PREVENTING SOFTWARE PIRACY WITH CRYPTO-MICROPROCESSORS

Robert M. Reit
1601th St. N.W.
Seattle, WA 98155

Abstract

A crypto-microprocessor executes a program which is stored in cipher to prevent it from being altered, disassembled, or copied for use in unauthorized processors. Each instruction, just before it is executed, is deciphered by the crypto-microprocessor under control of one or more secret encryption keys which are different for each program. Microprocessors lacking these keys cannot execute an enciphered program or process deciphered data. Valuable proprietary programs and data files can thus be distributed in cipher and with dedicated crypto-microprocessors for use by numerous and anonymous people, without risk of piracy or unauthorized alteration of programs.

The Piracy Problem

Protecting computer software from piracy is becoming crucial for sustained growth in the microcomputer industry. Declining hardware costs coupled with increasing software costs have made piracy increasingly attractive, which has made development of large, comprehensive software systems increasingly risky. Developers of. particular software find it difficult to protect their software investment, while manufacturers of microcomputers needing software often find it difficult to justify the risks in developing their own.

Conventional methods of software protection such as copyright and licensing, while useful for mainframe systems, are often ineffective for small computers, especially when the cost of defending a proprietary software system exceeds its value. Publishing of software as if it were a copyrighted book and pricing it low enough to make piracy financially unattractive, makes heavy investment in high-quality software uneconomically unattractive. Instead, what is needed is a protection method that is cheap, easy to implement, and which prevents piracy automatically before it happens, rather than belatedly providing remedies after the damage is done.

Fortunately, there are essential differences between a computer program and a book. The book must ultimately be revealed to the eyes of a reader, which makes it difficult to both publish and conceal it. But a program is peculiar in that it need never be revealed to the end user who owns and uses a copy of it. Only the internal circuitry of the microprocessor need access the program instructions. Consequently, the internals of a proprietary program can be sold and yet kept secret at the same time.

Crypto-Microprocessors

Programs can be protected from piracy by distributing them in cipher. An enciphered program is meaningless to anyone who tries to analyze it and is also meaningless to ordinary microprocessors. To execute enciphered programs a new type of microprocessor is needed: a 'crypto-microprocessor' (henceforth called CMP) which contains deciphering circuitry and encryption keys needed to make sense of the cipher.

An enciphered program can be executed only by a CMP that contains encryption keys that match the keys used by the software developer to encipher the program. These keys are stored by the developer into a limited number of authorized CMP's which are sold with copies of the enciphered programs. A user who has a copy of an enciphered program and a matching CMP can execute the program but cannot access the program instructions or the

Figure 1. Crypto-microprocessor chip
keys. Although anyone can make copies of an encrypted program, only CMP's with matching keys can execute it. Hence, copies of encrypted programs have no value to a pirate.

CMP's are similar to ordinary microprocessors except for additional circuitry on each chip which decrypts instructions and data. A CMP decrypts each instruction as it fetches it for execution, but these instructions are not output by the CMP chip. Hence, they cannot be copied nor disassembled for use in competing software products. Unauthorized alteration of the program is also prevented because such alteration will cause a CMP to erase its keys and thereby permanently disable itself.

Figure 1 shows a typical CMP which includes conventional registers, ALU, an aid for calculating memory addresses, a fetch-ahead instruction queue, an instruction decoder, and execution control circuitry. (The decoder is not a cryptographic unit, but is rather a conventional logic array which relates instruction op-codes to microinstructions in a control store). The instruction set of a CMP can be identical to that of an ordinary microprocessor, so that the same development system can be used for both.

CMP's should not be confused with data encryption chips such as the Intel 8254 which are used for secure data communications. Unlike encrypted messages which are disclosed by deciphering, enciphered software (cipherware) is deciphered by a CMP solely for execution within that CMP and is not disclosed to anyone. Conventional encryption chips are not suitable for cipherware because of the ease with which the owner of an encryption chip can probe the deciphered information which the chip outputs.

The cryptographic circuitry included on a CMP chip decrypts blocks of enciphered instructions as they are being read into the chip and stores plain deciphered instructions into a fetch-ahead instruction queue. Instructions and data should be deciphered differently using different keys to prevent a program from reading and deciphering itself as data. Different keys should be used for different programs so that disclosure of the key for one program will not endanger other programs. Selective activation of keys can be done by technicians using passwords without revealing either the keys or the deciphered instructions. In multi-level security systems, each level should use different keys to enforce separation of levels.

Cipherware Circuit Board

Dedicated CMP's can be packaged on the same circuit board (see Figure 2) with read-only memory or magnetic bubble memory in which an encrypted program is stored. A dedicated CMP, which executes a cipherware program stored on the same circuit board, need not resemble an ordinary microprocessor, but rather can consist in the same computer with other microprocessors which execute plain unencrypted programs. The CMP/cipherware package can provide these other microprocessors a self-contained proprietary system such as a data base management system.

An encrypted program can also be distributed on disc or tape along with a module containing a CMP which has a matching encryption key. The program can then be loaded into main memory (still in cipher) for execution in the CMP. Proprietary data can also be securely distributed in cipher on an optically-readable videodisc or magnetic disc along with the program that processes it. The program for processing this ciphered data must itself be in cipher to prevent a pirate from patching the program to decipher and output all or the data on the disc.

Encryption Methods

There are seven requirements which an encryption method should satisfy to be suitable for a CMP. It should be sufficiently fast so that execution speed is comparable to that of ordinary microprocessors. Since a program generally contains loops and jumps, the encryption method should work in a random-access sequence. The encryption method should also be impractical to break. A simple method might deter as amateur, but a determined pirate could be expected to spend many hours trying to break the cipher.

