include the following:[116]

> *Read* access provides users with the capability to view information in a system resource (such as a file, certain records, certain fields, or some combination thereof), but not to *alter* it, such as delete from, add to, or modify in any way. One must assume that information can be copied and printed if it can be read (although perhaps only manually, such as by using a print screen function and retyping the information into another file).
>
> *Write* access allows users to add to, modify, or delete information in system resources (e.g., files, records, programs). Normally user have read access to anything they have write access to.
>
> *Execute* privilege allows users to run programs.
>
> *Delete* access allows users to erase system resources (e.g., files, records, fields, programs).[117] Note that if users have write access but not delete access, they could overwrite the field or file with gibberish or otherwise inaccurate information and, in effect, delete the information.

Other specialized access modes (more often found in applications) include:

> *Create* access allows users to create new files, records, or fields.
>
> *Search* access allows users to list the files in a directory.

Of course, these criteria can be used in conjunction with one another. For example, an organization may give authorized individuals write access to an application at any time from within the office but only read access during normal working hours if they dial-in.

Depending upon the technical mechanisms available to implement logical access control, a wide variety of access permissions and restrictions are possible. No discussion can present all possibilities.

## 17.2 Policy: The Impetus for Access Controls

Logical access controls are a technical means of implementing *policy decisions*. Policy is made by

---

[116] These access modes are described generically; exact definitions and capabilities will vary from implementation to implementation. Readers are advised to consult their system and application documentation.

[117] "Deleting" information does not necessarily physically remove the data from the storage media. This can have serious implications for information that must be kept confidential. See "Disposition of Sensitive Automated Information," CSL Bulletin, NIST, October 1992.

a management official responsible for a particular system, application, subsystem, or group of systems. The development of an access control policy may not be an easy endeavor. It requires balancing the often-competing interests of security, operational requirements, and user-friendliness. In addition, technical constraints have to be considered.

This chapter discusses issues relating to the technical implementation of logical access controls – not the actual policy decisions as to who *should* have what type of access. These decisions are typically included in system-specific policy, as discussed in Chapters 5 and 10.

Once these policy decisions have been made, they will be *implemented* (or *enforced*) through logical access controls. In doing so, it is important to realize that the capabilities of various types of technical mechanisms (for logical access control) vary greatly.[118]

A few *simple* examples of *specific policy issues* are provided below; it is important to recognize, however, that *comprehensive system-specific policy* is significantly more complex.

1. The director of an organization's personnel office could decide that all clerks can update all files, to increase the efficiency of the office. Or the director could decide that clerks can only view and update specific files, to help prevent information browsing.

2. In a disbursing office, a single individual is usually prohibited from both requesting and authorizing that a particular payment be made. This is a *policy decision* taken to reduce the likelihood of embezzlement and fraud.

3. Decisions may also be made regarding access to the system itself. In the government, for example, the senior information resources management official may decide that agency systems that process information protected by the Privacy Act may not be used to process public-access database applications.

## 17.3 Technical Implementation Mechanisms

Many mechanisms have been developed to provide internal and external access controls, and they vary significantly in terms of precision, sophistication, and cost. These methods are not mutually exclusive and are often employed in combination. Managers need to analyze their organization's protection requirements to select the most appropriate, cost-effective logical access controls.

### 17.3.1 Internal Access Controls

*Internal* access controls are a logical means of separating what defined users (or user groups) can or cannot do with system resources. Five methods of internal access control are discussed in this section: passwords, encryption, access control lists, constrained user interfaces, and labels.

[118] Some policies may not be technically implementable; appropriate technical controls may simply not exist.

#### 17.3.1.1 Passwords

Passwords are most often associated with user authentication. (See Chapter 16.) However, they are also used to protect data and applications on many systems, including PCs. For instance, an accounting application may require a password to access certain financial data or to invoke a restricted application (or function of an application).[119]

> The use of passwords as a means of access control can result in a proliferation of passwords that can reduce overall security.

