Data dependent keys for a selective encryption terminal

by ROBERT FLYNN
Polytechnic Institute of New York
Brooklyn, New York

and

ANTHONY S. CAMPASANO
AIL, A Division of Cutler-Hammer
Deer Park, New York

INTRODUCTION

The concept of a data dependent key for use in conjunction with a selective encryption device is presented. The use of an encryption key extracted from the data or a function of extracted data eliminates the problems associated with a statistical assault on encrypted data which was all encrypted using the same key. If this key extraction is done in hardware then the security of the key is reduced to the physical security of the device which encrypts/decrypts and not on the personnel doing the data entry. This work is premised on and builds from work done by Carson, Flury, Summers and Welsh,1,2 establishing the concept of selective encryption terminals as a feasible technology.

With the passage of the Privacy Act of 1974,3 a new area of concern has been presented to the data processing industry, i.e., the required protection of data on individuals. The recently published report on the Privacy Protection Study Commission5 goes further. “. . . the ability to search through . . . records to identify individuals with particular characteristics of interest is at once the most important gain, and the most important source of potential harm, stemming from the automation of large scale personal-data-record-keeping systems, and its importance is likely to grow in importance as ‘textual search’ techniques are refined.”6

In light of this, various methods of protecting data have been suggested. Among these is the adoption by the National Bureau of Standards of an “Encryption Algorithm for Computer Data Protection.” This algorithm is designed to encipher and decipher blocks of data consisting of sixty-four bits under control of a sixty-four bit key.4 This algorithm was implemented in an experiment performed by the METREK Division of the MITRE Corporation in 1975 and 1976. They developed the Selective Encryption Terminal which consists of a Tektronix 4023 terminal and an LSI-II microcomputer in communication with a host computer where the database was stored. For this experiment the host was an IBM 370/145 running the VM/CMS operating system.1,4

SELECTIVE ENCRYPTION

The concept of selective encryption allows the terminal operator to encrypt only those portions of an individual’s record for which protection is required or to encrypt the identification portion of a record so that the individual is not associated with the data. The latter is useful in databases where statistical studies are made with respect to certain aspects of the individual’s files. Selective encryption also allows the terminal operator to change the key so that different sections of an individual’s record can be encrypted with different keys. In the METREK experiment, a hospital database was used. Here, medical information was encrypted under one key for use by physicians while accounting information was encrypted under another key for use by the hospital’s administration.

The most important feature of the Selective Encryption Terminal is best expressed in this excerpt from the METREK report.

*The protected material remains in encrypted form throughout its residence in the host system and is decrypted only upon return to the terminal. In this manner, neither the protection (encryption/decryption) software, the protection key, nor the plaintext of the protected information is ever present in the host computer.*

The plaintext referred to is the data before encryption.

VULNERABILITY OF THE KEY

It is important to note that the security of the METREK system and any other system utilizing the NBS algorithm is entirely dependent on the security of the key or keys. Given the information that the NBS algorithm was being used, an outside intruder could get access to the protected data by one of three ways. First, the key could be obtained by trial
and error: iterating through all possible values of the key until found. Second, the encrypted text could be analyzed statistically so that any patterns in the data might reveal the key. And finally and most simply, the key could be obtained from those personnel who are exposed to the key, such as terminal operators.

What is to be considered then, are variations in the implementation of the algorithm which would combat these problems. One such variation is the concept of using data dependent keys in performing the encryption. Discovering the key by trial and error even when the data dependent keys are not used is a difficult task. With an eight byte key, each byte an ASCII character in which only seven of the eight bits vary, there are fifty-six bits which numerate all possible keys, and if it were possible to process each attempt in one microsecond, it would take on the average over one thousand years to find the key. The problem occurs if the intruder trying to break the system gets lucky or utilizes some fast and advanced multiprocessing system. With the key found, the entire database is at his disposal.

DATA DEPENDENT KEYS

Consider, rather, a data dependent key. For each record of data in the database, the key entered by the operator could be used to select certain bytes of the plaintext data which will not be protected by encryption in the record, which could then be used to derive the key for the encryption of the data to be protected. In this way, if the key used to encrypt one record was discovered, that specific key would be useless in decrypting any other records.

It is assumed, of course, that the data dependency of the key is kept as confidential as the key itself. However, if such is not the case, the data is still fairly well protected. Given that the eight bytes of the key are selected directly from the plaintext data with replacement, it would take on the average $\frac{(N^8)}{2}$ attempts to break the key where $N$ is the total number of plaintext unprotected bytes available. If $N$ were one hundred bytes and each attempt took one microsecond, the data would be protected for an average of over one hundred and fifty years. It should also be noted that the one microsecond figure is unreasonable for the technology available. The best figures to be found are those reflecting the implementation of the algorithm using $T^2L$ devices. Here the time to encrypt sixty-four bits is five microseconds.7

Protection using the above scheme could be further improved by not using the plaintext data itself as the key but rather as input to a function which creates some function of a key from the data chosen. A simple arithmetic maneuver, such as a hashing code, could increase protection manifold while not significantly affecting processing overhead. Different functions could be used for different fields and the types of functions used could be varied. Thus one function might have the property of creating widely scattered keys from varying plaintext. Such a key function would have a high velocity. Another function might be a highly noninvertible function (not one-to-one) such as adding the low order bits of the plaintext bytes.

