This page is intentionally left blank.

Chapter 12

Protection of Privacy

Balyard had stopped off at the local personnel office for the name of a representative American in the neighborhood.
The personnel machines had considered the problem and ejected the card of E. R. B. Hagstrohm, who was statistically average in every respect save for the number of his initials: his age (36), his height (5'7"), his weight (148 lbs.), his years of marriage (11), his I.Q. (83), the number of his children (2: 1 m.9; 1 f.6), his car (3 yr. old Chev. 2 dr. sed.), the number of his bedrooms (2), his education (h.s.grad., 117th in a class of 233 . . .
.

Kurt Vonnegut, Jr.
Player Piano

There are two aspects to the issue of protection of data. We will try to cope with both of them in this chapter.

The first, and commonly understood, aspect is that we wish to *deny access* to data to those people who do not have a right to access these data. This is also commonly referred as the *protection of privacy* for personal data and *maintenance of security* for governmental or corporate data.

The second, but equally important, aspect of protection is that we have to *guarantee access* to all relevant data to those people who exercise their access privilege properly. This means that owners of databases also have the responsibility to protect the data entrusted to them. Part of this responsibility means that there has to be a reliable computer operation. The other aspect is that data has to be protected from exposure, vandalism, or alteration.

Some examples may help to define the scope of the protection problem.

Individual Privacy In order to estimate the next month's operational cost at the Consolidated Iron Works, a program is written which uses personnel data to extrapolate costs from wages, normal working hours, and outstanding vacation days. Lucy Brown, who prepares the report, should not have access to the staff's psychiatric histories, which are kept on-line with the personnel data.

Professional Privacy Fred Fierce, a manager of the Widget Division of Everything International Corporation, has access to all operational data of the Widget factory which have been collected into the corporate database. Fred is in personal competition with the manager, Shelley Smart, of the Gadget Division, since he feels a promotion is imminent. Should they have access to each other's data? How can we distinguish similar data from different sources? If we cannot guarantee that we can keep divisional data separate, there will be the temptation to withhold data from the corporate database.

Completeness of Data Victor Velocity is stopped on the highway for speeding. An inquiry into the central police computer reveals that he is listed as *absent without leave* from his army unit. He is promptly arrested. Only after a few days can his lawyer prove that Victor rejoined his army unit after a week's absence and has since then been properly discharged. The later data had not been entered into the police files.

Audit Trail A credit inquiry by an oil company to determine whether to issue a credit card to Sam Salesman determines that he frequently has been late in settling his accounts with Hal's Haberdashery, even though now he is in the clear. When Sam himself inquires into his credit standing, he is told that all is okay, and that the credit bureau cannot understand why he did not get his credit card from Global Energy Inc.

Vandalism A programmer who has just been fired from Wholesome Wholesalers goes back to his terminal and deletes the accounts receivable file of the company.

Fraud The president of Super Innovative Systems modifies the data in the company database so that results for this year look very good. This is accomplished by moving committed expenses into next year so that auditors fail to note the discrepancy. The resulting increase in the price of SIS stock enables the president to sell his equity at a high profit and quit.

A Counterargument There is a question about privacy which goes beyond the problems posed in the examples above. We will present the argument, but its resolution is outside the scope of this book. The question concerns the social value of protecting privacy by mechanical means. The aspect of privacy which we attempt to protect here concerns itself only with access to facts, so that the more sensitive issue of privacy of thoughts should be kept separate.

We consider first that the restriction of access to facts will provide more benefits to those persons who have facts to hide, and second that any mechanism to protect privacy will be easier to defeat by those with great political or economic power than by average members of the population. A completely open society, without secrets, without privacy of facts, would provide an incentive to a great straightforwardness in human interaction, and would avoid the augmentation of existing centers of power with the power that results from differential access to data.

Data in such an environment would no longer have the questionable information value associated with secrecy. Databases would still be important for daily decision making and long-range planning.

12-1 COMPONENTS OF THE PROTECTION PROBLEM

Three types of elements combine to form the system which we use to discuss methods of protection:

1 The accessors

2 The type of access desired

3 The objects to be accessed

Each of these elements has to be propeerly identified in order to achieve control of access to data. It is also necessary to consider the *envelope*, or the boundary of the area within which the protection system is valid. Sections 12-2 to 12-5 will define these aspects one by one. This section will discuss various general topics in more detail.

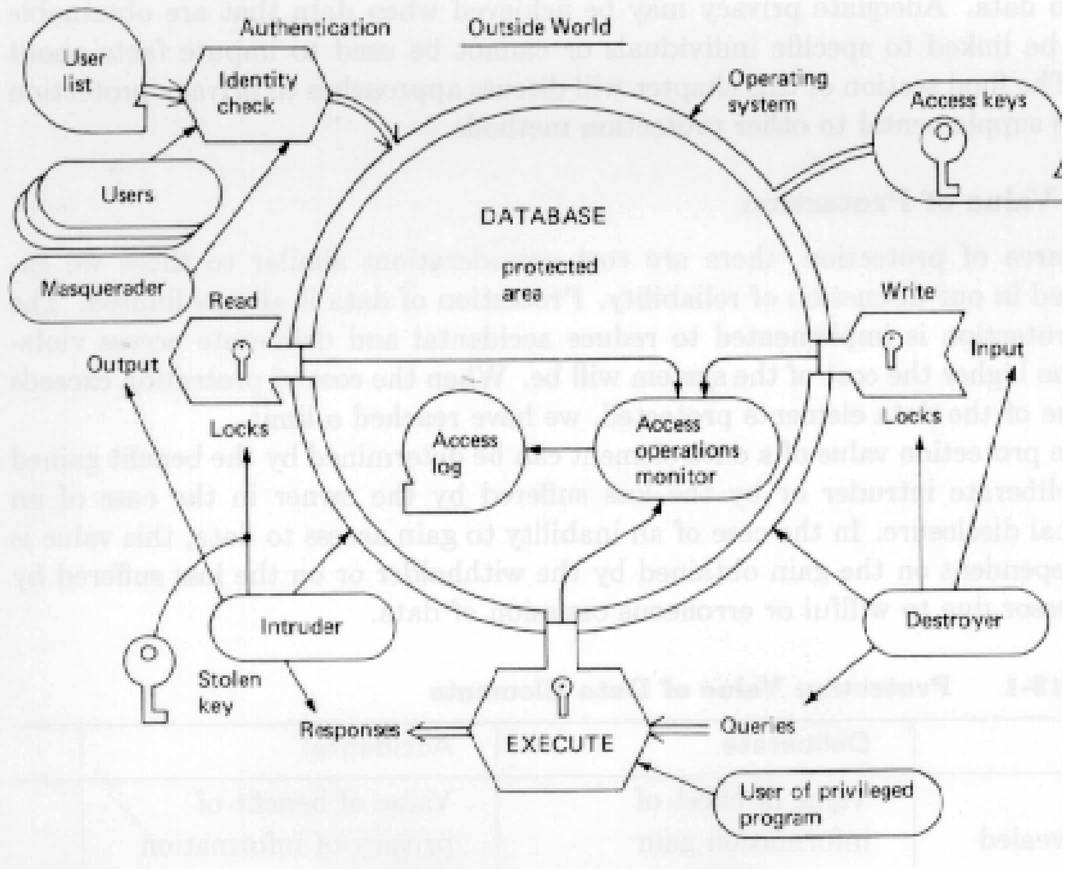


Figure 12-1 Elements of the protection problem.

Definitions We define a number of terms to clarify subsequent discussion of the mechanisms for achieving protection:

Envelope There is an area with a well-defined perimeter known as the database system.

Users and Intruders Within this area there may be active properly identified, *authenticated* individuals, individuals masquerading as valid users, and intruders.

Limited Scope The identity of individuals in the outside world is not known to the system.

Privileges Related to an individual's identification are various privileges of data access. The description of these is maintained as part of the database system.

Protection All data objects are protected to a specified extent while inside the database system area, and lose all system-provided protection when moved out of the area.

Reliability A high level of system reliability is a prerequisite to achieve protection of the database.

Protection, Privacy, and Secrecy Protection of data requires control over the reading, writing, and use of the data. Many of the methods which we will discuss do not distinguish between the protection of data, the maintenance of privacy, and the assurance of secrecy. Secrecy is achieved when there is no unauthorized access at all to data. Adequate privacy may be achieved when data that are obtainable cannot be linked to specific individuals or cannot be used to impute facts about them. The final section of this chapter will discuss approaches to privacy protection that are supplemental to other protection methods.

12-1-1 Value of Protection

In the area of protection, there are cost considerations similar to those we encountered in our discussion of reliability. Protection of data is always limited. The more protection is implemented to reduce accidental and deliberate access violations, the higher the cost of the system will be. When the cost of protection exceeds the value of the data elements protected, we have reached a limit.

The protection value of a data element can be determined by the benefit gained by a deliberate intruder or by the loss suffered by the owner in the case of an accidental disclosure. In the case of an inability to gain access to data, this value is again dependent on the gain obtained by the withholder or on the loss suffered by the accessor due to willful or erroneous omission of data.

Table 12-1 Protection Value of Data Elements

	Deliberate	Accidental
Data revealed	Value of effect of information gain to intruder	Value of benefit of privacy of information to owner
Data withheld	Value of effect of information loss to withholder	Value of benefit of use of information to owner

The rules summarized in Table 12-1 are based on the assumptions that any deliberate attempts on privacy are limited by the potential payoff, but that accidental release of information is not so limited. If the manager of the database has the responsibility to maintain the data, costs will be incurred when damaged or lost data have to be replaced. We can assume that the cost the manager will have to bear is limited by the value of the data to the owner.

It is obvious that the substitution of dollars into these value categories may be considered impossible. This problem of value measurement occurs in any area where decisions have to be made involving trade-offs between humanistic values, or utilities, and mechanical values, or costs. In order to achieve quantitative measures of the value of a data item to its owner or to an intruder, interrogation techniques have been used. By comparing the loss of items which have an assessable value with the loss felt due to a violation of privacy, an approximate value for data can be established.

A typical question in such a process could be

‘Would you request credit to buy this automobile if it revealed to the public your record of alcoholism?”

For many people, this choice would be simple and affirmative; to some people, however, it would be negative unless the financial loss becomes too great. To give a value to privacy protection for the category of alcoholism information, the maximum value determined will have to be used. These techniques have been applied in a major study by the State of Illinois [IBM G320-1373] during a project which demonstrated the use of a secure operating system, IBM RSS.

The relationship between the quantity of data items lost in a violation of privacy and the information loss may be highly nonlinear. In one instance, the loss of a single fact might be highly incriminating in one instance, and loss of additional data may not matter much afterward. In other cases, the more traditional assumptions that the information value increases with the number of data items released will be valid. The calculation of expected loss aggregates all types of risks and their probable costs for a given system approach. This then provides a basis to assess the cost-to-benefit ratio. In the study a loss reduction by a factor of 50 was feasible, but the optimum was at a factor of 30.

Responsibility It is not clear who, in a legal sense, is responsible for losses suffered because of violation of rules of protection of privacy. In many operations the reponsibility rests on top management, although specific areas may be delegated to data-processing management.

If programming is to be regarded as a professional activity, some responsibility will have to be borne by the members of the profession. The extent of responsibility eventually assigned to data processing and its management will depend on the public perception of the behavior of this group in the operation of database systems. The development of ethical standards for data-processing personnel is an on-going concern for the professional societies.

12-1-2 Cost of Protection

To achieve protection there will be costs to create physical security (building, personnel supervision), costs to develop adequate software, and increased operational cost. We will consider here mainly the operational cost.