Password-based access control is often inexpensive because it is already included in a large variety of applications. However, users may find it difficult to remember additional application passwords, which, if written down or poorly chosen, can lead to their compromise. Password-based access controls for PC applications are often easy to circumvent if the user has access to the operating system (and knowledge of what to do). As discussed in Chapter 16, there are other disadvantages to using passwords.

#### 17.3.1.2 Encryption

Another mechanism that can be used for logical access control is encryption. Encrypted information can only be decrypted by those possessing the appropriate cryptographic key. This is especially useful if strong physical access controls cannot be provided, such as for laptops or floppy diskettes. Thus, for example, if information is encrypted on a laptop computer, and the laptop is stolen, the information cannot be accessed. While encryption can provide strong access control, it is accompanied by the need for strong key management. Use of encryption may also affect availability. For example, lost or stolen keys or read/write errors may prevent the decryption of the information. (See the cryptography chapter.)

#### 17.3.1.3 Access Control Lists

Access Control Lists (ACLs) refer to a register of: (1) users (including groups, machines, processes) who have been given permission to use a particular system resource, and (2) the types of access they have been permitted.

ACLs vary considerably in their capability and flexibility. Some only allow specifications for certain pre-set groups (e.g., owner, group, and world) while more advanced ACLs allow much more flexibility, such as *user-defined* groups. Also, more advanced ACLs can be used to explicitly *deny* access to a particular individual or group. With more advanced ACLs, access can be at the discretion of the policymaker (and implemented by the security administrator) or

[119] Note that this password is normally *in addition* to the one supplied initially to log onto the system.

individual user, depending upon how the controls are technically implemented.

*Elementary ACLs.* Elementary ACLs (e.g., "permission bits") are a widely available means of providing access control on multiuser systems. In this scheme, a short, predefined list of the access rights to files or other system resources is maintained.

Elementary ACLs are typically based on the concepts of *owner*, *group*, and *world*. For each of these, a set of access modes (typically chosen from read, write, execute, and delete) is specified by the owner (or custodian) of the resource. The owner is usually its creator, though in some cases, ownership of resources may be automatically assigned to project administrators, regardless of the identity of the creator. File owners often have all privileges for their resources.

Example of Elementary ACL for the file "payroll":

Owner: PAYMANAGER
Access: Read, Write, Execute, Delete

Group: COMPENSATION-OFFICE
Access: Read, Write, Execute, Delete

"World"
Access: None

In addition to the privileges assigned to the owner, each resource is associated with *a named group of users*. Users who are members of the group can be granted modes of access distinct from nonmembers, who belong to the rest of the "world" that includes all of the system's users. User groups may be arranged according to departments, projects, or other ways appropriate for the particular organization. For example, groups may be established for members of the Personnel and Accounting departments. The system administrator is normally responsible for technically maintaining and changing the membership of a group, based upon input from the owners/custodians of the particular resources to which the groups may be granted access.

As the name implies, however, the technology is not particularly flexible. It may not be possible to explicitly deny access to an individual who is a member of the file's group. Also, it may not be possible for two groups to easily share information (without exposing it to the "world"), since the list is predefined to only include one group. If two groups wish to share information, an owner may make the file

Since one would presume that no one would have access without being granted access, why would it be desirable to explicitly deny access? Consider a situation in which a group name has already been established for 50 employees. If it were desired to exclude five of the individuals from that group, it would be easier for the access control administrator to simply grant access to that group and take it away from the five rather than grant access to 45 people. Or, consider the case of a complex application in which many groups of users are defined. It may be desired, for some reason, to prohibit Ms. *X* from generating a particular report (perhaps she is under investigation). In a situation in which group names are used (and perhaps modified by others), this explicit denial may be a safety check to restrict Ms. *X*'s access – in case someone were to redefine a group (with access to the report generation function) to include Ms. *X*. She would still be denied access.

available to be read by "world." This may disclose information that should be restricted. Unfortunately, elementary ACLs have no mechanism to easily permit such sharing.

