An overview of computer security

by R. C. Summers

Presented is an overview of computer security, including concepts, techniques, and measures relating to the protection of computing systems and the information they maintain against deliberate or accidental threats. Motivations for security measures are discussed. Security strategies are considered. Actions and events that threaten security are described, along with technical problems that can prevent the computer from adequately dealing with threats. Security models are surveyed. Specific technical and administrative measures for promoting security are described. Among the technical measures discussed are design of secure systems, hardware and operating systems, identification of users, encryption, and access control packages. Administrative measures include personnel, physical security of the computing system, and auditing. Also presented is the establishment of a security program. Reviewed are special problems and their solutions, including communications and networks, data base management systems, and statistical data bases. This paper is based on a paper by the author published in* The Handbook of Computers and Computing, *edited by Arthur H. Seidman and Ivan Flores, Van Nostrand Reinhold Company, Inc., New York (1984).

As organizations automate their record keeping and other operations, computer security becomes more and more vital to the functioning of the organizations. The purpose of this paper is to give those who are not working professionally in the field an appreciation for the wide range of topics within computer security.

The computer has become the main repository for most of an organization's records. Some of the records represent or are used for controlling resources such as money and inventory that can be lost to the organization through manipulation of the records. Some records are essential to the operation of an organization, some contain trade secrets, and some describe persons whose privacy must be protected. Thus one aspect of computer security is the protection of information against unauthorized modification, destruction, or disclosure.

Equally critical is the role of the computer in process control and on-line applications. Process control at a chemical plant, for example, involves the sensing of such process variables as temperature, pressure, and reaction products, followed by computation and feeding back of signals to control the process. Airline reservation systems are on-line applications that control the airline's only products—space and time. The needed data must be protected, and the computing system must be available to carry out the computations in a timely way. Thus another aspect of computing security is the maintenance of the integrity and availability of the computing system and its applications.

An additional impetus for security comes from the legal requirements of many countries and states that prescribe how personal records are to be handled. Other laws and regulations require organizations to control their assets properly. This includes assets maintained or controlled by a computing system.

The objective of computer security is to put the hardware, software, and data out of danger of loss. In this paper, the term *computer security* includes

concepts, techniques, and measures that are used to protect computing systems and the information they maintain against deliberate or accidental threats. We first consider motivations for security measures, then list possible strategies for computer security, including the controlling of access to information. Next we consider actions and events that threaten security and describe technical problems that can prevent the computer from adequately dealing with the threats.

Because a conceptual framework is essential for research in security, as well as for the intelligent application of security techniques, formal models of security have been developed. We survey some formal models and consider informal models that are implicit in many software systems and application environments.

One reason for security measures is to protect the privacy of individuals.

After developing the necessary framework, we consider two kinds of measures for promoting security: technical measures implemented within the computing system and administrative measures outside the computing system.

For measures within the system, we summarize general principles that have proved useful in designing secure systems. We describe techniques used in both hardware and operating systems to protect programs and data. We survey ways of identifying users who attempt access to a computing system. We describe encryption, a technique that can be used to guard against a wide variety of security threats. We discuss software packages that can control access to data and other resources.

Among the administrative measures we consider are those involving personnel, physical security of the computing system, and auditing and controls as they relate to computerized systems. Described next are ways by which an organization can establish a security program using the measures discussed.

Because computing systems are accessed over communication lines and are connected into networks, we review the special security problems introduced by communications and networks, and discuss the use of encryption as a security measure. Security aspects of local networks are discussed.

Data base systems have their own special requirements, which are discussed. Security features of data base management systems are introduced, and research on the security of statistical data bases is summarized.

Motivations and strategies for computer security

One reason for security measures is to protect the privacy of individuals,[1-3] so as to give them some control over information about themselves that is maintained in computerized systems. Personal information appears in the records of banks and credit institutions, doctors' offices and hospitals, taxing agencies, and in many other places. More and more, such information is being collected and kept, and wrong decisions have been made on the basis of inaccurate information, or sensitive personal data have been wrongly revealed. A number of countries now have privacy legislation. In the U.S., the Privacy Act of 1974 applies to all federal record systems, and other laws apply to specific areas of the private sector, such as credit and banking. Many states also have privacy laws. Although the various laws differ, common principles underlie them. We summarize here the principles of privacy[4] adopted by the member nations of the Organization for Economic Cooperation and Development (OECD):

- There should be limits to the collection of personal data, and the data should be obtained lawfully and fairly, with the knowledge or consent of the subject where appropriate. Here, the *subject* is the person about whom the data are being collected.
- Data should be relevant to the purposes for which they are collected, as well as accurate, complete, and up-to-date.
- The purposes for collecting the data should be specified when the data are collected and again whenever the purposes change. The new purposes must be compatible with the old ones.
- The data must not, in general, be used for other purposes.
- Data should be protected by reasonable safeguard against "loss or unauthorized access, destruction, use, modification, or disclosure...." In other words, data security should be provided.

- It should be possible to find out what personal record systems exist, their main purposes, and the names of the persons who are responsible for controlling the records.
- An individual should be able to find out whether a data bank has information about him, and, if so, to obtain access to the information. The subject should also be able to challenge the data, and, if successful, have it erased or corrected.
- A data controller should be accountable for complying with measures that implement these principles.

As another motivation for security-related controls, U.S. legislation (the Foreign Corrupt Practices Act of 1977[5]) requires all publicly held corporations to maintain internal accounting controls to ensure that transactions are executed in accordance with management's authorization, transactions are properly recorded, and access to assets is permitted only in accordance with management's authorization.

Computer-related crime as a motivation for computer security is perhaps better publicized than documented,[6] but there is reason to believe that it causes substantial losses.[7]

There is no single strategy for achieving security. One approach to security is *access control*, that is, ensuring that data are accessed by authorized persons in authorized ways only. Access control is not concerned with how the authorized person uses information legitimately obtained. Access control is concerned with access to the physical system, system software, applications, and data.

With a general strategy of access control, there is still a choice as to whether to try to prevent all unauthorized access or to allow access while at the same time detecting it and taking action against the violator. Another technique of access control is that of *information flow control*, which is the attempt to control the flow of information within the computing system and as it leaves the computing system. A longer-range goal, rarely aimed at in current systems, is that of *inference control*.[8] An inference, in this case, may be the result of combining a statistical summary with other facts one is aware of.

Another strategic approach to security is based on the concept of *integrity*. This approach, which can supplement approaches that control interaction, is designed to ensure that interactions leave data or systems in a correct or available state. *Data integrity* means ensuring that data are neither destroyed nor improperly modified. These data are correct or reasonable and accurately reflect the real objects they describe. *Application integrity* means ensuring that an application, which is typically viewed as a system, continues to operate according to its specifications and continues to be available. *System integrity* aims at the availability and correct operation of the entire computing system.