The cost of providing operational protection is high in traditional computers because computer hardware design has stressed capability and efficiency rather than protection. This preoccupation has had the result that costly software mechanisms are required to implement even marginal levels of security.

An example of lack of protection is the degree of accessing freedom accorded to operating systems. Processes executed by the operating system kernel have often unlimited accessing privileges so that they can control other processes efficiently.

Since resource control is a responsibility of the operating system, many services for the users' computations are performed by the operating system. In systems where kernel and service routines are not separated, a very wide path of information flow exists between protected and unprotected areas. Such a path may be misused through interspersion of inappropriate information into the stream of data. To protect this path, many validation procedures are required, measurably reducing computer performance. The complexity of the protection mechanism causes unreliability without assuring *confinement* of a malicious and knowledgeable intruder.

The value of adequate protection in commercial systems has been estimated at 10 to 20% of basic data-processing cost. One experiment within RSS [IBMG320-1376] caused an increase of 4.2% in file access cost but also limited flexibility and sharing. The total cost of implementation of a truly secure database is today larger by an order of magnitude than its benefit, since it requires development of substitute system programs and the training of personnel to monitor operations. Additional equipment may be needed for redundancy and logging.

Protection of privacy and system security have now been recognized as important concerns, so that we can expect that new systems will be designed with better support for protection. The cost to the user should then become less so that the benefits of a reasonable protection level outweigh the costs.

12-2 THE ACCESSOR

The external identification of accessors is primarily the name, as entered by them into the system. An accessor may also be identified by a *password*, which has to be typed in on request, or perhaps by a machine-readable key or badge. Methods have been proposed and tested which depend on the biological uniqueness of human beings. Manifestations of this unique coding of individuals are voice prints, fingerprints, and signatures. The database system, as defined here, will not be responsible for the primary decoding and validation or *authentication* of the presented information. Since access to operating-system services is a prerequisite to use of the database, the authentication task is left to modules of the operating system. The method used for authentication of an individual's identification depends very much on the technology available in a specific instance. The authentication subsystem will present to the database system a string of bits which we will consider the access key. This module has to prevent a *masquerader* from obtaining a key. All authorization and privileges given to accessors will depend on the access key.

12-2-1 Keys

To assure that access keys are not available to unauthorized accessors, the *individual's identification* has to be very difficult to mimic or copy. While an individual's name may be unique, this name is easy to mimic by anyone who can observe those who have access to the system, and is hence not suitable for a key. Once a system provided key is obtained this access key is used to enter the database system from the operating system. The responsibility for manipulating the key rests with both the accessor and the operating system.

To protect the process of obtaining a key the system, when the user logs in, requests a password with the users' name. The password is entered without displaying it to protect it from observers. The password will generally consist of a few letters, chosen by the user. A trial-and-error method of entering possible passwords could be used by an intruder to gain access. The time required to carry out a systematic trial is the principal discouragement to potential intruders. The expected time to open a specific lock without any prior knowledge is

$$T(\text{getin}) = \tfrac{1}{2}c^d \; t(\text{try}) \qquad 12\text{-}1$$

where c^d is the number of possible combinations and $t(\text{try})$ the time required to try a combination. For a three-letter password, $d = 3$ and $c = 26$ and the time for the interaction with the system may be $t(\text{try}) = 3$ seconds, so that $T(\text{getin}) \approx 7$ hours. If the authentication process is required only infrequently, an artificial delay in the opening process can increase the safety of the lock. An equivalent method is to allow only two or three attempts to enter a password. An intruder then has to go again through a full access procedure, maybe requiring about 20 seconds. The procedure can be more strict for access to critical data. When the security-control officer using RSS fails to provide his password correctly twice, the entire computer system is shut down. Extending the number of combinations is the other alternative to increase safety. A practical problem is that long passwords are harder to remember and will be frequently written down; another problem is that passwords are frequently chosen to be initials, middle names, friends' names, or other *aides-mémoire* and hence are easily guessed.

In order to establish legal responsibility, the accessors who have received a password giving them valuable privileges may be required to sign an agreement to keep the password confidential, and to promptly report any suspected loss of confidentiality. A frequent change of passwords can help limit the improper use of leaked passwords. Changing passwords puts an additional burden on users unless they are equipped with some automatic password generator which is matched by the receiving system. If we invent such a device, we also will have to protect its use.

Methods beyond passwords are used to increase protection. An *active authentication* procedure is provided by NCSS. A user can place an interrogation routine in the system which requests parameters during the authentication process. A possible query presents a number to the accessor, who is expected to multiply the number by 3 and add 12. This procedure is the "password" the user has to remember. Authentication methods based on keycards, badges, and so on can positively identify the card and hence the accessor, unless the card is copied, lost, or stolen. A card in combination with a password can be very effective.

An individual who has valid access to the system may, after initial entry, use the identification key to gain access to inappropriate privileges. An accessor may also be defined as a member of two projects. It may not be desirable that this user can have both corresponding access rights at the same time. These problems requires careful management of access privileges.

The existence of masquerading intruders may be determined a posteriori by reporting all access attempts in a given period to the owner of a file. To achieve faster detection, the file user can be informed on every access of the date, time, and a sequence number of the previous access.

Up to this point, we have characterized the accessor as an individual. Sometimes access authority is given to classes of individuals. Clerical personnel, for instance, may not be individually identified. Authentication by class is not very secure, since there is no individual responsibility.

Additional parameters may be added to authenticate the accessor more completely. The database system proper may remain relatively innocent of these further specifications, since these will be translated into separate access keys; the number of valid access keys may become much greater.

12-2-2 Programs as Accessors

Among the additional accessor attributes to be considered is the *program or process* that is used to access the data. There may be protected programs that are maintained and controlled by one set of accessors, and used by others. Use of such a program may confer unto the user some of the privileges accorded to the controller of such programs. The transfer of privilege is appropriate if the program acts as a filter of the information. Typical of such filters are programs which perform statistical analysis on data. A program which reads data from a set or subset of records and produces only an average of the data performs such a filtering function. The average value may well be appropriate for a general audience, whereas the details of the data are confidential. This program should refuse to produce a result if the number of records selected to provide data for the resulting average is so small that the result can be associated with an individual data record.

A query by a clerk of the United Tamale Corporation to obtain *"the average salary of all employees with a Beverly Hills zipcode in their address"* should be aborted since it would provide, in effect, the salary of the president of the company.

The concerns of *statistical access* to individual records arise mainly in the maintenance of larger demographic databases, such as health records or census data. It remains possible that, through multiple intersecting queries, some confidential data is retrievable. The result obtained directly by the above query can also be obtained from the average salary of all employees and the average of all employees who do not live in Beverly Hills. The transaction-log which was discussed in the previous chapter provides at least an ex post facto audit trail for such access attempts. We will discuss in Sec. 12-4 other aspects associated with programs as accessors.

12-2-3 Access Parameters

The *location* of the accessor, or more precisely the destination where the results are to be returned, can be another parameter of the access description. Confidential data are not to be shipped into areas that are not secure. The location also may define the type of device. Access to a check- or ticket-writing device may be made especially secure.

Sometimes it may not be desirable that a permanent paper record be created which could cause later confusion. An example might be an unverified clinical test result that is made available on a display to aid in quick diagnosis but should not become part of the permanent medical record. It may be more advisable to identify such preliminary information specifically rather than to devise separate access keys for hard and soft devices.

The *time and day* of access may be a necessary adjunct to the destination parameter in the accessor description. A location may be considered secure only during normal office hours. There also may be a time component to the value of data, so that data of a certain age may be more liberally accessible than very current information.

The *frequency* with which an element is accessed may be of concern. A sales total may be requested quarterly for the public stockholders report; the same value produced every hour could be correlated with specific sales activity which a company might wish to keep confidential. An unusually high access frequency would generally be indicative of an unusual situation and possibly reveal foul play.

12-2-4 Access Key Management

To lessen the possibility of theft within the system of accessor keys, a transformation which is not uniquely reversible can be applied to the keys. Within the system only the transformed keys are stored and used for access privilege verification, so that the original key is visible within the system only for a short time. The hashing techniques discussed in Sec. 3-5 provide this type of transformation.

The full number of access keys could be as large as:

$$\begin{aligned}\#(\text{accesskeys}) = {} & \#(\text{individuals}) \times \#(\text{access procedures}) \\ & \times \#(\text{distinct devices or locations}) \times \#(\text{time categories}) \\ & \times \#(\text{frequency categories})\end{aligned} \qquad 12\text{-}2$$

We will discuss in Sec. 12-6 methods which can be used to map this set of access keys into a more manageable set of accessor categories or *cliques*.

12-3 TYPES OF DATA ACCESS

Accesses to data can be categorized by type. Distinctions are commonly made between authorization to read and authorization to write data. Magnetic-tape reels, cartridges, and some disks have inserts which, when removed, make writing physically impossible. Some disk drives have write-protect switches. Many shared operating systems have added an execute-only privilege. Here one can allow an accessor to use a program without allowing reading or copying the program. Such a facility provides a protection for the products of programmers. The execute-only privilege may in turn protect the data used by a protected program, since its access mechanism and identification key is hidden.

Computer systems limited to these three types of authorization still provide fairly unsatisfactory protection. The procedures which are part of the operating system are in general authorized to read or write anything, anywhere, and could not function if this privilege were removed. The read privilege is frequently available to any user and only a knowledge of the structure of another user's data is required to gain access. Various systems have implemented additional protection privileges. We will not survey these in detail but rather will summarize implemented and suggested types of access categories below.

Seven Access Types We will present in the remainder of this section a set of seven distinct access privilege types which in combination allow a great degree of protection control.

`READ` access grants the privilege of copying data into the accessor's environment. Once the data are copied, the copy can be manipulated by the accessors as they please. This type of access privilege protects the stored data, but not the information they contain.

`EXECUTE` access grants the privilege of use of a program or a procedure to the accessor. With the use of the privilege may be associated privileges of extended data access as discussed earlier. The text of the procedure and data read by the procedure are not made available, nor can the program be modified without additional privileges. Using this privilege information can also be selectively protected.

`CHANGE` access provides the accessor with the conventional write access. This access privilege provides capability to destroy stored data. In view of the *extend* privilege discussed below, it may be appropriate to restrict this privilege to the updating or changing of existing data items.

The `DELETE` access privilege is closely related to the privilege of write access. This privilege allows destruction of the information that a data object existed, and of course also destroys the data in the object itself.

Change access allows only destruction. It is not clear whether the fine distinction between these two access types is worth the addition of this access type. A delete access causes a change of file size; for this reason it may be desirable to distinguish the delete and change privileges.

The `EXTEND` access allows the addition of data to a file without the capability to destroy other data in the file and without the privilege to read previous stored data unless those privileges also were conferred. A file thus referenced will grow.

Files where only extend access is appropriate are the files which contain system accounting data or audit trails. Most data-entry operations also fall into this category. Many conventional write privileges combine change and extend.

The concept of `MOVE` access provides another privilege which is not commonly available. This privilege provides the capability to move data fields without the privilege to read their contents. Many operations within computing and data-processing systems involve the movement of passive data elements associated with key data elements. In order to make decisions, the key has to be read, and read access to the key has to be granted. The need to read the key does not imply that all associated data should be available.

Specific examples where the move privilege is appropriate and adequate for the tasks can be found in many operating system functions. The movement of users' data to output buffers for blocking as well as the movement of data pages in virtual memory systems requires only move access. The transposition program in Sec. 4-2 provides an example of no requirements to *read* the data whatsoever. In most commercial operations such as banking, public utility, and medical data processing, records are continuously being sorted or moved based on identification keys. The fact that the programmer who writes the programs that move the data also obtains read and write privileges has aided both intentional fraud and unintentional alteration of other data fields. Not all such errors can be prevented by protection but the additional detection capability obtained will help in minimizing problems.