*Advanced ACLs.* Like elementary ACLs, advanced ACLs provide a form of access control based upon a logical registry. They do, however, provide *finer precision* in control.

Advanced ACLs can be very useful in many complex information sharing situations. They provide a great deal of flexibility in implementing system-specific policy and allow for customization to meet the security requirements of functional managers. Their flexibility also makes them more of a challenge to manage. The rules for determining access in the face of apparently conflicting ACL entries are not uniform across all implementations and can be confusing to security administrators. When such systems are introduced, they should be coupled with training to ensure their correct use.

Example of Advanced ACL for the file "payroll"

| | | | | |
|---|---|---|---|---|
| PAYMGR: | R, | W, | E, | D |
| J. Anderson: | | R, | W, | E, - |
| L. Carnahan: | | -, | -, | -, - |
| B. Guttman: | R, | W, | E, | - |
| E. Roback: | R, | W, | E, | - |
| H. Smith: | R, | -, | -, | - |
| PAY-OFFICE: | | R, | -, | -, - |
| WORLD: | -, | -, | -, | - |

**17.3.1.4 Constrained User Interfaces**

Often used in conjunction with ACLs are *constrained user interfaces*, which restrict users' access to specific functions by never allowing them to request the use of information, functions, or other specific system resources for which they do not have access. Three major types exist: (1) *menus*, (2) *database views*, and (3) *physically constrained user interfaces*.

Constrained user interfaces can provide a form of access control that closely models how an organization operates. Many systems allow administrators to restrict users' ability to use the operating system or application system directly. Users can only execute commands that are provided by the administrator, typically in the form of a *menu*. Another means of restricting users is through restricted *shells* which limit the system commands the user can invoke. The use of menus and shells can often make the system easier to use and can help reduce errors.

Menu-driven systems are a common constrained user interface, where different users are provided different menus on the same system.

*Database views* is a mechanism for restricting user access to data contained in a database. It may be necessary to allow a user to access a database, but that user may not need access to all the data in the database (e.g., not all fields of a record nor all records in the database). Views can be used to enforce complex access requirements that are often needed in database situations, such as those based on the content of a field. For example, consider the situation where clerks maintain

personnel records in a database. Clerks are assigned a range of clients based upon last name (e.g., A-C, D-G). Instead of granting a user access to all records, the view can grant the user access to the record based upon the first letter of the last name field.

*Physically* constrained user interfaces can also limit a user's abilities. A common example is an ATM machine, which provides only a limited number of physical buttons to select options; no alphabetic keyboard is usually present.

### 17.3.1.5 Security Labels

A security label is a designation assigned to a resource (such as a file). Labels can be used for a variety of purposes, including controlling access, specifying protective measures, or indicating additional handling instructions. In many implementations, once this designator has been set, it cannot be changed (except perhaps under carefully controlled conditions that are subject to auditing).

**Data Categorization**

One tool that is used to increase the ease of security labelling is categorizing data by similar protection requirements. For example, a label could be developed for "organization proprietary data." This label would mark information that can be disclosed only to the organization's employees. Another label, "public data" could be used to mark information that is available to anyone.

When used for access control, labels are also assigned to *user sessions*. Users are permitted to initiate sessions with specific labels only. For example, a file bearing the label "Organization Proprietary Information" would not be accessible (readable) except during user sessions with the corresponding label. Moreover, only a restricted set of users would be able to initiate such sessions. The labels of the session and those of the files accessed during the session are used, in turn, to label output from the session. This ensures that information is uniformly protected throughout its life on the system.

For systems with stringent security requirements (such as those processing national security information), labels may be useful in access control.

Labels are a very strong form of access control; however, they are often inflexible and can be expensive to administer. Unlike permission bits or access control lists, labels cannot ordinarily be changed. Since labels are permanently linked to specific information, data cannot be disclosed by a user copying information and changing the access to that file so that the information is more accessible than the original owner intended. By removing users' ability to arbitrarily designate the accessibility of files they own, opportunities for certain kinds of human errors and malicious software problems are eliminated. In the example above, it would not be possible to copy Organization Proprietary Information into a file with a different label. This prevents inappropriate disclosure, but can interfere with legitimate extraction of some information.

