System and software security via authentication handshake in EPROM

by DAVID HOFF, ALAN FOLMSBEE, and LAWRENCE LETHAM

INTEL Corporation
Folsom, California

ABSTRACT

A 128-Kbit EPROM is described that can protect systems and software from unauthorized use. The KEPROM™ keyed-access EPROM will not allow read access to its array until an authentication handshake has been completed. The handshake uses two EPROMs and involves encryption of pseudorandom numbers using a secret 64-bit key stored in each part. Security is provided by requiring the two EPROMs to contain the same 64-bit key before either will grant read access. Because the secret key is known only to the system manufacturer, the use of unauthorized software with authorized systems, and vice versa, is prevented.

KEPROM™ is a trademark of INTEL Corporation.
The cost of software piracy is growing at an enormous rate. Disk-based software is routinely copied without a thought given to legality. Computer clubs often purchase one copy of a particular software package and distribute unlicensed duplicates to members. These and many other forms of prohibited software distribution account for the growing dilemma over piracy now faced by the software industry.\(^1\) With the ever-growing proliferation of computer equipment, software protection is needed to stop the spread of piracy.

Attempts to provide copy protection of disk-based software have proved feeble. For every security mechanism developed, means to break it follow shortly thereafter. Programs such as LOCKSMITH, from Omega Software Products, Inc., have foiled copy-proof disks. Conservative estimates of sales losses due to software theft can easily run as high as 30% of legitimate sales.\(^1\) This major loss to a billion-dollar-a-year industry can no longer be tolerated. Effective, inexpensive solutions are needed for this problem; otherwise, much of the incentive to produce a large selection of high-quality software will be gone, and the computer industry as a whole will suffer.\(^1\)

A new IC nonvolatile memory device has been developed to provide software and system security. The KEPROM keyed-access EPROM erects a barrier between software and pirate, protecting systems and software alike from unauthorized use. It is completely pin-compatible with the standard 128-Kbit EPROM and contains 128 Kbits of EPROM memory, a secret 64-bit key, a pseudorandom number generator (PNG), and encryption circuits. When the EPROM is first powered up, its array is locked and cannot be read. In order to grant read access, an authentication handshake must be performed between two EPROMs, each containing the same 64-bit key. If the two keys do not match, the handshake will fail and neither chip will grant read access. Only if both chips contain the same secret key will read access be granted. The correct key is known exclusively by the system manufacturer and his designated parties. Security is maintained by requiring both authorized chips to be present in the system before either will unlock.

Figure 1 shows a system using two EPROMs. The system EPROM, known as the recipient, contains vital operating system subroutines, without which the computer will not function. This EPROM is attached permanently to prevent attempted substitution by pirates. The chip may be epoxied to the circuit board or a metal shield may be riveted over it. Another technique is to place a seal on the system EPROM that must be broken to remove the chip. The customer is instructed that breaking the seal results in a voided warranty. This method of security is inexpensive, and allows for easy upgrading by the dealer. Few customers are willing to subject a system to the physical damage that results from removing a riveted or epoxied chip, nor are they willing to void a warranty by breaking a seal. With the system EPROM permanently attached, only authorized EPROMs containing the correct 64-bit key can be run on the system. This allows the manufacturer to protect the system and the software from prohibited use.

The removable cartridge contains one or more authorized software packages stored in its EPROM. The originator EPROM contains any one of the many popular software packages available today, including word processing, accounts payable and receivable, or spread-sheet. To run a particular package the user need only plug the correct cartridge into the system. The cartridge software calls certain vital subroutines in the system EPROM, requiring the application to run only on systems containing the EPROM with the correct key.