The move privilege can also be used to place data on high- or low-performance devices in order to optimize overall system performance. System tuning is frequently aided by outside specialists who should not need read access to data in order to carry out their business.

The control of access for `EXISTENCE VERIFICATION` completes the set of privilege types to be considered. Without this privilege an accessor cannot determine if an object, say a specific record or an attribute value, exists. It frequently is necessary to determine whether a specific data element exists in order to make decisions on the invocation of further processes. A program to which this privilege has not been granted cannot even determine the existence of the record.

There is a need for two distinct alternate responses when data are accessed to which read privileges have not been granted. A program constructing an index of records has a valid need to determine existence and location of a record, and other programs should obtain such data.

An example might be a request for a psychiatric health record. An intruder who attempts to read a psychiatric record of an individual might receive the message `"No such record"` if the person does not have a psychiatric record. If there exists such a record, the intruder who does not possess read privileges would be denied access to the information and might receive a message from the computer stating `"Improper access request"`. These innocuous responses actually provide valuable information. Lack of existence verification privilege would result in a message `"Thou shalt not attempt to look for such data"`. In many implementations, this privilege may be closely related to the privilege of access to the index to the data, which we will discuss in the next section.

Summary The privileges given can be conveniently coded by single bit indicators; say `"0"` disallowing and `"1"` granting access. The seven distinct access types recognized above provide $2^7 = 128$ combinations as shown in Table 12-2.

Table 12-2 Bit Assignment for a Protection Key Byte

`Bit 0`	Key byte itself is valid
`Bit 1`	Read access is granted
`Bit 2`	Execute access is granted
`Bit 3`	Change access is granted
`Bit 4`	Delete access is granted
`Bit 5`	Extend access is granted
`Bit 6`	Move access is granted
`Bit 7`	Existence Verification access is granted

It is probable that the no-access code (`"10000000"`) and the all-access code (`"11111111"`) combinations will occur most frequently. System programs will refer to users' data with `"10000011"`. Access keys themselves can be manipulated with this protection level. The validity bit provides safety during manipulation. An understanding of the combinations that these access types provide is necessary in order to be able to specify the access level of a data object for an accessor.

12-4 THE OBJECTS TO BE LOCKED

The data space, addressed by an accessor for a specific type of access, contains the objects to be provided or defended. Various types of such objects can be distinguished: *data objects*, *access paths*, *database programs*, and *schema entries*. Each of these objects can have multiple versions over time. We will discuss all of these in turn.

Data First of all, there are the recorded facts themselves. A fact becomes interesting to the accessor only when it is bound to a set of attributes which will identify it so that the fact can be related to a real-world person or object. These attributes may have been used as the access path to the element, and hence may be known. On the other hand, there is little probability that a random exposure of a single data field will do much harm. If it could, we should not allow visitors to our computer installation to take a photograph of the console of the shiny machine with its lights displaying some current value of some object.

The size of an object is of greater concern. It is very conceivable that each element of a record has a distinct usage, so that accessor and access type should be specified for one element alone. More practical is that identical access constraints are assigned to all attribute values of the same attribute.

Gathering of attribute types which have common access constraints, say `street address` and `city`, creates protection segments. In many systems segments defined for protection and record segments defined for manipulation have to be identical. Other systems limit protection boundaries to records, so that the entire file is the data object to be protected.

Access Paths In addition to the data elements which represent data values, there are elements in database which indicate relationships and which appear as pointers in the files. Other pointers exist in indexes to the data. These pointers present a separate class of objects to be locked. In many instances there is the need for distinct protection privileges for pointers to objects and for the value of objects.

Let us consider some instances.

1 A data entry has to be updated: the pointer is read, but the data are read and rewritten.

2 A survey wants to know the number of alcoholics on the staff but should not be able to determine who they are. The pointer is read and counted if valid; the data are not accessible.

3 A file-maintenance program reorganizes the file; new pointers are created and written, but the data are only to be moved and are supposedly of no interest to the writer or executor of this task.

In each of the three examples of access to data via a path using pointers, it seems desirable to protect the pointers to a different degree than the data themselves.

There is some conceptual overlap between pointers as protection objects and the availability of existence verification as an access type. Complete omission of either may, however, leave some gaps in a protection system.

Programs Programs are also objects which already have generated a need for a specific protection type. Since programs can be active elements in a database system, their protection is of equal concern. Where protection for data is reasonably complete, programs, if put under the same rules, also will be protected. The classification of data objects as records with attributes, however, does not apply. The existence of program text has to be specifically recognized. The other aspect of a program, that of an active accessor, was discussed in Sec. 12-3-1. Programs will have to be treated as distinct names units if use of a specific program can convey extended access privileges.

Schemas The database schema, if one exists, is another object of protection. Here a concern is one of adequate object naming in a database schema environment, where protection is provided through the schema itself. Some systems may keep the schema again in a schema-controlled file and apply protection through this upward recursion. A proper termination of this nesting of schemas will have to be provided. If the schema is relatively static, it is possible to include the schema file in a single schema description.

The allocation of schema access privileges is very critical. Many accessors will have read privileges to the descriptors, but very few will have change or write privileges. An error when modifying a schema can make the entire database inaccessible. If the schema file is itself controlled by the same schema, the use of these modify privileges is awkward to control.

Stronger schema access protection may include a requirement that more than one accessor assigns a change access privilege to a process which will modify the schema task. Both will receive reports of all activities. This joint authorization is equivalent to the double signature requirement in many companies for writing of large checks. An alternate method of control is to maintain two schemas that control each other and that which are assigned to different accessors at the top level so that again joint control exists.

Time The age of a data object may be of concern in the assignment of access privileges. Data can lose information value over time and be made more generally available outside the system. There are other circumstances where information should not be made available after a legal statutory time limit, in order to protect individuals from continuing embarrassment or damage.

Summary The number of data objects that are candidates for protection is the sum of the counts of the object categories (data, access paths, programs, schemas) multiplied by a number of time-division categories. Section 12-6 discusses how to deal with this large number.

12-5 ENVELOPE OF PROTECTION

Since database and file services are an integral part of computer-system facilities, we cannot limit protection consideration to the database system alone, and yet there will be a area beyond which the responsibility for protection is no longer in the hand of the database management. The definition of the *envelope* where protection is maintained is frequently neglected. The omission of a statement defining the envelope, coupled with specifications of the security designed into the internal operation of the system, can easily be misleading. Users of computing systems may be only too willing to believe that adequate protection exists.

Protection-System Components We frequently will have to include in our considerations the operating system, the input and output subsystems, the authentication subsystem, as well as the staff involved in the operation. In the ideal case, we can protect the database as a separate unit. If the operating system is included, there is likely to be a lower reliability because of the large part counts and greater number of accessors. The state of security provisions in existing computer systems is such that genuinely secret work is carried out on unshared computers, and in protected buildings, by checked and bonded personnel.

We assume throughout that the database is accessed only through the database or through a file system which recognizes the protection boundaries. In systems where no transaction-log is part of the system, backup files will be created by using *utility* programs which often ignore protection conventions. The backup tapes as well as output from unauthorized execution of utility programs provide convenient opportunities for the breach of protection boundaries.

Loss of Control over Data Another problem in the definition of the protection envelope is due to routine exchange of information.

An example of a privacy leak is the movement of medical record data to insurance carriers and beyond, where the data is used to determine eligibility for insurance coverage and to evaluate claims for health care services provided. An extensive study [Carrol72] has shown that this is one of the largest leaks of private information. This transfer proceeds directly from provider to insurance company and then to centralized reference files so that potential loss of privacy is hard to perceive by outsiders.

Another case which has been cited is caused by law-enforcement support systems which service more than one state while these states operate under different legislative rules concerning privacy. The multinational companies, frequently with centralized computing facilities, are difficult to control legally by courts of the countries where they do business and on whose citizens they may keep records.

Rigorous definition of files, control of input and output, and identification of personnel entering the protected envelope are the most important tools currently available to prevent unauthorized access. Logging of all attempts, successful or not, to gain access helps in ex post facto analysis of violations. Genuine concern by management is needed to make the staff aware of the importance of protection.

Response to Violations The action to be taken when a violation of the protection mechanism is detected has to be carefully considered.

Not everybody who attempts to access the wrong record has to be thrown instantly into jail. A large number of access violations are due to programmer debugging or inquiry clerk typing errors. A system that exhibits an excessively paranoid behavior is likely to frustrate legitimate use as well. On the other hand, there should be adequate logging of deviant access behavior to allow detection of illicit action and eventual entrapment of perpetrators.

It has been recommended that, with the current level of protection provided by data-processing systems, commercial firms employ one data-processing oriented auditor for every 10 to 20 programmers on the staff [Browne79]. The high cost of such an approach, as well as the historic failures of auditors to detect glaring instances of computer abuse, should provide an impetus to make improvements in protection systems.

12-6 ACCESS KEY ORGANIZATION

In order to implement the requirements stated in the preceding section, the protection requirements have to be categorized and organized. A three-dimensional access matrix which is the product of the number of accessors, the number of access types, and the number of data elements and links will be excessively large.

12-6-1 Cliques

The effect of a large number of users can be reduced by establishing user categories or cliques. For example, all data entry personnel may be identified as a member of the same clique. Thus there will be a table which maps the individual user identification into a clique identification. A further reduction of number of cliques required is possible if an individual can be a member of more than one clique, since fewer special categories may be needed. On the other hand, unexpected privileges may be the result of multiple, concurrent clique memberships. To avoid this problem, an accessor may have to state the database area to which access is desired, so that only one clique membership is active at a time.

A clique-identification number is assigned internally. Once a clique member is authenticated, this number is hard to forge and will provide a better protection than user-identification keys. It may be desirable to periodically reauthenticate the submitters of clique-identification numbers. A monitoring process can check if the user's identification is appropriate for a legitimate member of the clique or would suggest that a stolen clique number exists. Clique numbers which have been compromised can be withdrawn and new numbers assigned.

12-6-2 Data Objects

The organization of the protection objects depends greatly on the type of database control which is provided in the system. For an object to be protected, it has to be accessed through a *naming* mechanism. Objects on levels where actual addresses are used cannot be identified for protection.

If the system provides detailed descriptions of record contents and linkages (i.e., a schema), quite detailed protection is possible. If no facilities of this type are provided, protection generally is limited to the file level.

Files Virtual-memory systems (see Sec. 4-8) can provide a hardware definition of files through the use of named storage segments, so that protection is easier to enforce. In other cases files are defined by software and control information, obtained by the operating system when a file is opened. In either case, files are the smallest units which have symbolic names or symbolic numbers. The method used for file protection is always determined by the operating system. To identify files uniquely within a system, their names will be prefixed automatically by a qualification term, constructed from the user name.

Example 12-1 User Authentication and File Names

```
Login:                             NAME? Gio Wiederhold
                                   PROJECT? DataBase
                                   ...
Execution:                         ...
                                   OPEN FILE(mine) UPDATE;
will define a file named                G_Wieder.DataBase.mine
```

This method is augmented to enable users to access common system files and to enable sharing of files to the extent desired. To access files other than one's own is made possible by specifying the fully qualified names of files such as

`SYSTEM.Library.Regression` or `JCOther.DataBase.Observations`

The operating system can check whether `Joe Carlos Other` has given anybody, or specifically someone with the system name `G_Wieder`, the privilege to access his data. A list of authorized users can be kept with the file directory and will indicate the type of privilege accorded. Once `READ` access privilege has been given, the original user has lost control over the data, since the reader can again permit another reader to access the files.