Labels are well suited for consistently and uniformly enforcing access restrictions, although their administration and inflexibility can be a significant deterrent to their use.

### 17.3.2 External Access Controls

*External* access controls are a means of controlling interactions between the system and outside people, systems, and services. External access controls use a wide variety of methods, often including a separate physical device (e.g., a computer) that is between the system being protected and a network.

> One of the most common PPDs is the *dial-back modem*. A typical dial-back modem sequence follows: a user calls the dial-back modem and enters a password. The modem hangs up on the user and performs a table lookup for the password provided. If the password is found, the modem places a return call to the user (at a previously specified number) to initiate the session. The return call itself also helps to protect against the use of lost or compromised accounts. This is, however, not always the case. Malicious hackers can use such advance functions as call forwarding to reroute calls.

#### 17.3.2.1 Port Protection Devices

Fitted to a communications port of a host computer, a port protection device (PPD) authorizes access to the port itself, prior to and independent of the computer's own access control functions. A PPD can be a separate device in the communications stream,[120] or it may be incorporated into a communications device (e.g., a modem). PPDs typically require a separate authenticator, such as a password, in order to access the communications port.

#### 17.3.2.2 Secure Gateways/Firewalls

Often called *firewalls*, secure gateways block or filter access between two networks, often between a private[121] network and a larger, more public network such as the Internet, which attract malicious hackers. Secure gateways allow internal users to connect to external networks and at the same time prevent malicious hackers from compromising the internal systems.[122]

Some secure gateways are set up to allow all traffic to pass through except for specific traffic

---

[120] Typically PPDs are found only in serial communications streams.

[121] *Private network* is somewhat of a misnomer. *Private* does not mean that the organization's network is totally inaccessible to outsiders or prohibits use of the outside network from insiders (or the network would be disconnected). It also does not mean that all the information on the network requires confidentiality protection. It does mean that a network (or part of a network) is, in some way, separated from another network.

[122] Questions frequently arise as to whether secure gateways help prevent the spread of viruses. In general, having a gateway scan transmitted files for viruses requires more system overhead than is practical, especially since the scanning would have to handle many different file formats. However, secure gateways may reduce the spread of network worms.

which has known or suspected vulnerabilities or security problems, such as remote log-in services. Other secure gateways are set up to disallow all traffic except for specific types, such as e-mail. Some secure gateways can make access-control decisions based on the location of the requester. There are several technical approaches and mechanisms used to support secure gateways.

Because gateways provide security by restricting services or traffic, they can affect a system's usage. For this reason, firewall experts always emphasize the need for policy, so that appropriate officials decide how the organization will balance operational needs and security.

**Types of Secure Gateways**

There are many types of secure gateways. Some of the most common are packet filtering (or screening) routers, proxy hosts, bastion hosts, dual-homed gateways, and screened-host gateways.

In addition to reducing the risks from malicious hackers, secure gateways have several other benefits. They can reduce internal system security overhead, since they allow an organization to concentrate security efforts on a limited number of machines. (This is similar to putting a guard on the first floor of a building instead of needing a guard on every floor.)

A second benefit is the centralization of services. A secure gateway can be used to provide a central management point for various services, such as advanced authentication (discussed in Chapter 16), e-mail, or public dissemination of information. Having a central management point can reduce system overhead and improve service.

#### 17.3.2.3 Host-Based Authentication

Host-based authentication grants access based upon the *identity of the host* originating the request, instead of the identity of the user making the request. Many network applications in use today use host-based authentication to determine whether access is allowed. Under certain circumstances it is fairly easy to masquerade as the legitimate host, especially if the masquerading host is physically located close to the host being impersonated. Security measures to protect against misuse of some host-based authentication systems are available (e.g., Secure RPC[123] uses DES to provide a more secure identification of the client host).