When power is first applied, both EPROMs are locked; their arrays cannot be read, except for locations 0 through 01FF and 3FF0 through 3FFF hexadecimal. These 528 locations are provided for bootstrap code and can always be read. The microprocessor begins execution out of the system EPROM bootstrap code area. No program code is executed out of the cartridge until it has been shown to be authorized to run on the system, via the handshake. The handshake is two-way so that both sides are authenticated to each other. It begins with the originator EPROM, located in the cartridge, generating a 32-bit pseudorandom number. This number is transferred serially by the microprocessor to the recipient.
Eight dummy read cycles are then performed on the recipient to load the secret key from the array into the encryption circuits. Each chip scrambles the number using its own secret key. The encryption algorithm is a complex, nonlinear substitution-permutation algorithm modeled after the Data Encryption Standard (DES). After the encryption is complete, the 32-bit result from the recipient is transferred back to the originator, where it is internally compared against the originator’s own encryption result. If the two EPROMs contain the same secret key, then the encrypted outputs will match and the originator will grant read access to its entire array. At this point, the system has been authenticated to the cartridge; it is authorized to run the software.

To provide protection against running unauthorized software on the system, a second half of the handshake is performed to authenticate the cartridge. The two EPROMs reverse roles, with the system EPROM now generating the pseudorandom number. Upon successful completion of the second half, the cartridge has demonstrated that it is authorized to run on the system. The system EPROM will grant read access and execution of the software package may proceed at full speed. The total time for the two-way handshake ranges from 100 milliseconds to 10 seconds, depending on the setting of a programmable delay counter. The architecture of the EPROM provides a high level of security at no penalty in system performance. Once unlocked, the KEPROM device operates as a standard EPROM.

The two-way handshake is a powerful feature of the KEPROM device in that it protects both the software and the system from unauthorized use. The random-number encryption scheme allows two initially unfriendly devices to authenticate each other as containing the same secret key, without either side revealing what the key is. If the key were exchanged, a pirate could decode it after one unsuccessful handshake—and security could be violated. The PNG makes every handshake different, which prevents memorization of the correct sequence for authorization. Each device can authenticate the other while maintaining its own security. This makes the EPROM useful in numerous friend–foe applications, including those not directly related to the security of software and systems.

In the example shown in Figure 1, the system EPROM is able to authenticate any authorized software cartridge EPROM containing the correct 64-bit key. There are many applications in which it is desirable to have the system EPROM authenticate EPROMs containing different keys. Versatile personal computers must be able to run numerous software packages from many different distributors, all of whom want to keep their software secure from theft. Sharing one key among all software vendors would significantly degrade security for each program. To allow flexibility when securing several software applications using different keys, the key manager mode has been added. When the EPROM is programmed as a key manager half of its array becomes storage for 1024 keys. The other half is used to store data and programs. Once locked as a key manager, the keys can never be read at the outputs again. The key manager is typically used in conjunction with other nonkey managers. The one key manager is able to authenticate up to 1024 different EPROMs, each with its own key.

In the example, the system EPROM could be a key manager able to authenticate different EPROMs. Each authorized application would have its own key, allowing it to authenticate to the system key manager. Different software vendors would be allocated private sets of keys, which would protect their security. Thus, the system manufacturer also has the ability to control the software run on his system. The manufacturer could even have several versions of key manager EPROMs for the same system hardware. The level of sophistication for a given model could be based on what keys were programmed into its key manager. A complete system EPROM would be programmed with the full set of keys; the key manager in the economy system would have fewer. A low-cost system designed to run only word processing and spread sheet software would have its key manager programmed with keys for those specific packages only. Therefore, a pirate or an unscrupulous customer would be unable to remove the key manager, thus preventing the use of pirated software on the system.

When a customer wants to upgrade his system legally, the system board is traded in for one with a key manager containing more keys. If a seal is placed on the key manager, the dealer simply replaces the key manager alone. A new seal is applied to ensure that no tampering occurs with the new key manager.

Another benefit of the key manager mode is that several different software cartridges can be authenticated to the system simultaneously. Each cartridge can contain valuable programs or data. The system shown in Figure 2 has the key manager authenticated to four cartridge EPROMs. The handshake is performed four times; once for each cartridge.