Preventing access by an unauthorized user also helps to protect the integrity of the data.

Clearly, these strategies require different kinds of security techniques. Information flow control, for example, is much more demanding of an operating system than is access control and may also involve special programming languages. It is also true that one security measure can contribute to more than one strategy. For example, preventing access by an unauthorized user also helps to protect the integrity of the data, because such a user may change the data improperly.

A later section of this paper discusses models of access control and information flow control. Inference control is discussed as applied to statistical data bases. Measures to protect application and system integrity are discussed later in various sections. This paper does not deal specifically with data integrity, but References 1 and 9 do introduce this important topic.

Sources of security threats

Consider threats to computer security from outside the computing system. The computing system hardware (including data storage devices) can be physically damaged by flood, fire, earthquake, sabotage, traffic accident, and so forth. The same events can likewise damage data stored away from the system on tapes, disks, or diskettes. Information may be accidentally destroyed if a wrong data volume, such

as disk pack or tape reel, is used on the system. Off-line storage can be stolen or copied, as can printed output. Communication lines are vulnerable to eavesdropping or to the insertion of unauthorized messages.

Without safeguards authorized users can perform improper actions either deliberately or by accident.

A person can gain unauthorized access to data by masquerading as a different person. An application program can be improperly modified by using the normal procedures for changing programs. If an application lacks adequate safeguards, its authorized users can perform improper actions either deliberately or by accident. An authorized user may act as a spy, passing restricted information outside the system.

The people who are sources of threats may have legitimate access to the system—such as application users, application programmers, system programmers, operators, system administrators—or they may be outsiders who succeed in penetrating the system.

Looking now within the computing system, we can have errors in application programs or operating systems, inadequate protection mechanisms in the hardware and operating system that result in failure to isolate user programs properly, or hardware failures. The immediate result is usually the unauthorized reading or writing of data in memory or on disk. That, in turn, may lead to a system crash with consequent denial of service, or theft or improper modification of data, theft of proprietary software, or other dire results.

Certain generic problems in system software may cause the system to fail in its protection against threats. One of these is known as the "TOCTTOU problem," which is derived from its longer name, the time-of-check-to-time-of-use problem.[10] The TOCTTOU problem results from an event such as the following. Information, such as a parameter of a request to the operating system, is checked and found valid, but the information is changed by the user before the system actually carries out the request. Another class of problem is termed the "residues problem." Here, when an area of memory is released by a user or when a file is deleted, the information stored in memory or on the disk may remain there, although it is inaccessible in the normal way. With skilled programming, that information can then be read by the next user to whom the space is allocated. Another problem that is not guarded against in current systems is the passing of information by covert channels. That is, information is passed using means other than the normal channels provided by the computing system. For example, a program may convey information to the operator by varying its speed of reading a tape. Also, one program may convey information to another program by varying its amount of computation and its use of memory. Thus the intelligently modulated rate of progress of the other program is the covert channel. We conclude that threats, both accidental and deliberate, come from all types of accessors of the system.

Before discussing security measures that can be taken to counter the various threats, we introduce models of security that are useful in describing the measures.

Security models

Access matrix model. The best-known security model is the *access matrix* model, which is described by Lampson.[11] The basic elements of the model are subjects, objects, and access types. The model grew out of work on operating systems, which is why each element can be interpreted in terms of operating system concepts. A *subject* is an active entity capable of accessing objects. In the operating-system context, a subject is a *process*, which is sometimes defined as a program in execution. In a time-sharing system, for example, a number of processes run concurrently on the same computer, sharing the memory and processor. Thus each process represents a different user. An *object* is anything to which access is controlled. Examples of objects known to an operating system are files, programs, and segments of memory. An *access type* is simply a kind of access to an object. For each type of object, there is a set of possible access types. Files, for example, have such access types as Read, Write, or Erase.

An access matrix M relates the three types of elements of the model as follows. In this matrix the rows represent subjects and the columns represent objects. Each cell M_{ij} contains a list of access types permitted to subject i for object j. These are sometimes called access rights, privileges, or permissions. Figure 1 shows an example of an access matrix in which Process 1 can Read and Execute Program 1 and can Read and Write Segment A. However, Process 1 has no access at all to Segment B. Process 2 can Read Segment B, but has no access to Program 1 or Segment A. Since operating system subjects and objects can be created and destroyed dynamically, and access rights change continually, the dimensions and contents of the access matrix also change.

The elements of the access matrix model can also be given interpretations at a different level. The subjects then become the users of a computing system, and they have rights to persistent resources such as application programs or data base objects.

For implementing access control, as opposed to using a model, it is generally inefficient to represent access-control information by a matrix, because the matrix is typically sparse. That is, there are many objects and subjects and relatively few rights, with the result that the access matrix has many voids or zero elements. Two ways are commonly used to store access-control information. An *access control list* associated with an object lists all the subjects who can access the object, along with their rights. A *capability list* associated with a subject lists all that subject's rights to all objects. Figure 2 shows the information of Figure 1 in access-list form, and Figure 3 shows the same information in capability-list form.

Figure 1 Access matrix

	PROGRAM 1	SEGMENT A	SEGMENT B
PROCESS 1	READ EXECUTE	READ WRITE	
PROCESS 2			READ

Figure 2 Access control list

ACCESS CONTROL LIST FOR PROGRAM 1:
PROCESS 1 (READ, EXECUTE)

ACCESS CONTROL LIST FOR SEGMENT A:
PROCESS 1 (READ, WRITE)

ACCESS CONTROL LIST FOR SEGMENT B:
PROCESS 2 (READ)

Figure 3 Capability list

CAPABILITY LIST FOR PROCESS 1:
PROGRAM 1 (READ, EXECUTE)
SEGMENT A (READ, WRITE)

CAPABILITY LIST FOR PROCESS 2:
SEGMENT B (READ)

Models using levels and compartments. A different type of model was developed by U.S. military services because they wanted systems that would enforce the military security policy. According to that policy, as described by Landwehr,[12] information is either unclassified or classified into *sensitivity levels* such as confidential, secret, and top secret. People are given *clearance* to access information up to a certain sensitivity level. Thus a person cleared for secret information could also access confidential and unclassified information, but not top secret information. The person must also have a *need-to-know* for the specific information accessed. In addition, some information also has one or more *compartment designations*, such as NUCLEAR, and access to such information requires clearance for all its compartments. Thus a *security level* consists of both a sensitivity or clearance level and a set of compartments.