An example of host-based authentication is the Network File System (NFS) which allows a server to make file systems/directories available to specific machines.

## 17.4 Administration of Access Controls

[123] RPC, or Remote Procedure Call, is the service used to implement NFS.

One of the most complex and challenging aspects of access control, administration involves implementing, monitoring, modifying, testing, and terminating user accesses on the system. These can be demanding tasks, even though they typically do not include making the actual decisions as to the type of access each user may have.[124] Decisions regarding accesses should be guided by organizational policy, employee job descriptions and tasks, information sensitivity, user "need-to-know" determinations, and many other factors.

There are three basic approaches to administering access controls: centralized, decentralized, or a combination of these. Each has relative advantages and disadvantages. Which is most appropriate in a given situation will depend upon the particular organization and its circumstances.

**System and Security Administration**

The administration of systems and security requires access to advanced functions (such as setting up a user account). The individuals who technically set up and modify who has access to what are very powerful users on the system; they are often called system or security administrators. On some systems, these users are referred to as having *privileged accounts*.

The type of access of these accounts varies considerably. Some administrator privileges, for example, may allow an individual to administer only one application or subsystem, while a higher level of privileges may allow for oversight and establishment of subsystem administrators.

Normally, users who are security administrators have two accounts: one for regular use and one for security use. This can help protect the security account from compromise. Furthermore, additional I&A precautions, such as ensuring that administrator passwords are robust and changed regularly, are important to minimize opportunities for unauthorized individuals to gain access to these functions.

### 17.4.1 Centralized Administration

Using centralized administration, one office or individual is responsible for configuring access controls. As users' information processing needs change, their accesses can be modified only through the central office, usually after requests have been approved by the appropriate official. This allows very strict control over information, because the ability to make changes resides with very few individuals. Each user's account can be centrally monitored, and closing all accesses for any user can be easily accomplished if that individual leaves the organization. Since relatively few individuals oversee the process, consistent and uniform procedures and criteria are usually not difficult to enforce. However, when changes are needed quickly, going through a central administration office can be frustrating and time-consuming.

[124] As discussed in the policy section earlier in this chapter, those decisions are usually the responsibility of the applicable application manager or cognizant management official. See also the discussion of system-specific policy in Chapters 5 and 10.

### 17.4.2 Decentralized Administration

In decentralized administration, access is directly controlled by the owners or creators of the files, often the functional manager. This keeps control in the hands of those most accountable for the information, most familiar with it and its uses, and best able to judge who needs what kind of access. This may lead, however, to a lack of consistency among owners/creators as to procedures and criteria for granting user accesses and capabilities. Also, when requests are not processed centrally, it may be much more difficult to form a systemwide composite view of all user accesses on the system at any given time. Different application or data owners may inadvertently implement combinations of accesses that introduce conflicts of interest or that are in some other way not in the organization's best interest.[125] It may also be difficult to ensure that all accesses are properly terminated when an employee transfers internally or leaves an organization.

### 17.4.3 Hybrid Approach

A hybrid approach combines centralized and decentralized administration. One typical arrangement is that central administration is responsible for the broadest and most basic accesses, and the owners/creators of files control types of accesses or changes in users' abilities for the files under their control. The main disadvantage to a hybrid approach is adequately defining which accesses should be assignable locally and which should be assignable centrally.

## 17.5 Coordinating Access Controls

It is vital that access controls protecting a system work together. At a minimum, three basic types of access controls should be considered: physical, operating system, and application. In general, access controls within an application are the most specific. However, for application access controls to be fully effective they need to be supported by operating system access controls. Otherwise access can be made to application resources without going through the application.[126] Operating system and application access controls need to be supported by physical access controls.

## 17.6 Interdependencies

Logical access controls are closely related to many other controls. Several of them have been discussed in the chapter.

---

[125] Without necessary review mechanisms, central administration does not *a priori* preclude this.