A project name, as `DataBase` used above, can be used to imply sharing of data files for all members of a project. A project becomes an external manifestation of a clique organization. A higher level of authorization can be required for access by users outside of the project.

12-6-3 System Support for Protection

Protection is largely a system function, and knowledge of the available capabilities, or the lack thereof, is necessary to equip database systems appropriately. File systems can rarely provide much protection beyond the facilties of the operating system and the underlying hardware.

Hardware Mechanisms The services provided by computer systems can be considerably enhanced by hardware mechanisms which can limit access at the point of data-element fetch or store. Hardware mechanisms are based on matching a protected access key provided by a user with a key field associated with data storage. Two types of methods are in use.

Rings A hierarchical assignment of access privileges has been used in some systems. Figure 12-2 illustrates the concept of protection rings where the degree and type of access available is dependent on a level number assigned to the accessor. Such a system is modeled on the basic military secrecy hierarchy of

Unclassified
Confidential
Secret
Top secret

In practice, only few levels are usable, since a single privacy hierarchy of data has to be established appropriate to all accessors. The MULTICS system [Graham[68]] associates ring privileges specifically with processes and storage segments. Systems with rings are effective in protecting operating systems and subsystems, and can contribute much to reliability. For databases the hierarchical constraints of rings have to be combined with control over files, record segments, and access paths.

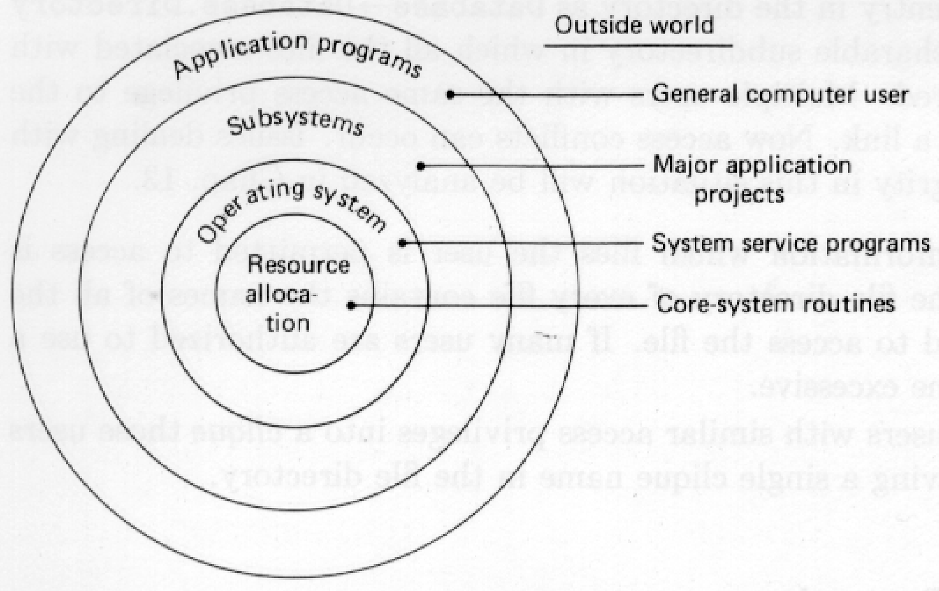


Figure 12-2 Protection rings.

Capabilities An allocation not based on a predetermined hierarchy of access privileges has been provided in so-called *capability-based computer systems*. Here the access control is provided through access keys which are shown and verified when data objects are accessed. Capabilities for access are initially given to a user when new objects are created. To avoid that access keys are copies or assembled by intruders, they are kept in a separate mechanism. In order to share data, the owner of the data can give an access key with selected capabilities to another user. To implement capability management in hardware, a machine architecture with virtual storage segments will be chosen.

Computer systems may combine access lists, rings, and capabilities to provide the basic protection mechanism.

Software Mechanisms No hardware system provides protection adequate for all circumstances, so that operating systems carry much of the responsibility for file protection. We find the two approaches used for hardware mirrored in operating systems.

Common to all software mechanisms is the use of catalogs which relate accessors to file names. Granting any type of access to a combination of user and file will in operating systems provide access to all data in the file.

Much implicit protection used to exist when access to computers was difficult. Now a large fraction of the population knows how to use computers and many computers are available on nationwide networks. This aspect of protection is no longer viable and should be discounted.

User-Based The information which files the user is permitted to access may be stored with the user's directory. The names of files which may be accessed by more than one user will appear in multiple directories; the entries will contain the access codes authorized by the user. The system has to protect these codes, so that the users cannot change entries in their own directory.

Excessive duplication of file entries may be avoided by having a hierarchical file directory structure. An entry in the directory as `Database` → `Database.Directory` can provide a link to a sharable subdirectory in which all the files associated with some database are entered. Multiple users with the same access privilege to the database may have such a link. Now access conflicts can occur. Issues dealing with the maintenance of integrity in this situation will be analyzed in Chap. 13.

File-Based The information which files the user is permitted to access is stored with the files. The file directory of every file contains the names of all the users who are authorized to access the file. If many users are authorized to use a file, the list might become excessive.

By now organizing users with similar access privileges into a *clique* those users can be authorized by having a single clique name in the file directory.

12-6-4 Schemas and Protection

The system-methods presented above are restricted to files as their unit of protection. A database system can distinguish much finer data units.

In a schema-oriented environment, privacy constraints for data elements can be kept with the data element descriptions in the database schema. We mentioned this in Sec. 8-1-3. Since we can expect many element types to have similar requirements, a reference to a data-privilege table can be used in the data-element description, so that duplication of access lists is avoided. Figure 12-3 illustrates this organization. This approach, while it keeps the access matrix relatively small, gives access to all or to no data values associated with a given attribute name.

By controlling access to index elements or selected rings of records, this organization can be extended to provide access to selected data elements of a certain name according to hierarchical requirements. Figure 12-4 expands on this notion in relation to one of the examples in the introduction to this chapter.

Fred Fierce can protect working data of his division by protecting an object used to define the access path to these data. Another manager can protect his portion of the database similarly.

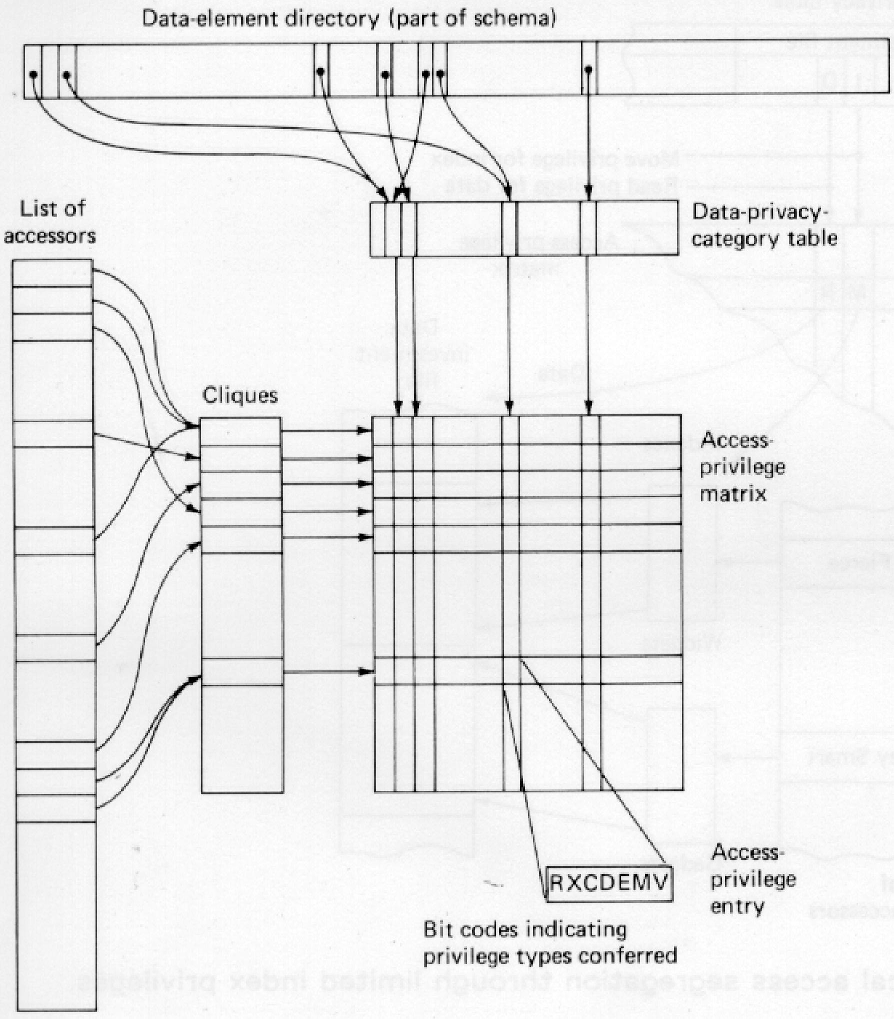

Data-element directory (part of schema)
List of accessors
Data-privacy-category table
Cliques
Access-privilege matrix
RXCDEMV
Access-privilege entry
Bit codes indicating privilege types conferred

Figure 12-3 Schema-oriented access matrix.

The access path is probably defined via an index. Protection of *horizontal fragments* of files via access paths is associated with the use of hierarchically structured files. Network and relational structures are more difficult along the access path. In a relational system, INGRES, *constraint statements* (see Sec. 9-2-5) can be given for that purpose. These disallow retrieval based on values found in the tuples. The system will access these data before being able to apply the constraint.

A more general means of providing checks on accesses to subgroup information is provided through protective database procedures. An access verification program can look at system variables such as `user`, `date`, and `time`, and the database variables being accessed to determine if access is permissible. These routines are specified in the the schema and will be automatically invoked.

The use of a program allows checking of the originator of a request and the path by which the request arrives, and can also write an audit record to register the accesses. This approach is, however, costly in execution time and open to subversion. The verification program must have access privileges to the user area being verified, to the area where the access criteria are stored, and to the log file. Such extensive privileges are inherently dangerous.

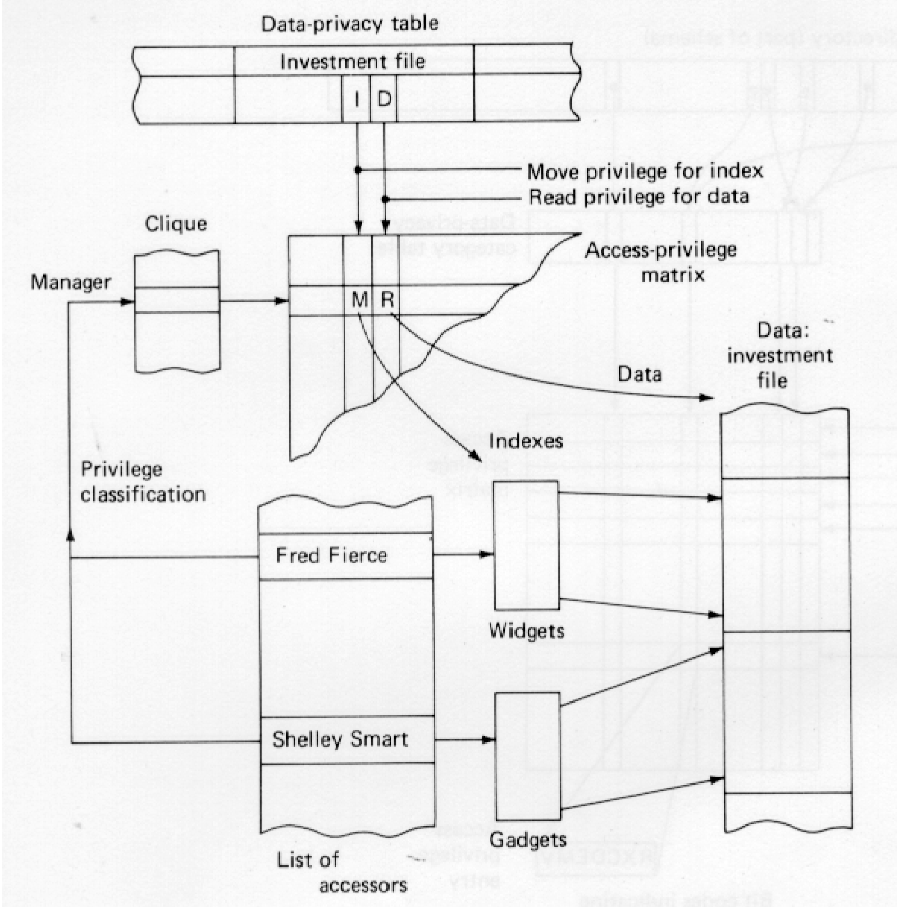


Figure 12-4 Vertical access segregation through limited index privileges.