Data dependent keys improve security against the second method mentioned, discovering the key by statistical methods. With each data record encrypted with a different key, the amount of encrypted data in each record would be too minimal to draw any conclusions about the key.

MASSIVE DATA ENTRY SYSTEMS—ENCRYPTING THE SOURCE

For a large database where data is entered by many different people but only retrieved by a few, another scheme might be in order. The premise here is that some data is available to all for statistical purposes but only a few could examine or identify a specific individual. Most massive data entry systems employ the key to disc concept. Many operators enter records at many terminals which tie into one processor storing the data on a disc. If this processor is physically local to the terminals, i.e., there is no security problem due to lines which can be tapped, then the encryption can be done at the processor and not at the terminal.

Given this configuration, the data can be encrypted at the key to disc processor using some subset of the identifying fields of the record, such as the name and/or the social security number, as the encryption key. With this key the protected records including the identifying fields can be encrypted. To retrieve the data, a special decryption terminal can be designed which, when given the identifying information used to encrypt the record, can determine the key in the same manner and retrieve the record. Here one would extract the plain text seed for various key functions. Encrypt the fields that are to be protected and then encrypt the plaintext header information (name, social security) using as a key a function of extracted subfields of the plaintext header data.

In this manner, only an individual authorized to use a decryption terminal could enter the specific name and social security number and from this extract the records on that individual. No one else could ever identify uniquely those individuals who had characteristics of interest. The data base is essentially not invertible and yet certain fields which are not encrypted could be made available to everyone for analysis.

This scheme is superior to those previously discussed in that there is no plaintext stored in the database. Furthermore, each encrypted record is encrypted with its own key, making attempts to intrude on the database via statistical methods futile. And perhaps most important, the data entry personnel do not see the encryption and may not be aware that it exists and do not have any retrieval capability at their terminal at all.

HARDWARE IMPLEMENTED KEYS

Protecting the key from the third source of disclosure, disloyal personnel, is probably the more difficult problem to solve. Personnel, such as terminal operators who are readily exposed to keys in use as well as some information concerning the implementation of the encryption are the weak-
Data Dependent Keys for a Selective Encryption Terminal

1129

est link in the security system. Depending on the terminal used, information concerning whether the key entered at the terminal is used to derive a data dependent key or applied directly as the encryption key might easily be obtainable through the keyboard. What is needed here then is a scheme by which the operator may utilize the encryption/decryption facilities of the terminal but without ever allowing disclosure of the key or the details of the implementation.

One such scheme would involve entering the key at the terminal using hardware. It is possible to fashion an intelligent terminal with inserible ROM's (Read Only Memories) similar to those used on Hewlett Packard's 9820 calculators. HP 9820 users can purchase various pre-programmed ROM's which when inserted in the calculator give the user access to a library of subroutines. A terminal with a microprocessor could be so equipped and utilize subroutines on the ROM for encrypting, decrypting, deriving keys from data and supplying the main key itself to these routines on the ROM. What is important to note about the architecture of the HP 9820 is that the information on the ROM can only be used as instructions for the processor. The code in the ROM is not available to the user as data. Data and user supplied programs can only be entered and retrieved from the calculator's RAM (Random Access Memory). It is in the RAM where the program which calls the ROM subroutines resides.

Given a terminal with the mentioned specifications, the key and the method of implementing the encryption can now be physically protected. The ROM could easily be designed so that it can be locked into the terminal for safety only to be removed by some authorized personnel. All that the operator will see is the data being processed at the terminal.

It might be useful to consider designing the terminal with two ROM ports for use when converting an entire database from encrypted under one key to encrypted under a new key. A ROM in one port would be used to re-encrypt the data.

Security, of course, would be required when the ROM program is burned in. Once the implementation of the algorithm is established, a number of ROMs could be prepared at one time leaving the key field blank to be entered when needed and at a proper time and place. There should also be some method of sealing the ROM.

Another variation of this scheme would make use of recently developed semiconductor devices, such as the Motorola "Info-guard," to perform the NBS encryption/decryption of the data. If this device were already present in the terminal and its capability available to the data processing program in the RAM, the ROM need only contain the routines for deriving and/or supplying the key to the devices.

Using a dedicated device rather than a general purpose processor to perform encryption/decryption generally implies that one could expect much faster processing time.

CONCLUSION

The Selective Encryption Terminal is by far one of the most promising developments for Privacy Protection. The data is secure from the moment it leaves the terminal. By simply supplying extra protection to prevent disclosure of the encryption key using either data dependent keys and/or a hardware implemented key, a data base is well protected from any accidental disclosure or unfriendly intrusion. The protection of data is now reduced to the physical protection of a device.

Using data dependent keys for encryption and supplying a key to the terminal via ROM instead of through the keyboard may seem awkward in terms of processing overhead and generating the ROMs themselves. However, each contributes significantly in spoiling a possible intruder's plans to a point where any attempt would be completely discouraged. These ideas, then, appear to be well worth developing. They may provide an attractive solution to the simultaneous needs of privacy and statistical reporting in government applications.

REFERENCES