To ensure that the correct key is loaded out of the key manager array during the handshake, a special register has been included on the EPROM. The nonkey manager has this register programmed with the address of its key in the key manager.
manager array. The key address is read by the microprocessor and is used to load the correct key from the key manager for encryption. If the wrong key is loaded or the EPROM is not authorized to be used on the system, then the keys will be different, resulting in a failed handshake.

Computer users who are involved in software pirating have varied levels of sophistication. The casual copier duplicates a disk containing a popular word processor or video game. Mostly, these copiers have minimal technical skill and are able to copy software simply because it is easy to do so. Many business owners and hobbyists fall into this category. Uncopyable disks are easily duplicated by these casual copiers once the special trick is learned.1

The EPROM will severely inhibit the casual copier who attempts to use an EPROM programmer to copy the KEPROM memory. This attempt will not work because the EPROM is locked and its contents are unreadable. Any other such attempts will likewise fail. Even if the EPROM software is copied from an authenticated system by monitoring the busses, it cannot be run on another authenticated system. If it is stored on disk, the disk cannot perform the handshake needed to unlock the system EPROM. Storing the code in another KEPROM memory is likewise futile without the secret key. The casual copier soon will give up and purchase a legitimate copy of the software.

The next level of software pirate may be termed the technician copier. These pirates have a basic understanding of electronics and computer architecture. They also have power supplies, development systems, and test equipment to help them. Most software pirates fall into one of these two categories. The technician copier can put the cartridge in an authorized system and allow the handshake to occur, unlocking both parts. He then monitors the address and data busses and, in time, discovers all of the data stored in the EPROMs. But he still does not know the secret key and—like the casual copier—cannot use the software on an authorized system. Without the correct key, his bogus cartridge will fail the handshake and the system will stop execution. The technician copier is thereby prevented from peddling his software to authorized system users.

The most sophisticated copiers have access to IC reverse-engineering lab equipment. In the lab, the EPROM can be dissected in an effort to determine the secret key. The success of this attack is highly unlikely, because the key is stored only as a charge in EPROM cells. It cannot be seen by examining the die. Even if the attack is successful, the cost would run into the millions for lab equipment alone. This is far beyond what most copiers can afford and it prevents any effective attack on the EPROM. In addition, any organization large enough to effectively reverse engineer the EPROM also can be sued for copyright infringement. The EPROM is thus able to protect valuable software from pirates. Above all, the EPROM removes the economic benefit of theft that has allowed software piracy to become so widespread.

The encryption and PNG circuits critically affect EPROM's security. The pirate foiled in his attempt to break into the EPROM will most certainly attack these two important circuits. The PNG must generate truly random-looking numbers. If the output were to repeat too frequently or degenerate into a sequence, the EPROM could fall prey to cryptographic attack. The encryption algorithm also is subject to attack by pirates, who can purchase an EPROM with the desired software and feed it chosen random numbers. The encrypted outputs could be analyzed to determine the key. This form of attack is known as a chosen-text attack and is the most difficult to withstand.2 Such considerations were taken into account in the design of these two vital circuits.

When considering the analysis of a sequence of numbers to determine their degree of randomness, one is stuck by the question of how one can determine whether a sequence is random at all. Any sequence can be random; a sequence of zeroes has a finite probability of being output from a random-number generator. The mathematician, Donald Knuth, in his book Seminumerical Algorithms, explores this question at great length.3 Among his major points is that a sequence can be evaluated as random using properties that are present in ideal random sequences.

The following seven tests were used to evaluate 900,000 random numbers collected from ten EPROMs. First, a search was performed to determine the most common pseudorandom number generated. The number 00000000 appeared 18 times. 128 numbers appeared between 2 and 11 times, resulting in more than 99.97% of the numbers generated appearing only once. The low percentage of repeat numbers generated hinders attacks by a pirate who monitors the PNG output for multiple occurrences of the same pseudorandom number. A listing of the numbers occurring three or more times is given in Table I.

In the second test, the correlation coefficient between consecutively generated pseudorandom numbers was tested to see if the next output generated is a function of the present one. The value of 0.00238 was impressive, indicating no such connection.