The CODASYL specifications include most aspects of privacy protection. Access to data items, a record, an area, as well as the schema, can be protected by using statements as shown in Fig. 12-5. Here the accessor is assumed to be submit either a fixed password, which is specified in the schema itself (`literal`), or a password kept as a variable (`lock_name`), or is checked by means of a procedure (`data_base_procedure`). Such a procedure is invoked when the element is accessed and will not only have the ability to request the accessor's password but can also check on the terminal location, the access history, and other relevant material.

The access type can be made dependent on the operation being performed, i.e., `GET`, `MODIFY`, or `STORE`, for access to individual data elements, or may apply to all three operations on the elements. A `PRIVACY LOCK` on `LOCKS` refers to all locking operations in an area and would typically be very securely assigned to the database administrator.

Often the smallest data element that can be protected is the record. Elements that belong logically together but need separate `PRIVACY` constraints have to be kept in separate, parallel records segments.

Database procedures are also provided in general, and can be specified for execution under similar access conditions. In that case they can be use to maintain *integrity constraints*, for instance, disallow deletion of a referenced record.

12-6-5 Summary

We have reviewed in this section in a general sense the protection mechanisms which are available in operating systems. Current developments in operating systems are increasingly cognizant of the existence and needs of databases. We notice in the examples shown the interrelationship between protection methods and file system design. Such a feedback is difficult to avoid, and makes the design of secure systems an iterative undertaking.

It remains important for a database designer to review the available facilities and understand their weaknesses. Saltzer[75] has collected 10 principles to be considered in the analysis of protection systems.

Economy of Mechanism An excessively complex protection system does not allow verification of correctness and completeness.

Fail-Safe Defaults If something goes wrong, access should be denied. Even conscientious users will not report unexpected access privileges.

Complete Mediation Every access path to the database has to go via the protection mechanism.

Open Design Protection is not enhanced by keeping the mechanism used for protection secret. A secret mechanism can also install an unwarranted level of faith in the users.

Separation of Privilege Critical data should be protected, so that even a single authorized user cannot subvert the mechanism.

Least Privilege A program should receive only essential access privileges.

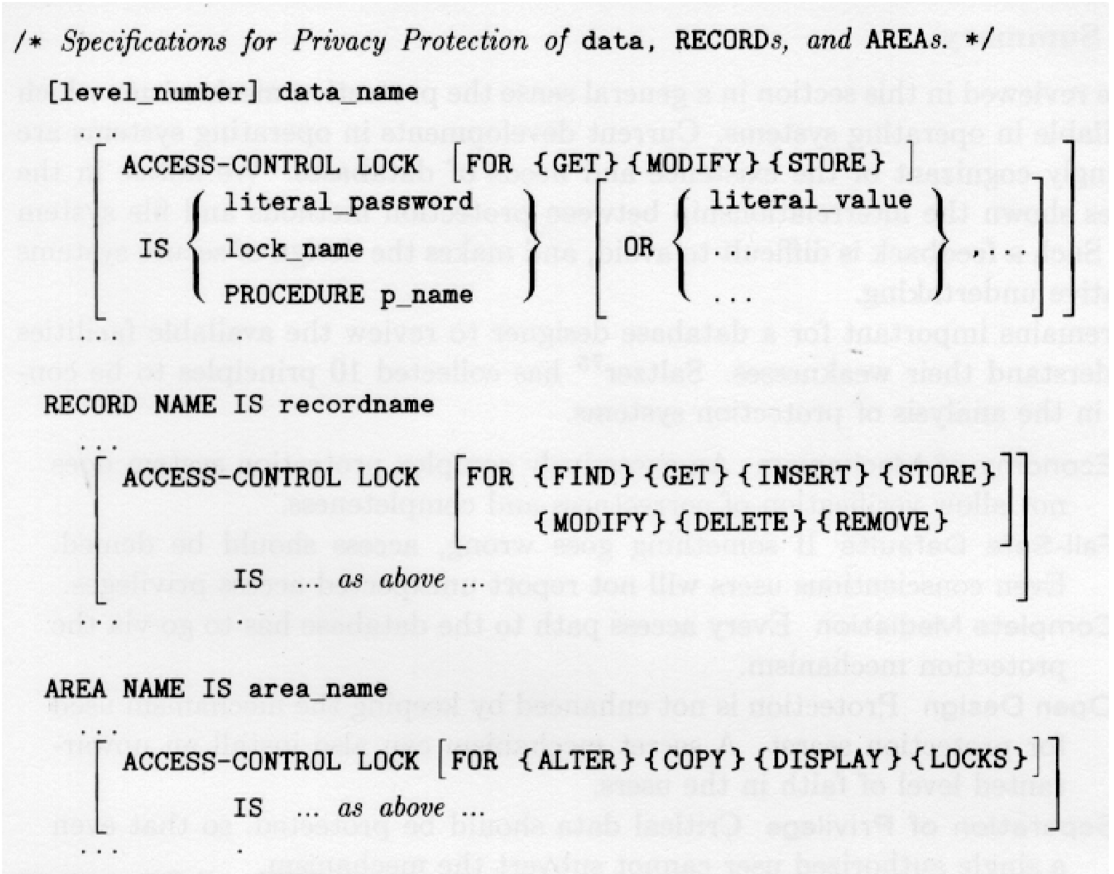
Least Common Mechanism Avoid all unnecessary sharing. We note that in an operating system sharing is perceived as a burden often incurred for economy of storage or computing facilities. In databases sharing is the major benefit, so that this principle is best restated as: avoid keeping data that cannot be shared in a database.

Psychological Acceptability The mechanisms must match the user's data model and the protection required for it. Failure to do so creates usage errors and an incentive to bypass the system.

Work Factor The effort to subvert the mechanism must be greater than the benefit which could be gained. The intruder will probably also have access to a computer so that he can exert much effort at little cost.

Compromise Recording Logging of accesses can create the audit trail necessary to correct problems and trap the perpetrators. Intruders into the innards of a system may, of course, be able to avoid creating a trail or can erase the evidence.

Where systems are found inadequate, the database designer may augment the facilities but should also provide to the users a realistic assessment of the available level of protection. If storage of critical data can be avoided, privacy protection is simplified. If the atmosphere of a database operation is open, vandalism will be less of a problem. Since the objective of a database is to enable the sharing of information, the removal of barriers to sharing is important.



```
/* Specifications for Privacy Protection of data, RECORDs, and AREAs. */
[level_number] data_name

 ⎡ ACCESS-CONTROL LOCK [FOR {GET}{MODIFY}{STORE} ]                        ⎤
 ⎢     ⎧ literal_password ⎫  ⎡    ⎧ literal_value ⎫   ⎤                  ⎥
 ⎢  IS ⎨ lock_name        ⎬  ⎢ OR ⎨ ...           ⎬   ⎥                  ⎥
 ⎣     ⎩ PROCEDURE p_name ⎭  ⎣    ⎩ ...           ⎭   ⎦                  ⎦

RECORD NAME IS recordname
 ...
 ⎡ ACCESS-CONTROL LOCK ⎡FOR {FIND}{GET}{INSERT}{STORE}⎤ ⎤
 ⎢                     ⎣    {MODIFY}{DELETE}{REMOVE}  ⎦ ⎥
 ⎣             IS  ... as above ...                      ⎦
 ...

AREA NAME IS area_name
 ...
 ⎡ ACCESS-CONTROL LOCK [FOR {ALTER}{COPY}{DISPLAY}{LOCKS}] ⎤
 ⎣             IS  ... as above ...                        ⎦
 ...
```

Figure 12-5 DBTG privacy clauses.

12-7 CRYPTOGRAPHY

An alternate protection technique is provided by transformation of data into a form that does not provide information when intercepted. Such methods can provide privacy in otherwise insecure environments and can provide effective secrecy in conjunction with the methods which were discussed previously.

Three basic techniques are available:

1 Encoding of information

2 Transposition of codes representing data

3 Substitution of codes representing data

The encoding of data is a common process even without privacy consideration. Letting the value "1" mean `male` and the value "2" stand for `female` is a form of encoding, which will protect that field to someone who does not have access to a code book or schema. We will discuss methods of encoding in detail in Chap. 14.

Operations where basic symbols are transposed or substituted in order to garble the data are referred to as *enciphering*. Such a transformation creates *cipher text* from *plain text*.

In order to understand the message, the recipient will have to *decipher* the cipher text, according to rules and a key used in the enciphering operation. The intruder, who wishes to understand the text, will have to resort to *decrypting* techniques. Figure 12-6 shows the components of cryptography.

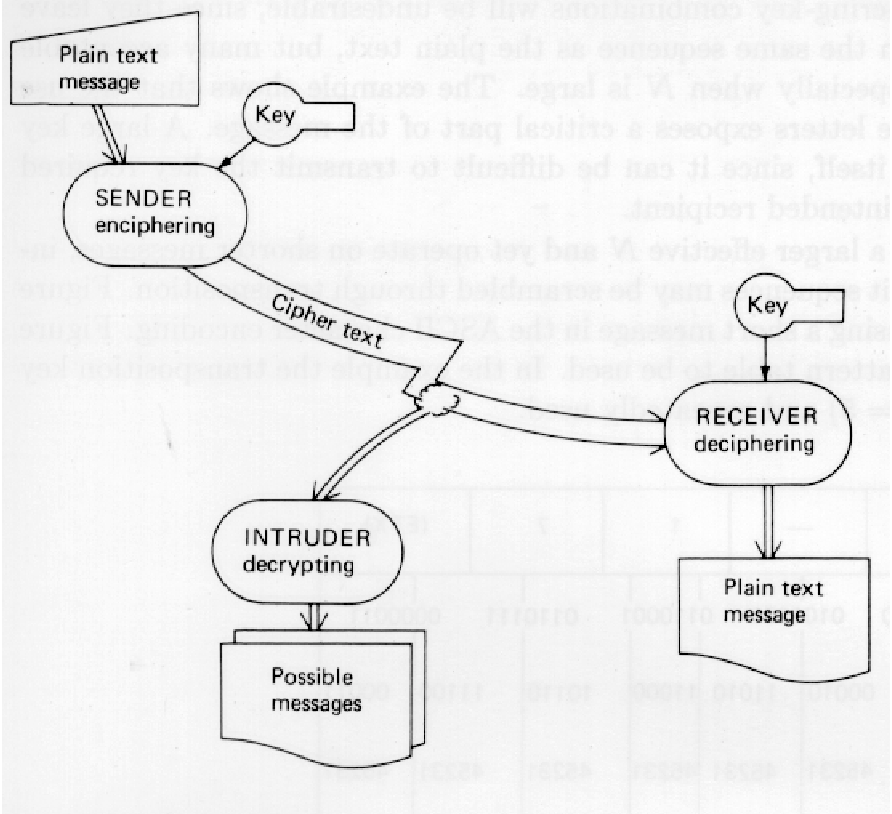


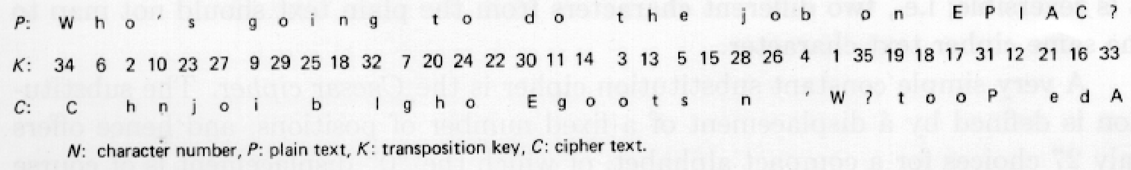
Figure 12-6 Elements of cryptography.