The military policy has been formally specified as a first step toward the goal of demonstrating convincingly that computing systems correctly enforce the security policy, whatever the actions of programs and users. The best-known model is the one developed by Bell and LaPadula.[13,14] The subjects in this model again usually represent processes, and the objects represent files or other containers of information. One security level—call it A—*dominates* another level B when two conditions are true simultaneously: (1) A's classification or clearance level is greater than or equal to that of B, and (2) A's set of compartments contains those of B. The access types are the following: Read (observe only), Append (alter

only), and Write (observe and alter). The state of the system is described by the following: (1) the *current access set*, where each access includes subject, object, and access type; (2) an access matrix representing need-to-know; and (3) the security levels of all subjects and objects. The system state is changed by a *request*. The system's response to the request and the new state are determined by a *rule*. If it can be proved that each rule preserves security, so that any request results in a new secure state, the system is said to be *secure*.

A secure state is defined by the *simple security property* and the **-property*. The simple security property is as follows: For an "observe" access type, the level of the subject dominates the level of the object. In other words, there is no reading upward in level. The simple security condition does not prevent a spy with secret clearance from reading information from a secret object and writing it into a confidential object. The star property (*-property) prevents such writing downward, and is defined as follows:

- For Read: the subject's level dominates that of the object.
- For Append: the object's level dominates.
- For Write: the levels are equal.

A third property, *discretionary security*, requires every access to be explicitly authorized by the access matrix.

Information flow models. Landwehr points out that the Bell and LaPadula model is formulated in terms of access to objects rather than information flow. A model, termed the *lattice model*, described by Denning,[15] treats information flow more directly. The lattice model also generalizes levels and categories and their relationships. The model provides a basis for eventually analyzing source programs to determine whether they violate the information flow properties of a specific security structure.

Models implied by commercial systems. Many operating systems, data base management systems, and application systems provide access control facilities, and these usually imply an access matrix model. A user or group of users has the authority to specify the contents of the access matrix. Because the number of objects may be very great, this authorization function is usually distributed among different people. That is, for any object, some user has the right to specify the column in the access matrix or, equivalently, the object's access control list. A more detailed survey of security models can be found in Reference 12.

Security measures within the computing system

This section describes technical measures that can be taken within the computing system to promote security. Some of these measures have to do with the structure and design of the system and can be called *passive measures*. *Active measures* are steps taken in addition to the usual system processing.

The design of security measures should be simple and small to allow careful checking of its accuracy.

Principles of secure systems. Some quite general principles can be stated about how to design security measures in hardware, in various levels of software, and also in system administration. The discussion here is based on Reference 16 by Saltzer and Schroeder.

The design of the security measures embodied in both hardware and software should be simple and small so as to allow for careful checking of its accuracy.

The default situation should be of the *no-access* type in which access requires explicit permission. The implementation of this principle creates *fail-safe defaults*, which characterize a *closed* system as opposed to an *open* one.

Every access must be checked against the access-control information, including those accesses occurring outside normal operation, as in recovery or maintenance. This is termed the principle of *complete mediation*.

A lock mechanism that demands two keys for access is safer than one requiring only a single key. To illustrate this point, Saltzer and Schroeder use the

analogy of the two keys required to open a bank safe deposit box. Each key can be in the custody of a different component of the system. Then a single failure does not result in a security breach. This rule is known as the *separation of privilege.*

The Saltzer and Schroeder principle of *least privilege* states that every program and every user of the system should operate using the least set of privileges necessary to complete the job.

According to Popek's rule of *least common mechanism,*[17] the design should minimize the mechanism shared by different users for their mutual security.

In practice, encryption mechanisms are not completely public.

Such shared mechanism is crucial. Keeping it small and isolated helps to keep it correct.

Security mechanisms must be psychologically acceptable. They should not interfere unduly with the work of users, while at the same time meeting the needs of those who authorize access.

A final principle differs from the others in addressing not the design of the security system itself but rather its dissemination. It is generally believed that the design of a security system should be open rather than secret. Although encryption keys, for example, must be secret, the encryption mechanisms that use them should be open to public scrutiny. They can then be reviewed by many experts, and users can therefore have high confidence in them. In practice, encryption mechanisms are not completely public, and attempts are being made to control the dissemination of encryption research.[18]

Protection techniques in hardware and operating systems. Computing systems are typically shared by many users and many applications. The needs and privileges of these users and applications vary, and they differ as a whole from the needs and privileges of such system components as the operating system. The hardware provides protection features that isolate the executing programs from one another, protect the operating system from user programs, and allow only the operating system to perform such sensitive operations as physical I/O.

We begin our discussion with two of the most important and most universal of such features, states of privilege and virtual memory. The discussion makes use of the process concept introduced earlier in this paper in the discussion of the access matrix model.

States of privilege. Certain machine instructions are intended for use by the operating system only. These include, for example, I/O instructions and the instructions that control the protection features themselves. In most computers, these instructions are valid only when the processor is executing in a privileged state. On System/370,[19] for example, the supervisor state contrasts with the problem state, which is used for application programs. A machine can have a number of privileged states, and these states can be used for different operating-system functions that themselves vary in privilege. The VAX-11/780[20] has the following four states, called *access modes*: (1) Kernel, the most privileged state, which is used for interrupt handling and physical I/O; (2) Executive, for higher-level I/O functions; (3) Supervisor, for command interpretation; and (4) User. Multics[21] generalizes the state concept, providing a number of *rings* of privilege.

Virtual memory. Virtual memory, although its primary function is to expand the memory available to programs, is a valuable protection feature. The real memory of a computer provides a numbered sequence of cells that must be shared by all processes and all operating system components. A virtual memory is a corresponding sequence that can be used by a program as if it were real. The virtual memory seen by one process is not the same as that seen by another process. For example, the real memory might provide one million bytes, whereas each process sees a virtual memory of 16 million bytes. The virtual memories of the different processes either do not overlap at all or overlap at one end to allow the processes to share the system code that resides there. Thus one process has absolutely no access to the private data and code of another process.

The virtual memory is considered as divided into *pages,* in which the page size is 2048 or 4096 bytes on the System/370 and 512 bytes on the VAX-11/780.

The real memory is divided into *frames* of the same size. Page tables keep track of where each page is, either in a frame of real memory or on a storage device. Upon each memory reference, either the virtual address is translated by hardware into the corresponding real address or—if the page is not in real memory—an interrupt occurs. The needed page is then brought into real memory. Many systems (System/370, for example) also use a larger unit than the page, which is termed a *segment*.

A capability is a ticket that allows its holder to gain access to a specified object.

States of privilege and virtual memory can be used together to enforce appropriate access rights for processes. In the VAX-11 the page-table entry for each page specifies the kind of access (Write, Read, or none) that is allowed from each access mode. The access mode of a process changes during execution because it calls or returns from system procedures, and different pages become accessible.

Virtual machine systems. A still more powerful concept is that of the *virtual machine*. With a virtual machine system, such as VM/370,[22] each user has the illusion of commanding an entire computer, including a processor, memory, and I/O devices. All these virtual machines, however, are implemented by a single, real computing system. This structure provides a higher degree of isolation between users than virtual memory because the virtual machines are logically independent.