[126] For example, logical access controls within an application block User A from viewing File F. However, if operating systems access controls do not also block User A from viewing File F, User A can use a utility program (or another application) to view the file.

*Policy and Personnel.* The most fundamental interdependencies of logical access control are with policy and personnel. Logical access controls are the technical implementation of system-specific and organizational policy, which stipulates *who* should be able to access what kinds of information, applications, and functions. These decisions are normally based on the principles of separation of duties and least privilege.

*Audit Trails.* As discussed earlier, logical access controls can be difficult to implement correctly. Also, it is sometimes *not possible* to make logical access control as precise, or fine-grained, as would be ideal for an organization. In such situations, users may either deliberately or inadvertently abuse their access. For example, access controls cannot prevent a user from modifying data the user is authorized to modify, even if the modification is incorrect. Auditing provides a way to identify abuse of access permissions. It also provides a means to review the actions of system or security administrators.

*Identification and Authentication.* In most logical access control scenarios, the identity of the user must be established before an access control decision can be made. The access control process then associates the permissible forms of accesses with that identity. This means that access control can only be as effective as the I&A process employed for the system.

*Physical Access Control.* Most systems can be compromised if someone can physically access the machine (i.e., CPU or other major components) by, for example, restarting the system with different software. Logical access controls are, therefore, dependent on physical access controls (with the exception of encryption, which can depend solely on the strength of the algorithm and the secrecy of the key).

## 17.7 Cost Considerations

Incorporating logical access controls into a computer system involves the purchase or use of access control mechanisms, their implementation, and changes in user behavior.

*Direct Costs*. Among the direct costs associated with the use of logical access controls are the purchase and support of hardware, operating systems, and applications that provide the controls, and any add-on security packages. The most significant personnel cost in relation to logical access control is usually for administration (e.g., initially determining, assigning, and keeping access rights up to date). Label-based access control is available in a limited number of commercial products, but at greater cost and with less variety of selection. Role-based systems are becoming more available, but there are significant costs involved in customizing these systems for a particular organization. Training users to understand and use an access control system is another necessary cost.

*Indirect Costs*. The primary indirect cost associated with introducing logical access controls into

a computer system is the effect on user productivity. There may be additional overhead involved in having individual users properly determine (when under their control) the protection attributes of information. Another indirect cost that may arise results from users not being able to immediately access information necessary to accomplish their jobs because the permissions were incorrectly assigned (or have changed). This situation is familiar to most organizations that put strong emphasis on logical access controls.

## References

Abrams, M.D., et al. *A Generalized Framework for Access Control: An Informal Description*. McLean, VA: Mitre Corporation, 1990.

Baldwin, R.W. "Naming and Grouping Privileges to Simplify Security Management in Large Databases." *1990 IEEE Symposium on Security and Privacy Proceedings*. Oakland, CA: IEEE Computer Society Press, May 1990. pp. 116-132.

Caelli, William, Dennis Longley, and Michael Shain. *Information Security Handbook*. New York, NY: Stockton Press, 1991.

Cheswick, William, and Steven Bellovin. *Firewalls and Internet Security*. Reading, MA: Addison-Wesley Publishing Company, 1994.

Curry, D. *Improving the Security of Your UNIX System, ITSTD-721-FR-90-21*. Menlo Park, CA: SRI International, 1990.

Dinkel, Charles. *Secure Data Network System Access Control Documents*. NISTIR 90-4259. Gaithersburg, MD: National Institute of Standards and Technology, 1990.

Fites, P., and M. Kratz. *Information Systems Security: A Practitioner's Reference*. New York, NY: Van Nostrand Reinhold, 1993. Especially Chapters 1, 9, and 12.

Garfinkel, S., and Spafford, G. "UNIX Security Checklist." *Practical UNIX Security*. Sebastopol, CA: O'Riley & Associates. Inc., 1991. pp. 401-413.

Gasser, Morrie. *Building a Secure Computer System*. New York, NY: Van Nostrand Reinhold, 1988.