Next, the correlation coefficient between the two words within each pseudorandom number was calculated to see if the components of the number are related. The value of 0.0164 indicates no relationship.

In the fourth test, the correlation coefficient of two pseudorandom number lists beginning with the same value was calculated. This experiment examined the output for sequences. The value of 0.00936 strongly indicates that a sequence is not being generated.

In the fifth experiment, the correlation coefficient between

<table>
<thead>
<tr>
<th>Number</th>
<th>Occurrence Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000</td>
<td>18</td>
</tr>
<tr>
<td>20000000</td>
<td>9</td>
</tr>
<tr>
<td>80000000</td>
<td>8</td>
</tr>
<tr>
<td>40000000</td>
<td>5</td>
</tr>
<tr>
<td>10000000</td>
<td>4</td>
</tr>
<tr>
<td>28000000</td>
<td>3</td>
</tr>
<tr>
<td>9C800000</td>
<td>3</td>
</tr>
</tbody>
</table>
lists from two different EPROMs was tested to see if the outputs were mathematically related. The result of −0.0011 indicates no such relationship.

Next, the percent of 1 bits generated was shown to be 50.15%, which is close to the ideal value of 50%.

Finally, a frequency-of-occurrence table for nibbles was developed. This is shown in Table II. The maximum disparity in counts for nibbles is only 5%, indicating an even distribution of values generated.

The results obtained in these tests were quite impressive and more than adequate for the intended operation. The count of 18 for the most common number results in an average spacing of 50,000 between repeats for that number. The "birthday problem" theory predicts about a 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples. The PNG described here has a more than 50% chance of repeating a 32-bit number after 65,546 samples.

<table>
<thead>
<tr>
<th>Nibble value</th>
<th>Occurrence count</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>324786</td>
</tr>
<tr>
<td>1</td>
<td>319417</td>
</tr>
<tr>
<td>2</td>
<td>311149 MINIMUM</td>
</tr>
<tr>
<td>3</td>
<td>322481</td>
</tr>
<tr>
<td>4</td>
<td>312293</td>
</tr>
<tr>
<td>5</td>
<td>317407</td>
</tr>
<tr>
<td>6</td>
<td>312401</td>
</tr>
<tr>
<td>7</td>
<td>326608 MAXIMUM</td>
</tr>
<tr>
<td>8</td>
<td>318696</td>
</tr>
<tr>
<td>9</td>
<td>315960</td>
</tr>
<tr>
<td>10</td>
<td>319898</td>
</tr>
<tr>
<td>11</td>
<td>318512</td>
</tr>
<tr>
<td>12</td>
<td>319930</td>
</tr>
<tr>
<td>13</td>
<td>320164</td>
</tr>
<tr>
<td>14</td>
<td>326456</td>
</tr>
<tr>
<td>15</td>
<td>325890</td>
</tr>
</tbody>
</table>

detailed explanation of the PNG's internal operation is described in Reference 4.

Copies of proprietary software alone are not functional in an authenticated system; the pirate must obtain the secret key stored in the part. He may either try to analyze the encryption algorithm or perform some form of IC reverse engineering to determine the key. The latter technique is beyond the financial means and technical abilities of the vast majority of pirates and copiers. Because the key is never revealed at the outputs, the pirate is forced to attempt analysis to determine it. The pirate can obtain a recipient EPROM containing the desired software, feed it random numbers and collect the encrypted outputs. The numbers can then be analyzed in an effort to determine the key. To ensure that this analysis will fail, extreme care was used in the design of the EPROM encryption algorithm.

Principles from the DES, along with additional security concepts, were included in the design of the algorithm. Figure 3 shows a block diagram of the EPROM encryption algorithm. The random number is loaded into the accumulator, and the key register holds the lower 32 bits of the key. The encryption of the accumulator value proceeds using the lower 32 bits of the key to control the operation. When the algorithm is complete, the upper 32 bits of the key are loaded into the key register and the process is repeated. The random number is encrypted twice using both halves of the key before being sent to the outputs.