Capability systems. A different approach to protection uses *capabilities*. A capability can be described as a ticket that allows its holder to gain a specified type of access to a specified object. Capability protection is usually implemented by special hardware or microcode that interprets capabilities when they are used and prevents them from being wrongly copied or manufactured. Because capabilities can be passed around and stored, they make possible very flexible protection schemes. Flexibility, however, can also lead to difficulty in controlling and auditing the capabilities that have been given out and in selectively revoking capabilities. An example of a capability machine is the Cambridge CAP system.[23]

The kernel approach. A line of research by the U.S. Department of Defense that has resulted in several experimental operating systems is based on the concept of a *security kernel*, which is a relatively small portion of the operating system that is responsible for enforcing security policy. Inasmuch as the kernel mediates all accesses, flaws in other portions of the operating system do not threaten security. The kernel itself must be tamperproof. That is, there must be no way to modify it or interfere with its behavior. The kernel must be verifiable in that it is possible to demonstrate convincingly that the design correctly implements the system's security policy and that the programs of the kernel correctly implement the design. It is, of course, essential to the kernel approach that the security policy be concretely and formally stated. Nearly all of the kernel-based systems use the Bell and LaPadula model. At the present time, we know of no commercially available kernelized approaches to security.

One way of verifying a kernel is to prove that it is correct, and special languages and verification programs have been developed for this purpose.[24] Other ways involve careful scrutiny of the code by experts and penetration attempts by "tiger teams." From a security point of view, it is not necessary to demonstrate that the kernel is correct in all respects, but that it correctly implements security policy. Other tests, of course, are necessary to demonstrate its efficacy as an operating system component.

Hardware support for the kernel approach includes the following: the checking of each memory access and the provision of independent control for different types of access; the isolation of the kernel by such methods as a kernel mode or by implementation of the kernel in read-only memory; support for the process concept; and efficient switching between modes and between processes.

One operating system that takes a kernel approach is KVM/370,[25] which is based on the VM/370 virtual machine system. The functions of the VM/370 component that implements virtual machines—the monitor—are split between a security kernel and a set of nonkernel monitors, one per security level. Each

nonkernel monitor supports all the virtual machines at its level, and the kernel enforces the military security policy. Discussions of the kernel approach can be found in References 26 through 28.

User authentication techniques. A prerequisite for almost any kind of security is the accurate identification of users. By *authentication*, the system verifies the user's claim of identity. The most widely used authentication technique is the *password*, which is a string of characters known only to the system and to the user that the user must provide to gain access to a system. The system stores each user's password for comparison with the password presented by the user. Often a one-way transformation is used for the stored passwords, so that they are not intelligible even if they are accidentally printed. The system then applies the same transformation to the password supplied by the user before it is compared with the stored password. A password scheme is economical and acceptable to most users, and it is easy to implement.

Often a one-way transformation is used for the stored passwords.

A password scheme has a number of problems, however. Although the method depends on the secrecy of the password, it is common for users to write down passwords in exposed places or divulge them to others. Other persons can observe the keying in of a password. If the terminal is itself a computer, it can try many passwords in a relatively short time. A system can guard against this by allowing only a few erroneous attempts. Another method that has been used to learn a password is *spoofing*, by which a user of a time-sharing system writes a program that generates a display exactly like the system's sign-on display. The program is started, and the terminal is left to be used by a victim. The victim unknowingly communicates his or her password to the first user's program. With knowledge of the password, the spoofer can be authenticated in place of the victim.

Another authentication technique involves a machine-readable object possessed by the user, such as a card or badge. With such objects, there is the danger of loss, theft, or forgery of the object.

Promising techniques that are still in the research stage involve recognizing some characteristic of the user—voice, hand, fingerprint, or signature. Such a technology must be both accurate and cheap if it is to be widely used.

Logging. Logging consists of recording events so that they can be monitored at a later time. This record is called a log or audit trail. Although logging is a valuable technique for both deterring and detecting unauthorized actions, it does not prevent such actions. Logging can be performed by applications, by data base management systems, or by special access-control software. Operating systems sometimes provide basic logging facilities that can be used by these other components. A typical entry in a log might include the following: the user's identity; transaction or job identifier; name of the object being accessed (a file, for example); type of access; data values actually read or written; and date and time. Useful features in a logging facility include the following: ways to specify the events to be logged without actually programming the logging; ways to start and stop logging of selected events dynamically; and programs to generate reports from the log.

Encryption. Encryption predates computers by many centuries. The technique consists of *encrypting* or *enciphering* data by transforming them into a form that cannot be understood. The encrypted data are useful only to someone who possesses the special knowledge needed to restore them to their original form. Encryption can be used for data stored on such external media as tapes or removable disk volumes, for data transmitted over communication lines, and for data stored in the computing system.

The process of encryption takes a sequence of *plaintext*, P, applies to it an encryption procedure, E, which is controlled by a *key*, K, to produce a *ciphertext* C. To recover the original plaintext, the process applies a *decryption* procedure, D, controlled by the same key K. These processes may be expressed as follows:

Encryption: $C = E_K(P)$
Decryption: $P = D_K(C)$

In conventional encryption systems the key, K, is

secret, and the encryption and decryption procedures, E and D, are normally public.

The strength of an encryption system, that is, its resistance to being broken, can be described in terms of the kinds of attacks it can survive. The strongest type of attack is the "chosen plaintext attack," whereby the attacker can submit any amount of any plaintext and determine the corresponding ciphertext. Current encryption systems are designed to withstand chosen plaintext attacks.

In 1977, the U.S. National Bureau of Standards adopted the Data Encryption Standard (DES).[29,30] The DES uses the same algorithm for both encryption and decryption. It uses a 64-bit key (of which eight bits are for parity checking) to encrypt 64-bit blocks of plaintext. One reason for adopting a standard was to encourage the development of inexpensive implementations of the algorithm. A number of hardware DES devices are now marketed.

The DES is a *private key* system in that the keys are kept secret. One of the problems in private key systems is that of finding a way to securely distribute and maintain the keys. There have been proposed *public key* systems that make use of two keys or procedures: a public procedure, E, for encryption and a private procedure, D, for decryption. A public procedure, E, is associated with each subscriber to the system. The two procedures must have the property that for any plaintext, P, $D[E(P)] = P$. Also, it must not be possible to derive D from E. To send a message to subscriber A we encrypt with E_A. This encrypted message can be decrypted only by the possessor of D_A, namely A. Public key systems are quite promising and have important applications, such as digital signatures. However, research is still needed to develop practical algorithms that meet the requirements of the method.

Encryption, although an extremely valuable technique, does not solve all security problems. It cannot prevent destruction of data, and it is difficult to apply to data as used within the computing system. Its application in network security is described in a later section. More information about cryptography may be found in References 31–33.