Haykin, M., and R. Warner. *Smart Card Technology: New Methods for Computer Access Control*. Spec Pub 500-157. Gaithersburg, MD: National Institute of Standards and Technology, 1988.

Landwehr, C., C. Heitmeyer, and J. McLean. "A Security Model for Military Message Systems." *ACM Transactions on Computer Systems*, Vol. 2, No. 3, August 1984.

National Bureau of Standards. *Guidelines for Security of Computer Applications*. Federal Information Processing Standard Publication 73. June 1980.

Pfleeger, Charles. *Security in Computing*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1989.

President's Council on Integrity and Efficiency. *Review of General Controls in Federal Computer Systems*. Washington, DC: President's Council on Integrity and Efficiency, October 1988.

S. Salamone, "Internetwork Security: Unsafe at Any Node?" *Data Communications*. 22(12), 1993. pp. 61-68.

Sandhu, R. "Transaction Control Expressions for Separation of Duty." *Fourth Annual Computer Security Applications Conference Proceedings*. Orlando, FL, December 1988, pp. 282-286.

Thomsen, D.J. "Role-based Application Design and Enforcement." *Fourth IFIP Workshop on Database Security Proceedings.* International Federation for Information Processing, Halifax, England, September 1990.

T. Whiting. "Understanding VAX/VMS Security." *Computers and Security*. 11(8), 1992. pp. 695-698.

# Chapter 18

# AUDIT TRAILS

Audit trails maintain a record of system activity both by system and application processes and by user activity of systems and applications.[127] In conjunction with appropriate tools and procedures, audit trails can assist in detecting security violations, performance problems, and flaws in applications.[128]

**The Difference Between Audit Trails and Auditing**

An *audit trail* is a series of records of computer events, about an operating system, an application, or user activities. A computer system may have several audit trails, each devoted to a particular type of activity.

*Auditing* is the review and analysis of management, operational, and technical controls. The auditor can obtain valuable information about activity on a computer system from the audit trail. Audit trails improve the *auditability* of the computer system. Auditing is discussed in the assurance chapter.

Audit trails may be used as either a support for regular system operations or a kind of insurance policy or as both of these. As insurance, audit trails are maintained but are not used unless needed, such as after a system outage. As a support for operations, audit trails are used to help system administrators ensure that the system or resources have not been harmed by hackers, insiders, or technical problems.

This chapter focuses on audit trails as a technical control, rather than the process of security auditing, which is a review and analysis of the security of a system as discussed in Chapter 9. This chapter discusses the benefits and objectives of audit trails, the types of audit trails, and some common implementation issues.

## 18.1 Benefits and Objectives

An *event* is any action that happens on a computer system. Examples include logging into a system, executing a program, and opening a file.

Audit trails can provide a means to help accomplish *several* security-related objectives, including individual accountability,

[127] Some security experts make a distinction between an *audit trail* and an *audit log* as follows: a *log* is a record of events made by a particular software package, and an *audit trail* is an entire history of an event, possibly using several logs. However, common usage within the security community does not make use of this definition. Therefore, this document does not distinguish between trails and logs.

[128] The type and amount of detail recorded by audit trails vary by both the technical capability of the logging application and the managerial decisions. Therefore, when we state that "audit trails can...," the reader should be aware that capabilities vary widely.

reconstruction of events, intrusion detection, and problem analysis.

### 18.1.1 Individual Accountability

Audit trails are a technical mechanism that help managers maintain individual accountability. By advising users that they are personally accountable for their actions, which are tracked by an audit trail that logs user activities, managers can help promote proper user behavior.[129] Users are less likely to attempt to circumvent security policy if they know that their actions will be recorded in an audit log.

For example, audit trails can be used in concert with access controls to identify and provide information about users suspected of improper modification of data (e.g., introducing errors into a database). An audit trail may record "before" and "after" versions of records. (Depending upon the size of the file and the capabilities of the audit logging tools, this may be very resource-intensive.) Comparisons can then be made between the actual changes made to records and what was expected. This can help management determine if errors were made by the user, by the system or application software, or by some other source.

