A TWO LEVEL DISK PROTECTION SYSTEM

by

Frank Sindelar and Lance J. Hoffman

Memorandum No. ERL-M452

24 May 1974

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720
A TWO LEVEL DISK PROTECTION SYSTEM

Frank Sindelar and Lance J. Hoffman

Computer Science Division
Department of Electrical Engineering and Computer Sciences
and the Electronics Research Laboratory
University of California, Berkeley, California 94720

Abstract

This paper introduces an inexpensive hardware method for protecting disk storage. A password scheme combined with data encryption at the disk controller provides two types of protection at a very modest cost in both hardware and CPU overhead.

Research Sponsored by National Science Foundation Grant GJ-36475.
Introduction

With the advent of inexpensive bulk storage devices, the collection and storage of sensitive information promises to increase. Protection of this information has usually not been implemented due to worries about the cost of the security measures necessary. Most security efforts have been implemented in software; in many cases this has increased overhead to such an extent that the cost-effectiveness of the scheme could be questioned.5,6,7

This paper describes an inexpensive hardware method which enhances the security of information on disk storage. One type of protection is a password scheme to insure the privacy of a file and the second method is the enciphering of the data on the disk. The enciphering hardware is located within the disk controller itself.

The disk drive used is a Shugart Associates SA901 Disk Drive. The host microcomputing system is a microcomputer similar to the Intel MCS-8 System but with some modifications. The main modification is the Direct Memory Access Channel added for I/O devices. This does not tie up the CPU for data transfers.

The cost of the encryption hardware is less than 5% of the total hardware cost of the controller electronics. With two methods of protection, passwords and encryption, it is felt that the protection afforded by these schemes is well worth the cost involved.

After designing the controller and password schemes, it was found that additional hardware had to be added into the microcomputer itself to insure the integrity of these protection schemes. Specifically, the microcomputer had no memory protection at all; therefore, a very simple
memory protection hardware circuit was incorporated to protect the hardware password algorithm.

Although we here discuss one direct implementation of this concept, the method may be expanded onto other computing systems and other disk drives.

The system described is currently a paper design only. The results obtained this far are based on analysis only. Work is now in progress to develop a hardware implementation of this system.

Host Microcomputer System and Disk Controller

The host microcomputer that communicates with the disk controller is an Intel MCS-8-type system. Figure 1 shows the functional interconnections between the CPU, main memory, and peripheral devices. The DMA channel was incorporated into the system design to allow for various peripheral devices to be utilized without degradation of the system speed. Figure 2 is a functional diagram of the microcomputer. Once the CPU initiates a data transfer to the disk, the disk controller asynchronously takes care of the entire data transfer of 128 words (each 8 bits). The controller then signals the CPU that the transfer has been done and whether or not it was a successful transfer. If an error is found during the transfer, the controller automatically does a re-read to attempt to recover from the error. After eight retries, if the error persists, the controller interrupts the CPU with an error indication.
Central Processing Unit

At the center of the microcomputer system is the Intel 8008, an 8-bit parallel CPU. The CPU uses a multiplexed 8-bit paralleled bus for all instruction and data transfers to and from the rest of the system. Figure 3 shows the CPU timing. A typical instruction cycle includes sending out the low order bits of the program counter during T1 followed by the high order bits during T2 over the bus lines. At T3, the instruction is then brought over the bus and execution is begun. T4 and T5 are reserved for instruction execution.

The instruction set for the Intel 8008 CPU consists of instructions that are from 1 to 3 memory cycles in length. An input-output instruction, denoted by the high order 2 bits (D6 and D7) of the high order address being off, is composed of 2 memory cycles. A memory cycle 1 is characterized by the next instruction for execution being brought into the CPU during T3. A memory cycle 2 follows this with the actual I/O instruction being sent out to the device controllers on the data base. Figure 4 shows the I/O Instruction timing. For additional information about the Intel 8008 CPU, see reference 2.

Memory

The microcomputer system's main memory, a maximum of $2^{14}$ words, is divided into pages of 256 words $\times$ 8 bits. This is a result of the CPU's addressing scheme utilizing 8 bits of low order address followed by 6 bits of high order address. The high order 6 bits specify the page number and the low order 8 bits designate the word number within the page. There are two types of main memory incorporated in the system; the first
type is the electrically alterable ROM (or PROM), which contains the system
monitor and the password software. The second type of main memory is
the RAM to be used for general user program storage and execution. For
additional details of the microcomputer system architecture, the reader
is directed to reference 1.