Software packages for access control. A number of software packages provide access control and related functions. The market for these programs appears to be growing as users become more concerned about security. Examples are ACF2,[34] RACF,[35,36] and Top Secret.[37] Typical functions provided by such packages are authenticating users, maintaining access control information, checking authorization to use files or other objects, logging, and producing reports.

A passive intruder listens to the communications and an active intruder can alter or insert messages.

RACF, for example, allows files, storage volumes, applications, data base transactions, or user-defined resources to be specified as protected objects. RACF maintains the access control list for each object. Users may belong to groups and receive all the privileges of their groups. RACF is basically an *open system* in that resources not defined to RACF are not protected. ACF2 is a *closed system*, in which resources not defined to ACF2 are protected. RACF can, however, provide closed-system protection for any specified set of resources.

Communication and network security

Since users increasingly access computing systems from remote locations, careful attention must be given to communication security. Also increasingly important is security when connecting computers into networks. Because of their relatedness, we consider communication and network security together.

The transmission mechanisms used for data communications are vulnerable to two types of intrusion. A *passive intruder* listens to the communications, and an *active intruder* can alter or insert messages, or retransmit valid messages. Both types of intrusion can be accomplished through wiretapping, that is, by physically connecting to a communication path. Passive intrusion can be done by picking up microwave or satellite transmissions. Vulnerabilities also exist at switching centers, which are themselves computing systems, and in the interfaces of computing systems (*nodes*) to the network. These vulnerabilities are of great practical importance in applications such

as Electronic Funds Transfer (EFT),[38,39] where billions of dollars are transferred daily, and a single message can involve millions of dollars.

The following are some of the objectives of network security measures: protect privacy by preventing unauthorized listening to messages; authenticate users and messages; prevent disruption of network operation (which can occur through blocking message delivery, altering messages, or overloading the network); and assist in access control. Physical security measures such as buried cables can help, but the most important measure is encryption.

Use of encryption. One issue in the encryption of messages is the level of the computing system at which the encryption is done. The most efficient place to do encryption is just before the message goes out on the communication line. At that point, encryption can be done in conjunction with other manipulations, such as compression, packet formation, or check-sum calculation. This is called *data link encryption.*[40] A weakness of this method is that either a single key must be used for all communications between a pair of nodes, or some central authority must be entrusted with all users' keys. Also, security of a message depends on correct functioning of all the levels of system software that intervene between the user and the communication line. In *end-to-end encryption* the key is chosen by the user or application and is not divulged to other system components except the encryption mechanism itself. The key can be changed whenever there has been a chance of compromise. A discussion of end-to-end security measures is found in Reference 41.

One of the greatest difficulties in managing network security is *key distribution.* Typically, with secret-key systems, a key is required for each potential pair of communicators. Thus the number of keys is large, and, at the same time, they must be distributed in some secure way. One approach is a Key Distribution Center (KDC) that maintains all the keys and by prearrangement has a special key for communicating with each node on the network. When one node wants to communicate with another it asks the KDC to send session-keys to both participants. Such an approach is vulnerable to failure or congestion of the KDC. One possible refinement might be that of distributing the key-distribution function among all the nodes.

Public-key systems have similar problems. Here there is no need to distribute secret keys, but there is a need to keep the public listing of keys correct and up-to-date. The keeper of this listing must authenticate all changes. A sender who has been given the public key of a potential receiver needs assurance that it is correct. Key distribution is discussed in References 42 to 45.

Authentication. One of the most important communication security procedures is that of the authentication of users. For example, passwords can be compromised by passive intrusion if they are communicated in plaintext. One technique for overcoming this is for the system to call the user back by way of a list of telephone numbers that it keeps. This at least restricts access to authorized locations. Messages must also be authenticated. The authentication of a message means that the receiver of a message validates it in the face of possible message alteration or insertion. The use of digital signatures—discussed in the next section—is one possible way of authenticating both users and messages.

One of the most important communication security procedures is that of the authentication of users.

Digital signatures. A paper transaction, such as a check or order, is typically authenticated by a handwritten signature. Electronic transactions need *digital signatures.*[46] Various schemes have been developed to use encryption for this purpose. A digital signature scheme has several requirements. It should not be possible to forge a signature. The receiver must be able to validate the signature at the time the message is received and must also be able to demonstrate at a later time that a valid message was received. The sender should not be able later to repudiate the message.

Both public-key and conventional encryption have been proposed as the bases for signature schemes, and they are summarized in References 44, 45, and 47. In the public-key method, the sender encrypts the message with his own private key, and the re-

Figure 4 Employee table

NAME	DEPT.	SALARY	MANAGER
BALL	COMPUTER	35000	CHAN
CHOW	SHIPPING	18000	DIAZ
FOX	COMPUTER	39000	CHAN
KATZ	MAIL	19000	ROTH
WOOD	BUILDING	27000	LEE

Figure 5 Restricted view of employee table

NAME	DEPT.	MANAGER
BALL	COMPUTER	CHAN
FOX	COMPUTER	CHAN

ceiver decrypts it with the sender's public key. If the resulting message is intelligible, it is valid. Without additional refinements, this scheme does not prevent repudiation of messages, for example, by asserting that a key has been compromised and someone has forged the message. Neither does the method allow validation at a later time. The conventional scheme requires a central authority to encrypt and later authenticate signatures. Development of digital signature schemes is continuing. (See, for example, Reference 48.) The practical success of digital signatures depends not only on technology but also on the legal and procedural environments in which these signatures are used.

Security of local area networks. A local area network (LAN) consists of computers and related devices that are connected within a limited geographical area, such as one or a few buildings; it is usually privately owned. Although not as vulnerable as long-haul networks, local area networks cannot be regarded as secure. Cables or other transmission media are located throughout the local area and are thus subject to intrusion. Each node in the network must have a way to authenticate messages arriving from other nodes, especially if these messages are requests for data and services. Proposed security methods for LANs use encryption to provide for authentication and access control. Encryption is used to create *protected identifiers* that behave something like capabilities and that cannot be forged or used if stolen. One proposal uses public-key encryption[49] and another uses conventional encryption.[50]

Data base security

Data bases contain structured data that are maintained by a *data base management system* (DBMS). A DBMS is usually a separate software component that runs on top of the operating system and provides the additional functions to use the data base. A DBMS may also include functions to manage transactions. A DBMS assumes one or more data models upon which the data are structured, such as relations (tables), hierarchies, or networks. As an example of a data model, consider a tabular or relational arrangement as shown in Figure 4.

Data base applications typically require a fine *granularity of access control*, which means that access is controlled not according to tables as a whole but according to certain columns and rows of tables. This is sometimes called field-level access control. A DBMS usually provides its own access control, using the operating system only to protect the large containers (segments or files) in which the data are stored.