---

![Figure 3—Encryption algorithm block diagram](image-url)
The P-box performs a nonlinear permutation of accumulator bits 29 through 32 using key register bits 24 through 26. The S-box is a substitution ROM with 6 address inputs and 4 outputs. The algorithm performed using both 32-bit halves of the key proceeds as follows:

1. Latch in the 4-bit result propagating from the accumulator and key register.
2. Shift the accumulator 4 bits to the left; disregarding bits 29 through 32.
3. Load the 4 bits from the latch into the accumulator.
4. Repeat steps 1 through 3 eight times.
5. Perform a 32-bit permutation of the accumulator. The permutation circuit (not shown) performs a wire crossing of all the bits in the accumulator. No two bits in the same nibble before permutation remain grouped together after it is complete. This foils the pirate analyzing the algorithm in terms of nibble operations.
6. Rotate the key register 7 bits to the left. The value in bit 32 is moved to bit position 7.
7. Repeat steps 1 through 6 32 times.

The substitution ROM is similar to the S-boxes in DES. The 32-bit accumulator permutation is likewise modeled after a function in the standard. Although there are similarities, the algorithm differs notably from DES for die size and security considerations. By using the current algorithm instead of a high-speed DES circuit 35,000 mm², or 70% of the total area, was saved. Additional modifications were made to ease concern about the security of DES. Thirty-two rounds, each performing a 32-bit permutation of the accumulator, are executed for each half of the key. DES has been criticized for performing only 16 encryption rounds.²

The EPROM algorithm is one-way, limiting the flexibility of the pirate in his attack. DES is a two-way algorithm, allowing both encryption and decryption of data. No parity bits are used in the key, so a true 64-bit key is available, as opposed to 56 in DES. Weaknesses have been found owing to partial linearity in the S-boxes,² the only nonlinear element in DES. The EPROM S-ROM was designed to be nonlinear. A change of a single bit in the address causes at least two output bits to change.

To heighten security, a special 6-bit permutation box was added. The 3 key bits into this P-box have a nonlinear effect on the output. By introducing the key using a nonlinear mechanism, security is enhanced. Because of this nonlinearity, the encryption is not invariant under complementation of plaintext, key, and ciphertext. The DES algorithm has been criticized for having this invariance, because it reduces the CPU time needed to do a brute-force search for the key. Using the key in a nonlinear way relieves security considerations about the DES key-scheduling algorithm.

Error propagation, the number of bits changing in the output for a single bit change in the input, is also an important property for encryption algorithms.³ A single bit change in either the random number input or the secret key should cause many bits to change in the output. The P-box and S-ROM enhance error propagation; a single input bit change usually results in about half of the bits changing at the outputs. With the enhancements present in the KEPROM encryption algorithm, many of the apprehensions about DES have been resolved. This has been done with a significant saving of die area as compared with a literal implementation of the standard. The level of security provided by the implementation is more than sufficient to thwart both the casual copier and the sophisticated pirate.

CONCLUSION

A new single-chip solution to software piracy has been described. This EPROM allows systems and software to be secured from unauthorized use, enabling the rightful owners and inventors of new technology to receive their just rewards. The KEPROM component contains 128 Kbits of EPROM memory for program storage, encryption circuits, and a pseudorandom number generator allowing it to determine if read access is to be granted. Only access by authorized systems is allowed, protecting the valuable software contained in the chip. The product is completely pin-compatible with the standard 128 Kbit EPROM and will be easily incorporated into future designs.

ACKNOWLEDGMENTS

The authors are grateful to Steve Domenik for formulating the KEPROM concepts, to design engineers Tai Nguyen and Emilio Yero, and to the layout designers, of whom the most notable are Steve Kiesler and John Zaferes. We also thank Robert Davis, Kurt Robinson, Scott Brock, Bill Spaw, Don Knowlton, Larry Palley, Ray Delissio, Dave Townsley, David Shrigley, and George Schneer for their support and work.

REFERENCES