We will consider mainly operations on data consisting of alphabetic messages. It should be noted that in fields outside of cryptography, the term "code" is commonly used for processes regarded as a "cipher" within the field. Only in this section will we be careful and make the appropriate distinction.

The effectiveness of cryptography depends on the plain text to be enciphered. Natural-language text has very distinctive patterns, and also provides much redundancy which promotes both comprehensibility of garbled text in day-to-day use, and the decrypting of enciphered data by analytical methods. We first will present the two basic enciphering methods and then discuss the security which they provide.

12-7-1 Ciphering

A *transposition cipher* operates on a group of characters. A key indicates the rearrangement pattern, as shown in Fig. 12-7. If the block length is N there are $N!$ possible rearrangement patterns.



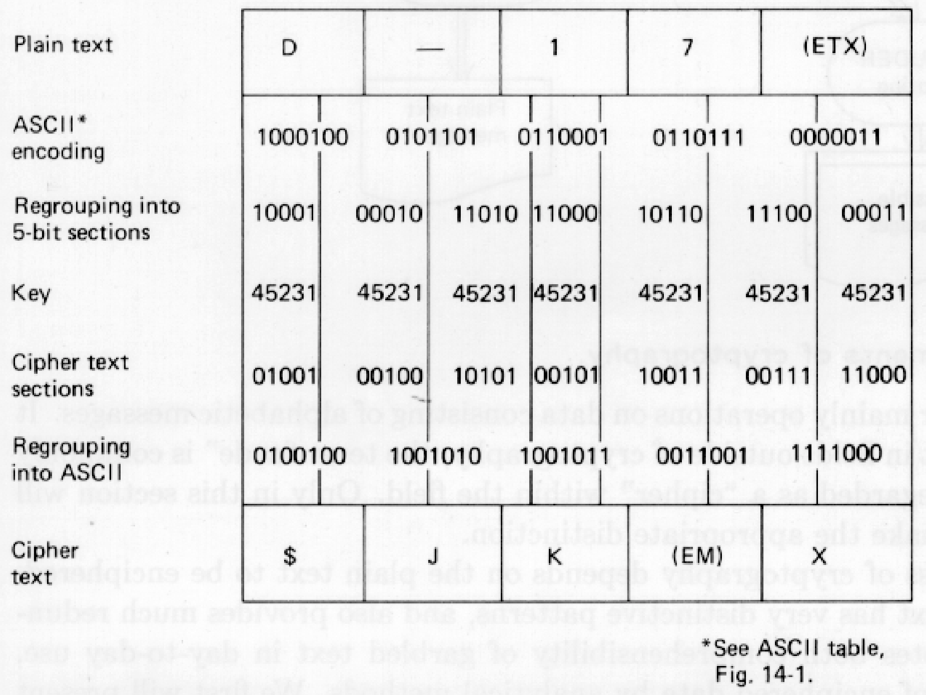
N:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35
P:	W	h	o	'	s		g	o	i	n	g		t	o		d	o		t	h	e		j	o	b		o	n		E	P	I	A	C	?
K:	34	6	2	10	23	27	9	29	25	18	32	7	20	24	22	30	11	14	3	13	5	15	28	26	4	1	35	19	18	17	31	12	21	16	33
C:	C		h	n	j	o	i		b		I	g	h	o		E	g	o	o	t	s		n		'	W	?	t	o	o	P		e	d	A

N: character number, *P*: plain text, *K*: transposition key, *C*: cipher text.

Figure 12-7 Transposition cipher.

Some of the $N!$ enciphering-key combinations will be undesirable, since they leave parts of the message in the same sequence as the plain text, but many acceptable alternatives remain, especially when N is large. The example shows that the use of infrequent uppercase letters exposes a critical part of the message. A large key presents a problem in itself, since it can be difficult to transmit the key required for deciphering to the intended recipient.

In order to obtain a larger effective N and yet operate on shorter messages, individual bits or small bit sequences may be scrambled through transposition. Figure 12-8 gives an example using a short message in the ASCII character encoding. Figure 14-1 provides the bit-pattern table to be used. In the example the transposition key used is very short ($N = 5$) and repeatedly used.



Plain text	D	—	1	7	(ETX)		
ASCII* encoding	1000100	0101101	0110001	0110111	0000011		
Regrouping into 5-bit sections	10001	00010	11010	11000	10110	11100	00011
Key	45231	45231	45231	45231	45231	45231	45231
Cipher text sections	01001	00100	10101	00101	10011	00111	11000
Regrouping into ASCII	0100100	1001010	1001011	0011001	1111000		
Cipher text	$	J	K	(EM)	X		

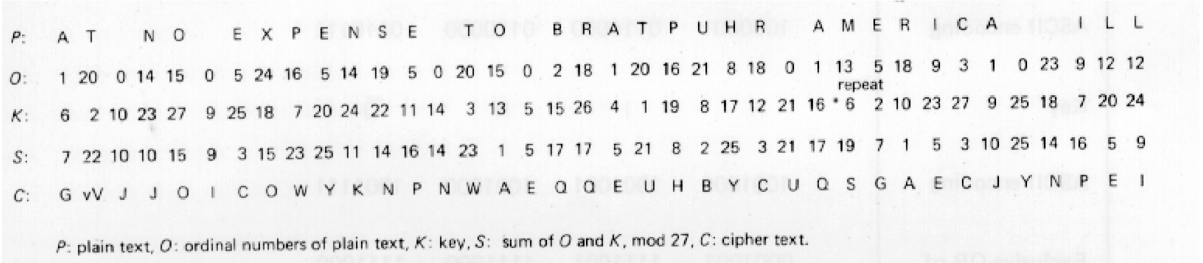
*See ASCII table, Fig. 14-1.

Figure 12-8 Transposition ciphering using a 5-bit transposition key.

A *substitution cipher* provides replacement of characters of the source message by one or more characters of a cipher alphabet. A *constant substitution cipher* provides an alternate unique character for every character of the source alphabet. The number of alternatives here is again N. The cipher can be described by a substitution table, or alternatively by the number of positions in the alphabet between the source and cipher character. The substitution must be chosen so that it is reversible; i.e., two different characters from the plain text should not map to the same cipher text character.

A very simple constant substitution cipher is the *Caesar cipher*. The substitution is defined by a displacement of a fixed number of positions, and hence offers only 27 choices for a compact alphabet, of which the "0" displacement is of course unsuitable. The choice of "1" has achieved fame in Arthur Clarke's movie script for "2001" by its use in naming the HAL computer.

A Gronsfeld substitution cipher is not constant. The key specifies the number of steps taken through the alphabet to obtain the cipher character. The number of steps can vary from zero to the number of characters which comprise the plain-text alphabet. The key can be produced by random permutation generator. Figure 12-9 provides an example of such a Gronsfeld cipher for an alphabet of 27 characters. The number of alternative encodings is now N^2 for an N-character alphabet, with very few choices that are unadvisable. A key containing many zeros would not provide satisfactory enciphering.



P:	A	T		N	O		E	X	P	E	N	S	E		T	O		B	R
O:	1	20	0	14	15	0	5	24	16	5	14	19	5	0	20	15	0	2	18
K:	6	2	10	23	27	9	25	18	7	20	24	22	11	14	3	13	5	15	26
S:	7	22	10	10	15	9	3	15	23	25	11	14	16	14	23	1	5	17	17
C:	G	vV	J	J	O	I	C	O	W	Y	K	N	P	N	W	A	E	Q	Q

P:	A	T	P	U	H	R		A	M	E	R	I	C	A		I	L	L	
O:	1	20	16	21	8	18	0	1	13	5	18	9	3	1	0	23	9	12	12
K:	4	1	19	8	17	12	21	16	repeat * 6	2	10	23	27	9	25	18	7	20	24
S:	5	21	8	2	25	3	21	17	19	7	1	5	3	10	25	14	16	5	9
C:	E	U	H	B	Y	C	U	Q	S	G	A	E	C	J	Y	N	P	E	I

P: plain text, *O*: ordinal numbers of plain text, *K*: key, *S*: sum of *O* and *K*, mod 27, *C*: cipher text.

Figure 12-9 Gronsfeld cipher.

Instead of using random key numbers, another message can be used. The ordinal values of the characters of the key message provide the amount of displacement in the alphabet value. This technique is called the *Vignere cipher*. The key text, of course, can be of any length, even as long as the message to be enciphered. Blanks in the key may be ignored. For instance, the key string `"AMERICA OF THEE I SING"` will generate a sequence $K = 01\,13\,05\,18\,09\,01\,15\,06\,20\,08\,05\,05\,09\,17\,09\,14\,07$. A Vignere procedure for a larger alphabet is shown in Example 12-2. Ordinal numbers representing the characters are added modulo the size of the alphabet, to determine the ordinal number of the cipher character.

Example 12-2 Vignere Ciphering Procedure

```
/* Input is the linein, key, and option = true for enciphering,
   false for deciphering. The result is in lineout */
cipher: PROCEDURE(linein,key,option,lineout) EXTERNAL;
    DECLARE (linein,lineout,key) CHAR;
    l_line = LENGTH(linein); l_key = LENGTH(key); lineout = '';
                                /* the typewriter alphabet */
    DECLARE abc CHAR(89) INITIAL('0123456789AaBbCcDdEeFf...
  /* Use one character to define the beginning of the key */
    k = INDEX(SUBSTR(key,l_key,1),abc); l_key = l_key-1;
    DO i = 1 TO l_line;
        ord = INDEX (SUBSTR(linein,i,1),abc);
        kod = INDEX (SUBSTR(key,k,1),abc);
        IF option THEN kod = -kod;
        cord = MOD(ord+kod,89)+1;
        SUBSTR(lineout,i,1) = SUBSTR(abc,cord,1)
        k = MOD(k,l_key)+1;
    END;
    RETURN; END cipher;
```

An alternative to the addition of ordinal numbers, modulo the size of the alphabet, to determine the ordinal number of the cipher character can be used when the characters use a fixed-bit space. The `exclusive-OR` operation, used in Sec. 3-5 for hashing, provides an operation which is symmetric for both enciphering and deciphering, as shown in Fig. 12-10.