One way to provide field-level access control is through *views*, which can be constructed from one or more of the basic data base tables. A view can eliminate columns or rows. For example, the view in Figure 5 eliminates from the employee table of Figure 4 the SALARY column and all rows except those for employees in the COMPUTER department. Then each user is given access to required views only. Views that eliminate rows provide data-dependent access control, so called because the user's access to a specific row depends on the data values in that row.

Another way to provide fine granularity of access control is to encapsulate sets of allowed accesses in precompiled transactions and to grant users access to certain transactions only.

DBMS authorization facilities. Some DBMSs provide facilities that allow certain users to specify access control information. These users are termed *authorizers*, and they may be a central person or group. Alternatively, authorization may be decentralized, with each group or individual owning or controlling a portion of the total data base. For example, in Structured Query Language (SQL)[51,52] the user who creates a new table can perform any operation on that table, can grant another user any of these privileges, and can revoke the grant. The grant can also allow the recipient to grant the privileges to still another user.

Because the complex validations required in data base systems can degrade performance if not carefully implemented, considerable attention has been given to reducing overhead by designing the systems to do some of the work prior to execution time (at compile time, for example). More information about authorization and enforcement of data base security can be found in Reference 1.

Security of statistical data bases. This section summarizes research on the security of statistical data bases. A statistical data base is one that contains information about individual persons, which data must remain confidential, although statistical summaries, such as counts or sums, are freely available. Examples of statistical data bases are census data and medical research data. The threat to confidentiality in a statistical data base comes from the potential for drawing an *inference*. This means that a user may be able to correlate statistical summaries and his own prior knowledge, which may lead to compromise or disclosure.

The threat to confidentiality in a statistical data base comes from the potential for drawing an inference.

The data model used in statistical data base research uses a set of *records* for n individuals. Fields of the records contain values of *attributes* (such as sex, age, or salary). Users query the data base under the assumption that there is no change to the data base between queries of a potential intruder. We now introduce the following terminology:

- n: the number of records in the data base.
- C: a characteristic formula, such as SEX = 'MALE' AND AGE < 30.
- Query: "What is the average age of all males in the data base?" for example.
- Query set: the set of records satisfying C.

It is easy to compromise a data base when the query set size is small. For example, one can ask for the average age where NAME = 'JOE' and learn Joe's age. User Dave can ask for the average age of Joe and Dave. It might seem reasonable to control such compromises by requiring the query set size to be greater than some minimum. Any queries with the required query set sizes would be considered *answerable*. Set size, however, is not a sufficient condition. To see why, consider an intruder who develops a formula called a *tracker*, which is a method of allowing compromise in spite of limits for answerable queries. A tracker for a specific individual can be developed quite easily if the intruder has prior knowledge of an answerable query that uniquely characterizes the individual. Even a *general tracker*, which works for anyone in the data base, can often be guessed in a reasonable number of tries.

Certain defenses can be used against compromise by a tracker. One defensive technique is to perturb the data by adding to them (either before or after computing the statistic) a pseudo-random value that depends on the data. Another defense is to release only a random sample of the original data base. This technique is used successfully for census data, but it is not practical for use in a rapidly changing data base. Audit trails can detect, but not prevent, sequences of queries that attempt compromise.

Other kinds of defense are being studied. For example, *data swapping*, a technique under research, attempts to build a new data base containing records different from the original data base that produces the same statistics as the original data base. Another technique, called *random sample queries*, randomly determines which records are in a sampled query set and computes the statistic from this sampled set. More information about these techniques can be found in References 53 to 56.

In summary, a growing body of research has revealed that most statistical data bases are subject to compromise. This same research, however, is directed toward devising useful defenses against compromise.

Administrative security measures

This section surveys security practices that are primarily administrative in nature. Administrative security practices are covered in greater detail in Reference 57.

Physical security. Among the administrative measures are physical security practices, which include controlling access to sensitive areas, such as the

computing facility and the data processing department as a whole. Many organizations have a policy of not allowing anyone but operators into the computer room. Access to terminals, especially those used for sensitive applications, may also be controlled. Employee cooperation in enforcing the access restrictions is often preferable to elaborate locks and fences. Libraries for the storage of tapes and disks have a separate area with their own separate authorized personnel.

Classification of data. An organization needs an explicit policy for the confidentiality of data. Some companies define three or four levels of sensitivity and prescribe handling and disclosure procedures for data in each level. The policy should be made known in writing to each employee at hiring and periodically thereafter.

The auditing profession is deeply involved with security.

Personnel considerations. Appropriate care in hiring and dealing with employees is of course important for security. Administrative practices[57] that can enhance individual security include the following: ensuring that vacations are taken; periodically rotating assignments, but with an unpredictable schedule; providing grievance channels to allow employees to discuss sources of dissatisfaction without jeopardizing their positions; evaluating performance periodically with supervisors trained to recognize such danger signals as refusal of vacations or promotions, alcohol, or gambling; employment termination procedures (avoiding layoffs, if possible, and carrying out terminations fairly, exit interviews, the changing of passwords, and notification of other employees). In general, these practices are directed at avoiding situations where employees are motivated to misuse computers and to interrupt their access to computers on a schedule that is not entirely within their control. Some security advisors advocate the prosecution of employees who have embezzled and the dismissal of those who have violated security policy. Inasmuch as security is not an organization's only objective, it may be a valid choice to seek an optimum position between security and a trusting attitude in employee relations. Therefore, based on the belief that employees will do the right thing if they clearly know their responsibilities, counseling often takes precedence over immediate dismissal.

Auditing and controls. The auditing profession is deeply involved with security, and there is a specialty within auditing, called EDP audit, that deals with computerized aspects of systems. The internal auditors employed by an organization maintain the organization's system of internal controls, and one function of the external auditors is to conduct periodic reviews of those controls. Controls include a wide variety of measures, many of which can be viewed as security measures. More information about audit and control in computer environments can be found in References 1 and 57 to 59.

Computer security auditing aims at identifying and evaluating the security measures for a specific installation. Rahden[60] lists five types of computer security auditing: (1) system development audit of the procedures which are intended to ensure that only secure systems are developed; (2) application review of the security controls in the design of a specific application; (3) installation security review of all controls of the installation (administrative, technical, or physical); (4) security function review of all generalized security functions that apply to multiple departments or applications, such as those of a data base management system; and (5) controlled test or penetration study to demonstrate security weaknesses.

Implementing a security program. In this section we discuss how an organization sets up a security program. The thoughts presented are based in part on IBM security practices.[61,62]

The first step is to establish a policy about information assets and computer security. This policy serves to set direction and to guide management and all employees. The keystone to such a policy is the enthusiastic support of the chief executive officer. The policy is presented to all line and staff units, and it is reviewed regularly and updated as needs and methods change.

In content, a security policy defines broad responsibilities for functional management, owners of information, users of information, and providers of ser-

vice. More detailed guidelines and instructions are covered separately.