Peripherals

Although this microcomputer allows for several peripherals (e.g.
disk, cassette, and CRT), as shown in Figure 1, this paper is concerned
solely with the disk controller and disk protection. The disk used was
the Shugart Associates SA901 Disk Drive. This is an inexpensive flexible
disk with a moderately fast transfer rate and excellent reliability. The
disk is divided into 77 concentric tracks. Each track is divided into
32 sectors of 128 words per sector. This allows for two sectors to be
read into memory to fill one page of RAM. The disk controller is not
commercially available; it has been designed and built by the principle
author. See reference 3 for complete details about the disk drive.

As mentioned earlier, the controller asynchronously transfers blocks
of data of 128 words (one half of a RAM page) to the disk. So along with
the file's track number (6 bits) and sector number (5 bits), the page of
RAM (6 bits) must be specified with a flag (1 bit) of whether the first
128 words of the page or the last 128 words of the page is to be trans-
ferred. The last bit remaining in this word then specifies whether a
read or write is to be done. Therefore, three words are needed for any
data transfer while the use of encryption necessitates sending the con-
troller a fourth word, the encryption key. Figure 5 shows the input
circuitry necessary to receive the encryption key and other information required to execute a data transfer to or from the disk.

**Software Password Validation and I/O Algorithm**

The first method of disk protection implemented was the password validation and I/O algorithm. This attempts to protect files by restricting access to only those users that know the password. The password is simply a code supplied by the user along with the name of the file he or she wishes to access. The password algorithm then takes this password and attempts to match it with the appropriate password retrieved from the file directory on disc. The password is allowed to be set only by the owner (originator) of the file. If there is a match of passwords, the I/O instruction is issued. If a mismatch occurs, then control is given back to the monitor with the information that an illegal access was attempted.

The file name and associated password are stored in the PROM which cannot be altered (written into) dynamically. These memory chips can be electrically altered but must be taken out of the microcomputer and placed into a special board that allows selective writing into the array. Since the MCS-8 is a general-purpose, inexpensive microcomputer, no method of memory protection is provided by the manufacturer. Thus, although the password algorithm cannot be destroyed, it would easily be bypassed by a clever user. Hence, an additional piece of hardware was incorporated into the microcomputer to insure the integrity of the password algorithm.
Hardware to Insure Proper Use of Password Algorithm

The controller recognizes an I/O instruction as valid only when the I/O command originated from the PROM containing the password algorithm. This rules out any I/O commands issued directly by the user (located in RAM) and bypassing the password scheme. This validation is done in hardware by monitoring the memory bus during time T2 of memory cycle 1 (see Figure 4). If the instruction being fetched is from the PROM containing the password algorithm, the data input to the Secure Flip-Flop (see Figure 6) is enabled. At time T2, the Secure Flip-Flop is clocked and the signal SECUR goes high enabling the controller hardware to accept the next command as a valid I/O directive that has passed the first security test, the password.

As mentioned earlier, a clever and malicious user could circumvent the password algorithm by setting up the same registers that the password algorithm uses with the file address he wants access to, and since there is no memory protection hardware restricting access to the PROM, he then could jump to the address in the PROM that gives the validated I/O command to the disk controller. The controller then would verify that the I/O did in fact originate from the password algorithm in the monitor's page, and would therefore honor the I/O command as valid. This is unacceptable; therefore, additional hardware was incorporated into the microcomputer to restrict entry into the password algorithm to only the beginning location of the algorithm. This was implemented in the following way.

A jump instruction is composed of 3 memory cycles. During memory cycle 1 the instruction is fetched from the RAM or PROM. At memory cycle 2, the low order 8 bits of the address (word number to be jumped to)
are brought in during time T3. During time T3 of memory cycle 3, the high order 6 bits of the address (page number to be jumped to) are brought in over the data bus (see Figure 7). Any jump instruction executed by the CPU activates a pair of comparators that allows execution to continue only if the address is less than or equal to the beginning word of the password algorithm OR greater than the last word of the algorithm. This allows jumps to other positions within the monitor page that do not affect the password integrity. Additional hardware was used to allow for any jumps to be executed if they originated from within the password algorithm itself.

The hardware just explained is really just an implementation of a gatekeeper. Figure 8 shows several legal and illegal jumps to the password algorithm and Figure 6 shows the hardware realization of this feature. So with these modifications added, the integrity of the password algorithm is insured.

