/key pairs each time, so the user cannot easily anticipate what the key will be. The main problem with detached devices is that they are inconvenient to the user because they increase his workload. The user can also err when transferring the query and key codes.

I/O port devices. Key devices attached to the computer via an I/O port are identical in principle to detached devices but are relatively transparent to the user, who need never be aware when validation takes place and who cannot introduce errors because he is not acting as a communication channel.

In theory, such a device could be used to decrypt programs, but the low bandwidth of the link would make this a time-consuming operation. The drawback to these devices is that the I/O port is usually needed for a peripheral, so the device must be transparent to peripheral data transfers, as Figure 2 shows.

In multitasking systems, an intelligent peripheral attached to the port may try to use the port at the same time the validation software is accessing the peripheral — and the peripheral's data will be lost.

Both serial port devices and parallel port devices are included in this category, because they have the same advantages and disadvantages. In practice, most I/O port authorization devices now available attach to serial ports. ADAPSO's proposed Software Authorization Standard defines communications through RS-232 serial ports

Table 2.
Comparison of implementation methods.

Implementation Method	Convenience	Installation
Floppy disk	Can be totally transparent	No special operation
Detached device	Requires user actions during every program execution	No special operation
I/O port device	Operation can be transparent Requires space at back of computer May interfere with I/O device	Relatively easy—plugged into port
Bus device	Operation can be transparent	Difficult—may require opening cabinet
Cryptoprocessor	Operation totally transparent	Very difficult—cabinet must be opened and chip replaced

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase® Smarter legal research.