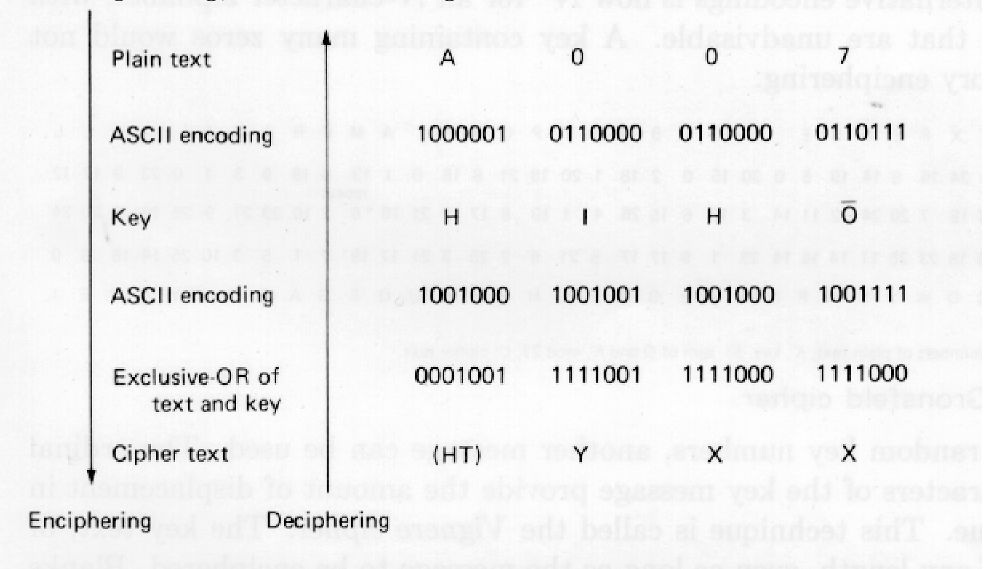

Plain text A 0 0 7
ASCII encoding 1000001 0110000 0110000 0110111
Key H I H Ō
ASCII encoding 1001000 1001001 1001000 1001111
Exclusive-OR of text and key 0001001 1111001 1111000 1111000
Cipher text (HT) Y X X
Enciphering
Deciphering

Figure 12-10 Ciphering with the exclusive-OR operation.

More sophisticated methods can be developed based on the two transformations discussed, but we first will describe common procedures of decrypting so that we can evaluate the weaknesses of the basic enciphering techniques.

12-7-2 Decryption

The ease with which a message can be decrypted depends on a number of factors:

- Is the method known?
- Is the source language known?
- How many alternative encipherings did exist?
- How much enciphered text using the same key is available?
- Is any matching plain text available?

A basic tool to aid in the decrypting of a cipher text is the generation of a *tableau*. If the method is known and the number of keys is limited, a tableau can be simply a listing of all possible decipherments. The tableau is scanned by eye and the plain text message will probably stand out.

If many, but yet a reasonably finite number of decipherments are possible, a *heuristic selection* of deciphered text can reduce the volume of the tableau to be listed. Simple heuristic algorithms might be that a vowel should exist in nearly all words, and that capitals and punctuation appear with certain regularity. Tableaus which do not show unreasonable pattern are not presented.

If the source language is known, substitution ciphers can be attacked using known *statistics* of letter frequencies. A simple substitution will generally yield the source text directly if the most frequent letter is replaced by E, the second most frequent letter by T, and so on, according to Table 12-3. Similar tables have been prepared for digrams (combinations of two letters), trigrams, initial and ending letters, and many other features of a language.

Even if such tables do not exist, as is the case when we have enciphered programs in a computer language, it may be possible to generate frequency tables from available source language samples. Programming language text also will be relatively easy to decrypt for a knowledgeable cryptologist, since the syntax is limited and the original variable names do not have to be reconstructed. Sequences of data digits will not provide much frequency information for decryption.

Table 12-3 Relative Frequency of Letters in English Written Text

E	133	D	43	G	14
T	93	L	38	B	13
O	85	C	31	V	10
A	81	F	29	K	5
N	75	U	28	X	3
I	71	M	27	J	2
R	70	P	22	Q	2
S	65	Y	15	Z	1
H	61	W	15	Total	1032

In the examples shown for the transposition ciphers and the Gronsfeld ciphers, *random-digit sequences* have been used that were obtained from a random number generator. Computer-software random-number generators, as well as hardware generators built from shift registers, have perfectly predictable sequences based on a few parameters. The parameters are

- The initial condition
- The multiplicand and shift amount
- The length of the register

The assumption that a random-number generator provides protection equivalent to its cycle length, which for a 21-bit generator can be up to 2^{21}, is not true for someone who has insight into its mechanism. Since, in practice, these random-number generators use parameters which have been carefully selected to give a maximal-length cycle without repeats, the number of alternatives is considerably less. For the 21-bit generator, the number of full cycle sequences is only 84 672 out of the 2^{21}, or more than 2 000 000 sequences.

Long keys provide more protection than shorter keys but may not be effectively used in retrieval from databases. For every block or record stored on a file the key is reset to its begin point to enable key synchronization on random retrieval. The key is then always limited to the length of the data unit. Only the first terms of random-number sequences are used in this case, but this allows more choices of random-number-generation parameters.

When the *key changes* a new decryption starts. frequent keys changes implies more transmission of keys between the enciphering point and the deciphering station, and this increases the susceptibility to having the keys intercepted or stolen.

Determination of the *key length* is important to the cryptographer who needs to reduce the search space. If enough cipher text is available relative to the key length, statistical tests on various groupings of the cipher text can provide clues. It has been stated that a ratio of twenty is adequate. The key length is estimated by generating descriptive parameters of various groupings of the cipher text. The greater cohesiveness of statistical results on the groups which match the original key length versus the groups which are unrelated provides key-length information to a computer-equipped cryptographer. Large amounts of cipher text in a database help to reveal the key patterns.

A considerable effort is required for decryption of nontrivial enciphered text. There is hence a great inducement to obtain some helpful extra information.

If the cryptoanalyst obtains a *matching segment* of cipher and plain text, the task is considerably simpler. If the segment is longer than the key, the key used can be determined in most cases where the enciphering method is known. A method to obtain matching text can be the use of index and goal records of sorted data, which allow the conjecture of data values by linkage or position.

If the intruder (gallantry presumes him male) has the capability to use the enciphering process, then he can submit data and key himself to obtain matching plain and cipher text and guess the method. If he can masquerade as a legitimate user of entry facilities, he may be able to obtain matching plain- and cipher-text based on the legitimate key, and guess both method and key.

12-7-3 Countermeasures

Having looked at the tools of the well-equipped cryptoanalyst we can survey countermeasures against attacks to these vulnerable spots. The countermeasures are listed in the same sequence that the decrypting devices were presented.

To minimize the effectiveness of the use of *tableaus*, we ensure that the number of encipherings is very high. The parameters for the basic methods have been given; combinations of the basic methods may produce the product or the sum of the individual alternatives. For instance, two successive Vignere ciphers — a so-called *two-loop system* — provides only the sum of the two ciphering options, although the key length will be increased if the loops are not of equal length. We always assume that the method is known to the intruder; after all, he may be using the same computer.

Reducing the redundancy of the source text will aid in disguising the message. In general, blanks and other punctuation will be omitted, and only uppercase characters may be used, transforming an 89-character typewriter alphabet to a 36- or 26-character set. By restricting enciphering to material which truly requires protection, the volume of available cipher text will be less and the analysis will be more difficult.

A number of approaches can be used to hide statistical features of cryptographic text. The use of *homophonic enciphering* can be very effective and relatively easy in a computer-based database. This method makes use of the fact that frequently there are more bit positions available in a computer code than are actually used. If eight bit positions are available so that there are 256 possible characters for our cipher text, and our source alphabet consists of only 26 characters, we can map the frequent characters into multiple cipher characters. Given the relative frequencies of Table 12-3, we can assign to the letter "`E`" $133 \cdot 256/1032 \approx 33$ different substitution characters in any kind of order and hide the frequency of this particular character. The character "`Z`", on the other hand, would require $\lceil 1 \cdot 256/1032 \rceil$ or one character position. Some adjusting is needed to assign an integer number of code positions to each character within the total of 256. The letter "`U`" would receive seven positions. We might assign then to "`U`" the code values: (17, 30, 143, 145, 173, 225, 249). A random choice of these seven values would be transmitted when enciphering. The receiver would substitute "`U`" for any of these values. The other characters translate similarly into nonintersecting subsets of the code.

The decrypter is robbed of all single-character frequency information. A more complex homophonic cipher attempts to avoid frequency information not only for all the single characters in the source alphabet but also for other linguistic features which might be useful to a cryptoanalysis. Digrams and trigrams are effectively hidden by providing a transposition in addition to the other techniques. Data which have been encoded prior to enciphering also provide artificial and misleading frequency information.

Random-number sequences used for key generation can become much harder to detect if only a portion of the random number is used in the actual enciphering method. The generators can be restarted from a key value associated with the user to obtain many alternate sequences.

In a database environment, keys can be well protected if the generator of the enciphered record is the same person as the receiver, since then the key itself resides in the system for only a short time. The remainder of the time the key is stored in the user's mind, purse, or wallet. A change of key to foil a suspected intrusion is, however, very costly, since all the data stored under a previous key has to be deciphered and reciphered. Such a process in itself opens opportunities to the intruder.

In order to make the determination of the key from data text more difficult, more than one key might be used. Selection from a number of keys, possibly based on a record number, can weaken statistical analysis operating on large amounts of cipher text.

Varying the key length can cause great difficulty to algorithms which an intruder may use in his initial attempts to crack cipher text. The expansion of short messages with data designed to provide noise in the decrypting process also can disturb cryptoanalysis. The additional characters should follow neither random nor natural patterns but should have purposely obfuscating frequencies.

To avoid release of matching plain text and cipher text, the need for enciphering should always be carefully considered. For instance, if in address records the city, but not the zipcode, is enciphered, a large volume of matching text can become available. The problems of system access by intruders in order to obtain matching text are best handled by the existing identification and access logging methods. Since decryption takes time, there is a good chance to catch an intruder when an unusual access pattern has been detected during a periodic review of the access log.

12-7-4 Summary

The previous two subsections have illustrated a cops and robbers game which can be continued ad infinitum. At some point, the effort required to decrypt data will be greater than the cost of other subversion tactics. Literature on cryptology rarely discusses alternative methods of gaining access, so that an overall security trade-off will have to be made by the system designer. Cryptography has been widely applied to data transmission, so that some factors which distinguish data storage from communications should be listed.

In data transmission, there is a continuous stream of characters over a line. The effective message length can be very long and can use a continuous enciphering sequence, subject only to resynchronization in order to cope with transmission errors. In data storage, we want to access records in random order, and frequently encipher only small, critical items of the data record. This places a limit on the key length. The intruder may be willing to gather many such fields from a file to obtain decrypting information equivalent to a long continuous tap of a phone line.

Cryptographic methods also provide tools for joint access control to critical data. Either two successive keys can be applied by two individuals or parts of a key can be allotted, which have to be joined to become useful. Individual and project security can also be accomplished by use of multiple encipherment.

Where data are not generated or analyzed in the processor, but simply entered, stored, and retrieved, the ciphering mechanism can be in the remote data terminal. Such devices have been developed and are becoming increasingly feasible because of the increased sophistication of terminals. With such an approach, ciphering methods have to be used that are both transmission- and storage-oriented.

In database stored on computer systems cryptographic techniques provide only one aspect of the required access protection. Protection from destruction, intentional or accidental, is not obtained by enciphering; in fact, data will be lost if either the cipher text is destroyed or the key is lost.

Enciphering provides the capability to destroy selected data from backup files by destroying the key. This is often the only tool users have to assure that their data do not become available to someone who happens to reuse a tape which was used for system backup. The implementation of a combination of protection methods, of which cryptography is an important component, can yield a high level of security.