**Encryption Hardware**

The second method of protection for the disk storage data is the encryption of the data on the disk. The Shugart Associates SA901 Disk Drive is a small unit that uses a Diskette as the recording medium. The Diskette is a flexible disk about the size of a 45 rpm record. Its small size makes it susceptible to anyone removing it inconspicuously from the drive and taking it elsewhere. Since there is no bulky case to enclose the Diskette, a user could easily place it between the pages of a book and leave unnoticed. This fact has prompted a second type of protection to be implemented on the controller, encryption. If a
user does in fact attempt to read or modify a file on the disk by taking it to another drive which does not contain the password hardware, the files on the entire disk, including the file directory and the passwords, may have been garbled by a privacy transformation performed by the original controller. If a file or an entire disk is meant to be readable by other drives an encryption key of all zeros is loaded; this key causes no encryption of the data being stored. This feature makes the controller and disk drive unit compatible with other similar drives.

The read and write circuitry of the controller has been implemented in such a way that encryption can be done without excessive hardware modifications. Figures 9 and 10 show the controller read and write circuitry, respectively.

The data stream received from the disk drive is bit-serial. The data is stored between clock pulses on the disk. A start read pulse enables the clock pulse to trigger a one shot multi-vibrator, which opens a 3 microsecond window for the data pulse. If the data pulse is present during this interval, it is latched in the data Flip-Flop. Now the encryption key is XOR'ed with the data and shifted into the serial-to-parallel register. Decryption is done on a bit by bit basis so minimal hardware and timing is required.

The write circuitry works in a similar manner by XOR'ing the clear text data with an encryption key as it is sent from the parallel-to-serial register to the write data one shot multi-vibrator.

The encryption key generation hardware involves ten D-type Flip-Flops connected as a shift register (see Figure 11). The first input is XOR'ed with various outputs of the string to modify the first Flip-Flop as the
shift register is clocked. This feedback arrangement can be set by the security officer to any configuration desired, thus changing the algorithm by which the key is transformed. The initial key is preset in the registers from the input latches loaded by the password algorithm. The output of the key generator is taken from the low end of the register and is fed to the XOR gates in Figures 9 and 10.

This method of key generation is not a very sophisticated linear transformation. A string of 1023 non-repeating bits is generated by the shift register; this is sufficient for the sensitivity of the data at this particular facility. If it is known that a linear shift register of length \( N \) is utilized as the random key, then a subverter need only obtain \( 2N \) bits of the key to determine for feedback arrangement.\(^{12}\)

Thus, the method described here introduces a work factor that would discourage many would-be subverters, but not all; for other applications or facilities a longer shift register or better, a non-linear transformation such as described by Feistel\(^{11}\) could easily be implemented.

**Conclusions**

The two types of protection of the disk files that are provided by this system -- passwords and encryption -- afford the user more security than standard in microcomputers at a relatively low cost. A malicious user has a 1 in \( 2^8 \) chance of foiling the password protection by guessing it. However, even if unauthorized access to the file is gained, the encryption of the data makes it meaningless until decrypted, and decrypting the data is nontrivial. A proper selection of feedback configurations can make this method of key generation a non-repeating key for 128 words.
of 8 bits.\textsuperscript{4} So a considerable work factor is involved to regain clear
text from the disk.

The cost of additional hardware involved to implement the encryption
was approximately $10.00, under 5\% of the cost of the disk controller
electronics. The CPU overhead involved to implement the password and
encryption schemes is less than 0.47\%. This figure has been derived
from analysis only since the implementation is not completed at this
writing. The 0.47\% CPU overhead figure was derived by noting that the
password and encryption key must be added to the file name when executing
an I/O instruction. This adds 50\% more instructions for the file design-
ation and 33\% more instructions for the transfer of the data to the
controller for an I/O operation. This includes the track address, the
sector number, the memory address, and the encryption key. The real
savings is gained in the asynchronous data transfer; since these four
instructions are needed for initialization of the I/O command, this time
is then spread over the 128 words that are being transferred. The
encryption of the data is of no cost (in time) since no interruption of
the data flow is needed. This is summarized in Table 1.

The results obtained here support the idea that data
encryption at the device controller level is feasible. Although the
particular design described here is for a specific microcomputer and
disk drive, the results are expected to be similar for other designs and
other systems.
References