A security management function is established, involving one or more management and staff people, depending on the size and complexity of the organization. This security management function coordinates the organization-wide security program. Direct responsibility for implementing the program belongs to line management. The security manager assists line managers in interpreting and implementing the policy for their units.

If the security program is to work, employees must understand the need for it and must support it. Therefore, an education program is designed and presented to the employees. The education program includes the training of managers, present employees, and new employees as they are hired. Employees are also updated on changes as organizational needs and security methods change.

An important step is to classify the data according to their sensitivity, and to establish a written policy on handling data of each type. Also, all data should have an *owner* who is responsible for their protection. Many organizations tend to leave the responsibility for security with the data processing organization. This is inappropriate, because it is the owners and users of the data who suffer the losses if something happens to their data.

Another important step is a *risk analysis*.[63] Risk analysis, as described in Reference 64, has two main aspects: (1) analyzing threats, and (2) identifying the undesirable events that can result. Threat analysis also identifies the security weaknesses that can permit each threat. Consider, for example, the threat that an unauthorized user may access the system from a remote terminal by means of a password found on a printout in the trash. Security weaknesses in this example are (1) inadequate physical security of the terminal; (2) failure to suppress printing of passwords; and (3) inadequate physical security of sensitive trash. These undesirable events are usually categorized as follows: unauthorized disclosure, modification and destruction of data, and denial of service.

Once the risks have been analyzed, a *risk assessment* is carried out. The undesirable events are rated and ranked according to severity. These ratings are not mathematically precise; rather, they put risks into a priority order. The likelihood of each event is then related to its acceptability. A program of security measures is then developed, with the cost of each measure being considered in relation to the losses it is intended to prevent. All this information can then be used as the basis for management decisions about a security program. After such a program has been implemented, it must be monitored and evaluated periodically for effectiveness.

There has been considerable success in developing solutions to security problems.

Concluding remarks

This paper has summarized the main threats to computer security. We have introduced models that provide a conceptual framework and surveyed the technology that is developing to counter the threats. Outlined are practical steps that organizations can take to protect their data and systems from unauthorized access. Discussed to a lesser degree is denial of service. We have indicated that security is a management issue as well as a technological one. As for management, it must define its expectations, explore and quantify its risks, select and implement protective measures, and follow up by continuing to evaluate the effectiveness of its policies and measures.

There is growing awareness of security, and there has been considerable success in developing solutions to security problems. However, security rarely receives the priority it warrants. Computers have become central in our social and economic lives, and rapidly changing technology is introducing new vulnerabilities along with new kinds of uses. The sponsors and designers of the new uses most especially need to foresee and avoid access and service vulnerabilities.

Cited references

1. E. B. Fernandez, R. C. Summers, and C. Wood, *Database Security and Integrity*, Addison-Wesley Publishing Company, Reading, MA (1981).
2. *Personal Privacy in an Information Society, Report of the Privacy Protection Study Commission*, U.S. Government Printing Office, Stock No. 052-003-00395-3, Washington, DC (July 1977).

3. R. Turn, *Trusted Computer Systems: Needs and Incentives for Use in Government and the Private Sector,* The Rand Corporation, Santa Monica, CA (June 1981).
4. "OECD guidelines governing the protection of privacy and transborder flows of personal data," *Computer Networks* **5**, No. 2, 127–141 (April 1981).
5. H. Baruch, "The Foreign Corrupt Practices Act," *Harvard Business Review* **57**, No. 1, 32–50 (January-February 1979).
6. J. K. Taber, "A survey of computer crime studies," *Computer/Law Journal* **2**, No. 2, 275–327 (Spring 1980).
7. D. B. Parker, "Vulnerabilities of EFTs to intentionally caused losses," *Communications of the ACM* **22**, No. 12, 654–660 (December 1979).
8. D. E. Denning and P. J. Denning, "Data security," *Computing Surveys* **11**, No. 3, 227–249 (September 1979).
9. C. J. Date, *An Introduction to Database Systems,* Vol. II, Addison-Wesley Publishing Company, Reading, MA (1983).
10. W. S. McPhee, "Operating-system integrity in OS/VS2," *IBM Systems Journal* **13**, No. 3, 230–252 (1974).
11. B. W. Lampson, "Protection," *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems* (1971), pp. 437–443. Reprinted in *ACM Operating Systems Review* **8**, No. 1, 18–24 (January 1974).
12. C. E. Landwehr, "Formal models for computer security," *Computing Surveys* **13**, No. 3, 247–278 (September 1981).
13. D. E. Bell and L. J. LaPadula, *Secure Computer System: Unified Exposition and Multics Interpretation,* Report ESD-TR-75-306, MITRE Corporation, Bedford, MA (March 1976).
14. J. K. Millen, "Security kernel validation in practice," *Communications of the ACM* **19**, No. 5, 243–250 (May 1976).
15. D. E. Denning, "A lattice model of secure information flow," *Communications of the ACM* **19**, No. 5, 236–243 (May 1976).
16. J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE* **63**, No. 9, 1278–1308 (September 1975).
17. G. J. Popek, "A principle of kernel design," *AFIPS Conference Proceedings* **43**, National Computer Conference (1974), pp. 977–978.
18. P. J. Denning, "A scientist's view of government control over scientific publication," *Communications of the ACM* **25**, No. 2, 95–97 (February 1982).
19. *IBM System/370 Principles of Operation,* GA22-7000, IBM Corporation; available through IBM branch offices.
20. W. D. Strecker, "VAX-11/780—A virtual address extension to the DEC PDP-11 family," in *Computer Structures: Principles and Examples,* D. P. Siewiorek, C. G. Bell, and A. Newell, Editors, McGraw-Hill Book Co., Inc., New York, 1982, pp. 716–729.
21. J. H. Saltzer, "Protection and the control of information sharing in Multics," *Communications of the ACM* **17**, No. 7, 388–402 (July 1974).
22. L. H. Seawright and R. A. MacKinnon, "VM/370—A study of multiplicity and usefulness," *IBM Systems Journal* **18**, No. 1, 4–17 (1979).
23. M. V. Wilkes and R. M. Needham, *The Cambridge CAP Computer and Its Operating System,* Elsevier North-Holland Publishing Co., New York (1979).
24. M. H. Cheheyl, M. Gasser, G. A. Huff, and J. K. Millen, "Verifying security," *Computing Surveys* **13**, No. 3, 279–339 (September 1981).
25. B. D. Gold, "A security retrofit of VM/370," *AFIPS Conference Proceedings* **48**, National Computer Conference (1979), pp. 335–344.
26. S. R. Ames, M. Glasser, and R. R. Schell, "Security kernel design and implementation: An introduction," *Computer* **16**, No. 7, 14–22 (July 1983).
27. R. R. Schell, "A security kernel for a multiprocessor microcomputer," *Computer* **16**, No. 7, 47–53 (July 1983).
28. C. E. Landwehr, "The best available technologies for computer security," *Computer* **16**, No. 7, 86–100 (July 1983).
29. *Data Encryption Standard,* National Bureau of Standards, Washington, DC, FIPS Publication 46 (January 1977).
30. W. F. Ehrsam, S. M. Matyas, C. H. Meyer, and W. L. Tuchman, "A cryptographic key management scheme for implementing the Data Encryption Standard," *IBM Systems Journal* **17**, No. 2, 106–125 (1978).
31. W. Diffie and M. E. Hellman, "Privacy and authentication: An introduction to cryptography," in *Tutorial: The Security of Data in Networks,* D. W. Davis, Editor, IEEE Computer Society, 5855 Naples Plaza, Suite 301, Long Beach, CA 90803 (1981).
32. A. Lempel, "Cryptology in transition," *Computing Surveys* **11**, No. 4, 285–303 (December 1979).
33. D. E. Denning, *Cryptography and Data Security,* Addison-Wesley Publishing Company, Reading, MA (1982).
34. D. Botnick, "Mellon Bank prepares staff to receive data security package," *Bank Systems and Equipment* **20**, No. 6, 72–73 (June 1983).
35. *OS/VS2 MVS Resource Access Control Facility (RACF) General Information Manual,* GC28-0722, IBM Corporation; available through IBM branch offices.
36. H. M. Gladney, "Administrative control of computing service," *IBM Systems Journal* **17**, No. 2, 151–178 (1978).
37. J. D. Whalen, "A false sense of security," *Infosystems* **30**, No. 8, 100–103 (August 1983).
38. B. Bosworth, "Computer and EFT system security," *Review of Business* **4**, No. 2, 9–11, 30 (1982).
39. B. Streeter, "People, more than technology, are still key to EFT security," *ABA Banking Journal* **74**, No. 7, 29–37 (July 1982).
40. A. S. Tanenbaum, *Computer Networks,* Prentice-Hall, Inc., Englewood Cliffs, NJ (1981).
41. V. L. Voydock and S. T. Kent, "Security mechanisms in high-level network protocols," *Computing Surveys* **15**, No. 2, 135–171 (June 1983).
42. S. M. Matyas and C. H. Meyer, "Generation, distribution, and installation of cryptographic keys," *IBM Systems Journal* **17**, No. 2, 126–137 (1978).
43. R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Communications of the ACM* **21**, No. 12, 993–999 (December 1978).
44. G. J. Popek and C. S. Kline, "Encryption and secure computer networks," *Computing Surveys* **11**, No. 4, 331–356 (December 1979).
45. D. E. Denning, "Protecting public keys and signature keys," *Computer* **16**, No. 2, 27–35 (February 1983).
46. S. G. Akl, "Digital signatures: A tutorial survey," *Computer* **16**, No. 2, 15–24 (February 1983).
47. S. M. Matyas, "Digital signatures—An overview," *Computer Networks* **3**, No. 2, 87–94 (April 1979).
48. H. Meijer and S. Akl, "Digital signature schemes for computer communication networks," *Proceedings of the 7th Data Communications Symposium,* 1981, pp. 37–41; available from the Association for Computing Machinery, Inc., 11 West 42nd Street, New York, NY 10036.
49. J. E. Donnelly and J. G. Fletcher, "Resource access control in a network operating system," *Proceedings of the ACM Pacific Regional Conference* (1980), pp. 115–126.
50. D. M. Nessett, "Identifier protection in a distributed operating system," *Operating Systems Review* **16**, No. 1, 26–31 (January 1982).