Pirates will study the external operation of an executing as to what it contains internally. Reconstructing the program from these clues should be made just as costly and time-consuming for them as coding a similar program independently. Pirates will also try to trick the processor into disclosing its instructions by altering bits in the encrypted program. This can be done by providing one or more self-disabling op-codes in the instruction set which erase the keys if they are executed. A pirate attempting to patch an encrypted program...
will create random-bit garbage in the instruction queue. Some of this garbage will include a self-destroying op-code which will permanently disable the CPU.

Simple substitution and transposition

There are several encryption methods which are not suitable for CPU's because they can be easily broken. Modifying the instruction decoder, for example, to use up-codes which are different from the published op-codes, is easy to break and does not protect address and data bytes. Similarly, crossing traces to scramble the bits in each byte is a transposition method which is easily broken.

Block ciphers

Very high security can be achieved by employing a program in blocks of 8 bytes using the IBM-developed Data Encryption Standard (DES). Since DES is practically unbreakable, using DES to encrypt programs has the strong advantage of establishing credibility. Unfortunately, microcode implementations of DES are much too slow for deciphering block ciphers even when timing matched for execution. Random logic implementations of DES using 16 cycles to decipher each 8 bytes of program are also too slow if CPU performance is to approximate that of ordinary microprocessors. A block cipher which decrypts 8 bytes in 8 cycles is shown in Figure 3. This design is similar to DES and uses many of the features of DES. The boxes labelled F are described in detail in the Federal standard. Each half-block 6 bytes is exclusive ORed to a complicated function of the other half-block. This is done alternately right to left, then left to right for 8 iterations. Each iteration takes one clock cycle and is controlled by one of the two keys. To change from deciphering to encrypting the two keys are interchanged. By overlapping bus addresses, fetch, and deciphering cycles, the delay introduced by this 8-cycle deciphering process can be minimized.

At least one of the iterations should be controlled by the address of the block being deciphered, so that each block is deciphered differently, depending on where in memory it is stored. This is necessary to prevent a pirate from rearranging the blocks to execute valid blocks in an unauthorized sequence or to decipher unauthorized data in lieu of authorized data.

If block encryption is used for data as well as program instructions, cache buffering of deciphered blocks is still done on a repeated deciphering of the same block. Once a block of data has been deciphered and stored into a cache buffer on the microprocessor chip, further reference to that block can be done directly to the cache buffer instead of deciphering the block again for each byte. Cache buffering of data can be avoided by using a polyalphabetic substitution cipher.

Polyalphabetic ciphers

In polyalphabetic stream ciphers such as the Vigenere, each bit of a message stream is exclusive-ORed to a bit in a long stream of quasi-random key bits. Such a system is not suitable for CPU's because a program must be able to process data in a random-access sequence. However, a similar cipher can be produced for any address sequence by exclusive-ORing each fetched byte to a scrambled function of that byte's address. This "scrambled-address cipher" has the advantages of simplicity and high-speed, and does not require cache buffering. It is also highly secure when used with read-only memory and a good scrambling function.

This scrambling function can consist of several alternating stages of half-byte substitution (similar to the S-boxes used in DES) and columnar bit-decomposition where the location is used for only one byte value, such a cipher is effectively a random-access version of

---

Figure 3. Block cipher for program encryption
a one-time-key Vernam cipher. This method (which is used in Figure 1) for dectyping data is very fast because cycles used for scrambling addresses can overlap the external bus addressing cycles. Exclusive-ORing a scrambled address to a byte can be done as the byte is being read from the data bus. Hence no additional cycles are needed and performance is not degraded.

Uses for Crypto-Microprocessors

CMF's can protect proprietary programs and data stored in diskettes, which have high development costs, which are difficult for competitors to reverse engineer, and which do not require customizing. Such software includes microcomputer operating systems, data base management systems, advanced compilers, small business systems, interactive systems which access proprietary databases, speech recognition systems, video graphics systems, etc. In general, microprocessor software which is likely to be pirated without protection can benefit from CMF's.

One example of a new application that can benefit from software protection is an interactive encyclopedia which is distributed on videodiscs to users who have videodisc players. A Phillips disc can store a billion bytes which is more than enough to store 4 sets of the Encyclopedia Britannica. A billion bytes of software obviously requires a very heavy investment, especially if it includes a substantial percentage of program instructions along with the data. Companies that currently make databases available through time-sharing networks, may want to distribute their data products on videodiscs for restricted access in microcomputers. CMF's can be used to televise that such proprietary data is accessible only for piecemeal inquiries and not for indiscriminate duplication.

CMF's can also be used to prevent alienation of sensitive programs by maintenance personnel. Programs used in automated toller machines (ATM's), security kernels, secure operating systems, etc., are too secure if anyone who has legitimate access to the wiring connected to a processor can alter the programs or the hardware with unauthorized patches and firmware trapsources. Such alteration can be prevented by ciphersware executed in a CMF.

Conclusions

The current debate over whether microprocessor software should be protected by copyright, trade-secret law, or non-disclosure contracts may eventually become moot. Anyone who wants to protect a new program or proprietary data from anonymous users may choose to encrypt it, thus gaining protection which is safe and effective unless which prevents piracy automatically before it can occur. CMF's, by preventing piracy, may encourage accelerated investment in microcomputer software. It is simply good business to minimize risk from piracy so that software can be priced for an attractive return on investment, and so that sales volume is not reduced by unfair competition.

References


3. Fiestel, Horst, W. A. Notz, and J. L. Smith, "Some Cryptographic Techniques for Machine-to-