<table>
<thead>
<tr>
<th>WITHOUT PROTECTION</th>
<th>WITH PROTECTION</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User Must Supply</strong></td>
<td><strong>Name of File</strong></td>
</tr>
<tr>
<td>Name of File</td>
<td><strong>File Password</strong></td>
</tr>
<tr>
<td><strong>File Handler Must</strong></td>
<td><strong>Search for Space</strong></td>
</tr>
<tr>
<td>Search for Space</td>
<td>AND <strong>Match Password</strong></td>
</tr>
<tr>
<td>OR</td>
<td>OR <strong>Set Password</strong></td>
</tr>
<tr>
<td>Assign Space</td>
<td><strong>Assign Space</strong></td>
</tr>
<tr>
<td><strong>Disk Controller Needs</strong></td>
<td><strong>Memory Address</strong></td>
</tr>
<tr>
<td>Memory Address</td>
<td><strong>Track Address</strong></td>
</tr>
<tr>
<td>Track Address</td>
<td><strong>Sector Number</strong></td>
</tr>
<tr>
<td>Sector Number</td>
<td><strong>Encryption Key</strong></td>
</tr>
<tr>
<td><strong>Memory Usage</strong></td>
<td><strong>File Name</strong></td>
</tr>
<tr>
<td>File Name</td>
<td><strong>Password</strong></td>
</tr>
<tr>
<td></td>
<td>Password Algorithm</td>
</tr>
</tbody>
</table>
Figure 1. Peripheral layout

- Teletype
- CRT keyboard
- Microcomputer system
- DMA
- Main memory
- Disk controller
- Floppy disk
- Cassette tape
- Disk Floppy controller
- Tape system memory
Figure 2. Microcomputer functional layout

- CPU 8008
- External input multiplexer
- Interrupt control
- PROM
- Low address latch
- DMA data
- DMA low address
- DMA high address
- RAM
- External data
- Interrupts
- Disk
- CRT
Figure 3. CPU Timing

- Execution
- Instruction or data in
- High order address out
- Low order address out
- T1
- T2
- T3
- T4
- T5

Symbols:
- $\phi_1$
- $\phi_2$
- Sync
Figure 4. I/O instruction timing

<table>
<thead>
<tr>
<th>Memory cycle 1</th>
<th>Memory cycle 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_1$</td>
<td>$T_1$</td>
</tr>
<tr>
<td>$T_2$</td>
<td>$T_2$</td>
</tr>
<tr>
<td>$T_3$</td>
<td>$T_3$</td>
</tr>
<tr>
<td>$T_4$</td>
<td>$T_4$</td>
</tr>
<tr>
<td>$T_5$</td>
<td>$T_5$</td>
</tr>
</tbody>
</table>

Lo order address (word #)  Hi order address (page #)  Instruction in  Decode and execute  Lo order address (word #)  Hi order address (page #)  Instruction out to device controllers  Not used for output instruction

- Fetching instruction from memory
- Issuing I/O instruction
Figure 5. Input latches for data

- Track number
- Sector number
- Memory page number
- Read/write bit
- Even/Odd bit
- Encryption seed number
Figure 6. I/O validation and gatekeeper circuitry
Figure 7. Jump instruction timing

<table>
<thead>
<tr>
<th>Memory cycle 1</th>
<th>Memory cycle 2</th>
<th>Memory cycle 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>( T_1 )</td>
<td>( T_1 )</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( T_2 )</td>
<td>( T_2 )</td>
</tr>
<tr>
<td>( T_3 )</td>
<td>( T_3 )</td>
<td>( T_3 )</td>
</tr>
<tr>
<td>( T_4 )</td>
<td>( T_4 )</td>
<td>( T_4 )</td>
</tr>
<tr>
<td>( T_5 )</td>
<td>( T_5 )</td>
<td>( T_5 )</td>
</tr>
<tr>
<td>Word# out</td>
<td>Word# out</td>
<td>Word# out</td>
</tr>
<tr>
<td>Page# out</td>
<td>Page# out</td>
<td>Page# out</td>
</tr>
<tr>
<td>(jump)</td>
<td>(word)</td>
<td>(page)</td>
</tr>
</tbody>
</table>

Word\# Page\# Inst. | Word\# Page\# Data | Word\# Page\# Data
Figure 8. Legal and illegal jumps

Controller (Disk)

PROM
(Password algorithm and monitor)

RAM
(User's program)

Start
Jmp to 76
...

Jmp to 100
...

I/O to disk
...

Jmp to 10
Stop

Start
Jmp to 40
...

Jmp to 100
...

I/O to disk
...

Stop

Legal

Illegal (blocked)

Legal

Illegal (blocked)

Illegal (blocked)

Legal

Legal
Figure 9. Read circuitry

Clock

Data

Encryption key

Start read

Done read

one shot 3\mu sec

Serial-to-parallel register

To DMA
Figure 10. Write circuitry

From DMA

Encryption key

Parallel to serial register

Write data one shot

Data and clock to disk drive

200 ns

Clock
Figure II. Key generation

Clock 1 2 \ldots 8

Encryption key