51. *SQL/Data System: Planning and Administration,* SH24-5014, IBM Corporation; available through IBM branch offices.
52. D. J. Haderle and R. D. Jackson, "IBM Database 2 overview," *IBM Systems Journal* **23**, No. 2, 112–125 (1984).
53. D. E. Denning, "Secure statistical databases with random sample queries," *ACM Transactions on Database Systems* **5**, No. 3, 291–315 (September 1980).
54. J. Schlörer, "Disclosure from statistical databases: Quantitative aspects of trackers," *ACM Transactions on Database Systems* **5**, No. 4, 467–492 (December 1980).
55. J. Schlörer, "Security of statistical databases: Multidimensional transformation," *ACM Transactions on Database Systems* **6**, No. 1, 95–112 (March 1981).
56. D. E. Denning and J. Schlörer, "Inference controls for statistical databases," *Computer* **16**, No. 7, 69–82 (July1983).
57. L. I. Krauss and A. MacGahan, *Computer Fraud and Countermeasures,* Prentice-Hall, Inc., Englewood Cliffs, NJ (1979).
58. *Systems Auditability and Control Study,* Report of the Institute of Internal Auditors, Altamonte Springs, FL (1977).
59. R. P. Fisher, *Information Systems Security,* Prentice-Hall, Inc., Englewood Cliffs, NJ (1984).
60. H. R. Rahden, "Computer security auditing," *WESCON 1979 Conference Record* **14**, No. 3, 345–351 (1979). Reprinted in *Advances in Computer System Security,* R. Turn, Editor, Artech House, Dedham, MA (1981).
61. *Staying in Charge, An Executive Briefing for Improving Control of Your Information System,* G505-0058, IBM Corporation; available through IBM branch offices.
62. *Information Systems Security: Execution Checklist. Security is a Management Issue,* GX20-2430, IBM Corporation; available through IBM branch offices.
63. S. Fordyce, "Computer security: A current assessment," *Computers and Security* **1**, No. 1, 9–16 (January 1982).
64. R. P. Campbell and G. A. Sands, "A modular approach to computer security risk management," *AFIPS Conference Proceedings* **48**, *National Computer Conference,* AFIPS Press, Arlington, VA (1979), pp. 293–303.

General reference

K. S. Shankar, "The total computer security problem: An overview," *Computer* **10**, No. 6, 50–73 (June 1977).

Reprint Order No. G321-5227.

Rita C. Summers *IBM Los Angeles Scientific Center, 11601 Wilshire Boulevard, Los Angeles, California 90025.* Since joining IBM in 1964, Ms. Summers has designed and implemented systems for interactive applications, computer-assisted instruction, and numerical control. She has received two IBM Outstanding Contribution Awards for her work on virtual memory systems. Currently, Ms. Summers is a senior programmer and project leader for the development of a prototype resource-sharing system for personal computers. Her work in the area of computer security includes leadership of a project that developed a design for a secure data base system. She has participated in the development and teaching of courses on data base security in both IBM and universities, and is the author of technical reports, conference papers, and articles on data base security. Ms. Summers is a coauthor with E. B. Fernandez and C. Wood of the book *Database Security and Integrity,* published by Addison-Wesley Publishing Company, Reading, MA (1981). Before joining IBM, she worked as a systems analyst and programmer at the Ramo-Wooldridge Corporation. She received her B.A. and M.A. degrees from the University of California at Los Angeles.