12-8 ANONYMITY AND POLLUTION

This section will concern itself with approaches to provide privacy in data banks which are used primarily for research, rather than for individual fact retrieval. When it is important to the suppliers of sensitive data that they cannot be traced, a researcher collecting such data may wish to provide such assurance, but then has to protect the individual's data from dissemination by coworkers, data-processing personnel, and possibly from inspection by government investigators with powers to subpoena records.

Many data banks exist to which these considerations apply. The reason for creation of large databases is the power they provide to spot trends and relationships. For individual data paper records are often more convenient.

12-8-1 Anonymity

Separation of the individual's identification from the data can provide the required anonymity but may impair aspects of the research. If, for instance, responses are gathered without identification data, the lack of a respondent's profile can make interpretation of the responses difficult.

An example of the desirability to have respondent information occurs when only educated individuals return a mailed query about the value of a certain TV program, because of some erudite language used in the questionnaire.

A separate data response and identification response, which cannot be linked, allows determination of who answered, and allows followup to rally participants who did not respond. However, no recourse exists to correct incomplete or inconsistent responses once the identification has been separated.

A reference number can be used with the data which link data responses to an individual, allowing determination of individuals through the responsible researcher only for followup. To protect the respondents, at least the reference numbers on the identification responses are destroyed when the collection is completed. If a long-term study requires multiple occasions of data entry by respondents, the identification file remains a liability.

If the participant in a study has the option to reinitiate followup, the participant can keep the reference number. The participant will be warned that if the number is lost no further followup is possible. To avoid the problem of loss a randomization procedure can create a reference number from an arbitrary identification sumbitted by the person. Hashing does not allow tracing of the individual from the reference number kept with the data. The identification string invented by the participant must not be traceable and must be destroyed after the reference number is obtained. Some consistency of the individual is required here, and there is always a finite chance of conflicting reference numbers. Data for conflict resolution may not be available (see Sec. 3-5).

12-8-2 Protected Linkage

The methods summarized above provide techniques which are useful before critical information enters a data file. These tools have been extensively used because of the deserved distrust of computer security mechanisms.

In many studies the need exists for the researchers to contact participants, so that an identification file has to be maintained. At other times data required for processing can contain enough information to identify individuals, or identification data remains necessary to allow the merging of data from other sources than the respondent.

A solution to this problem is the removal of the identification file outside the scope of possible intruders. Such a removal has been carried out by contracting with a data-processing agency in another country on a study where government interference was feared. Instead of removing the identification itself, a linkage file of matching reference numbers of data records and reference numbers of identification records was created and removed. This concept is sketched in Fig. 12-11. Matching,

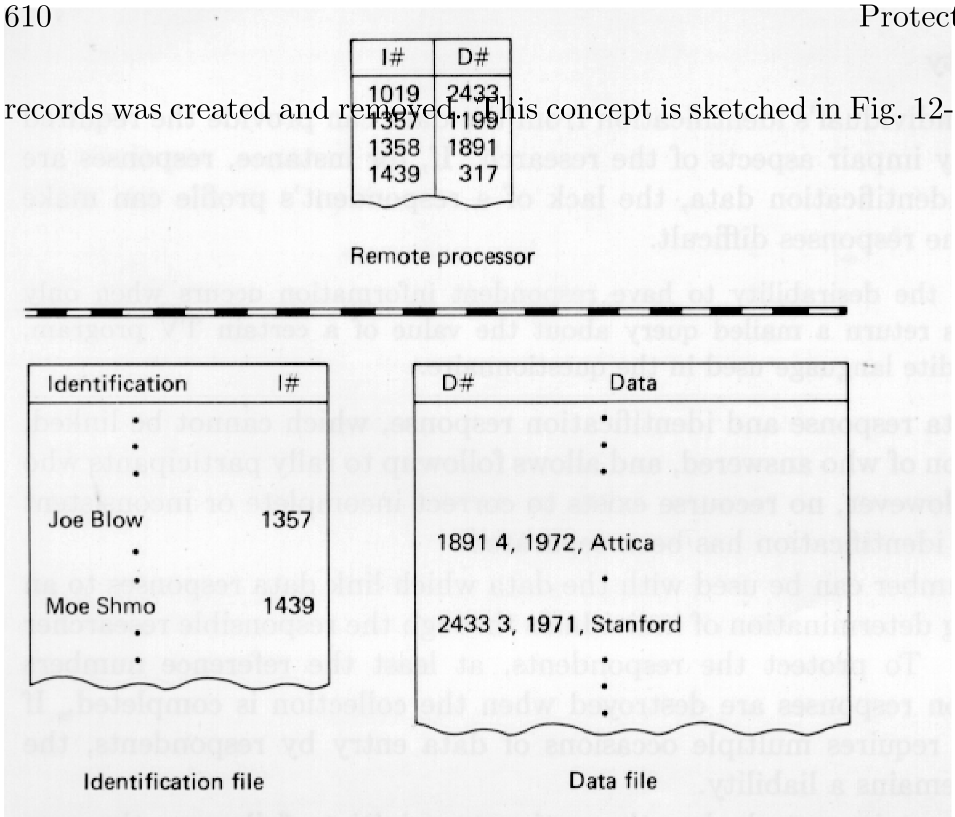


Figure 12-11 Remote linkage of data and identification.

when required, is carried out by the foreign operator, but the contract provides that only unidentified data are returned. The contract also specifies that the linkage file itself may not be returned, even to the contractees.

If data are to be matched at an insecure agency, but one is confident about one's own operation, enciphering of data other than the identification field may provide privacy.

12-8-3 Pollution

If the data are to be used only for statistical analysis, a systematic modification or *pollution* of the data may provide, if not privacy, at least protection from legal accountability. The modification should be such that statistical results remain valid but that individual data items cannot be ascertained.

A procedure of this type adds a normally distributed random number to a continuous observed data value. If the standard deviation of the normal is known, most statistical procedures can be used and their results adjusted. A lower level of confidence of the tested hypotheses is bound to occur because of the added degree of freedom in the data, but an increase in the number of observations can compensate for this loss. With binary data, such as yes/no responses, a certain number of the responses may be reversed on a random basis, which again can be accounted for in subsequent analyses. An individual record of such a file, while possibly embarrassing, is no longer of legal value.

The need to keep data private is an important responsibility in data-processing. The burden can be diminished by avoiding the collection of unnecessary data or unnecessary data elements. When data has lost its information value, it should be destroyed.

BACKGROUND AND REFERENCES

Westin^{71} and Miller^{71} survey history and legal issues of privacy protection. Petersen^{67} describes the threats to computer security. Papers on abuse of privacy have been collected and analyzed (Parker^{81}) at SRI International. VanTassel^{72} also provides samples and guidelines. Carrol^{72} shows how private data are used and misused. Many of the cases involve collusion and fraud; technical improvements to protect database access would not have prevented many of the abuses. An environment where reliability and protection is considered important might have discouraged many of the attempts. A professional code of ethics for computer professionals appears as part of a self-assessment procedure of the ACM in Weiss^{82}.

Data-processing practice and the use of RSS is described in IBM G320-1370–1376^{74}. Two conferences sponsored by NBS (Renniger^{74}) provide an overview of security and privacy concerns. Communication with a protected area is considered by Lampson^{73}. Evans^{74} and Purdy^{74} decribes mechanisms to keep passwords secure. Authentication of users can be aided by cryptography as shown by Gifford^{82} and Lindsay^{79}. The problem of protecting individual data from statistical access has been frequently analyzed. Contributors are Schwartz^{79}, Denning^{80}, Beck^{80}, Chin^{81}, Schlorer^{81}, and Ozsoyoglu^{82}.

The move instruction is proposed by Lewin^{69}. Hoffman^{77} surveys access protection oriented toward privacy problems. Saltzer^{75} provides a comprehensive summary of protection techniques in operating systems and includes a bibliography. Systems with protection features of interest are described by Weissman^{69}, Molho^{70}, Needham^{72}, Feustel^{73}, and Fabry^{74}.

Graham^{68} presents an extensive model of accessors versus data objects and applies it to several systems. A simple model is formally analyzed by Harrison^{76}. That the union of access privileges can be excessive is shown by Minsky^{81}. The access model shown here was developed by a SHARE committee (Wiederhold^{72}) as part of a requirements study.

MULTICS presents a major effort in the area of protection. Its development can be followed in Daley^{65}, Graham^{68}, Schroeder^{72}, and Saltzer^{74}. Carrol^{71} describes a system with adjustable object size for access control. Hoffman^{71} protects data by procedures controlled by users. The accessor-object matrix is described by Conway^{72}. How objects are managed is considered by Lindsay^{79}. Owens in Codd^{71} critically reviews many approaches and makes new suggestions.

Friedman^{70} defines cliques. Access control rules for relational databases are given by Fernandez^{81}. Stonebraker in King^{75} uses integrity assertions (see Example 9-10 and Sec. 13-3) to assure privacy protection. Capabilities were introduced by Dennis^{66}, and were used by Fabry^{74} in 1967. Access privileges are also granted to other users in a method developed by Griffiths^{76} and improved by Fagin^{78}. A schema-based approach with similar non-centralized privilege granting is developed by Arditi^{78}. Recovery in distributed systems is surveyed by Kohler^{81}.

The basic book for cryptographers is Kahn^{67}. Meyer^{73} describes hardware. Mellen^{73} surveys available methods; Stahl^{73} presents homophonic enciphering. Friedman^{74} evaluates the cost of ciphering. Feistel^{73} and Diffie^{76} present cryptology for computers. Gudes^{80} develops the use for file systems. Bayer^{76} applies enciphering to index trees and considers decryption and Davida^{81} encrypts fields within records. Needham^{78} applies encryption to users in distributed systems.

An implementation of DES for files is described by Konheim^{80}. Bright^{76} evaluated keys obtained from random-number generators. Gifford^{82} considers both communication and databases. Pollution to protect privacy is considered by Reed^{73}.

EXERCISES

1 a Construct an example and evaluate the possible cost of accidental disclosure. Document the case as if to demonstrate current system weaknesses to a manager in a commercial computer installation.

b Evaluate the loss for the case of Victor Velocity and of Sam Salesman in the introduction of the chapter.

c Set up an interview protocol to determine the value of privacy of a juvenile police record for the members of a freshman college class.

2 A sign-on procedure consists of the entering of a name and the entering of a character password. Three trials at a password are allowed, and if no correct password is given, the accessor is logged off. Which provides more security: increasing the password from three to four characters or adding 1 minute to the logoff procedure? The following operational times are to be assumed:

$T(\text{name}) = 2$ s
$T(\text{password}) = 0.1$ s/character
T(response: "`bad password`") $= 1$ s
$T(\text{basic logoff}) = 1$ s

96 different characters are allowed in the password.

3 a When is the above answer valid and when not?

b What is the actual expected average time required for these alternatives?

4 A file has an internal 32-bit key. The call and test to open the file lock takes 20 μs. What is the expected safe time against trial-and-error entry?

5 Given the computer system you are currently using, investigate the types of access control available for user's data stored:

a In primary memory

b In secondary storage

6 How would you implement some of the access types listed in Sec. 12-3 that are not now available on the equipment you are currently using?

7 Rewrite the file-transposition program of Example 4-1 with a `MOVE` operation so that the privacy remains protected. Is this adequate?

8 How do the DBTG specification satisfy Saltzer's criteria in Sec. 12-6.

9 Augment Fig. 3-39 with procedures and tables required to achieve privacy for the work history of the employees.

10 Determine the key used and decrypt the source plain-text message for the following cryptogram. You know that the encoding is a 3-character Vignere cipher.

`KMIRLLFXXRXL␣WXRXSUNIKCDDJIKXD␣SD␣QMLQD`

11 Write a program to do homophonic enciphering and deciphering. Then make a ranked frequency graph or table on the observed character frequency of some enciphered text.