Audit trails work in concert with logical access controls, which restrict use of system resources. Granting users access to particular resources usually means that they need that access to accomplish their job. Authorized access, of course, can be misused, which is where audit trail analysis is useful. While users cannot be prevented from using resources to which they have legitimate access authorization, audit trail analysis is used to examine their actions. For example, consider a personnel office in which users have access to those personnel records for which they are responsible. Audit trails can reveal that an individual is printing far more records than the average user, which could indicate the selling of personal data. Another example may be an engineer who is using a computer for the design of a new product. Audit trail analysis could reveal that an outgoing modem was used extensively by the engineer the week before quitting. This could be used to investigate whether proprietary data files were sent to an unauthorized party.

### 18.1.2 Reconstruction of Events

Audit trails can also be used to reconstruct events after a problem has occurred. Damage can be more easily assessed by reviewing audit trails of system activity to pinpoint how, when, and why normal operations ceased. Audit trail analysis can often distinguish between operator-induced errors (during which the system may have performed exactly as instructed) or system-created errors (e.g., arising from a poorly tested piece of replacement code). If, for example, a system fails or the integrity of a file (either program or data) is questioned, an analysis of the audit trail

[129] For a fuller discussion of changing employee behavior, see Chapter 13.

can reconstruct the series of steps taken by the system, the users, and the application. Knowledge of the conditions that existed at the time of, for example, a system crash, can be useful in avoiding future outages. Additionally, if a technical problem occurs (e.g., the corruption of a data file) audit trails can aid in the recovery process (e.g., by using the record of changes made to reconstruct the file).

### 18.1.3 Intrusion Detection

*Intrusion detection* refers to the process of identifying attempts to penetrate a system and gain unauthorized access.

If audit trails have been designed and implemented to record appropriate information, they can assist in intrusion detection. Although normally thought of as a real-time effort, intrusions can be detected *in real time*, by examining audit records as they are created (or through the use of other kinds of warning flags/notices), or *after the fact* (e.g., by examining audit records in a batch process).

Real-time intrusion detection is primarily aimed at outsiders attempting to gain unauthorized access to the system. It may also be used to detect changes in the system's performance indicative of, for example, a virus or worm attack.[130] There may be difficulties in implementing real-time auditing, including unacceptable system performance.

After-the-fact identification may indicate that unauthorized access was attempted (or was successful). Attention can then be given to damage assessment or reviewing controls that were attacked.

### 18.1.4 Problem Analysis

Audit trails may also be used as on-line tools to help identify problems other than intrusions as they occur. This is often referred to as *real-time auditing* or monitoring. If a system or application is deemed to be critical to an organization's business or mission, real-time auditing may be implemented to monitor the status of these processes (although, as noted above, there can be difficulties with real-time analysis). An analysis of the audit trails may be able to verify that the *system* operated normally (i.e., that an error may have resulted from operator error, as opposed to a system-originated error). Such use of audit trails may be complemented by system performance logs. For example, a significant increase in the use of system resources (e.g., disk file space or outgoing modem use) *could* indicate a security problem.

[130] Viruses and worms of forms of malicious code. A virus is a code segment that replicates by attaching copies of itself to existing executables. A worm is a self-replicating program.

## 18.2 Audit Trails and Logs

A system can maintain several different audit trails concurrently. There are typically two kinds of audit records, (1) an event-oriented log and (2) a record of every keystroke, often called keystroke monitoring. Event-based logs usually contain records describing *system* events, *application* events, or *user* events.

An audit trail should include sufficient information to establish what events occurred and who (or what) caused them. In general, an event record should specify when the event occurred, the user ID associated with the event, the program or command used to initiate the event, and the result. Date and time can help determine if the user was a masquerader or the actual person specified.

### 18.2.1 Keystroke Monitoring[131]

Keystroke monitoring is the process used to view or record both the keystrokes entered by a computer user and the computer's response during an interactive session. Keystroke monitoring is usually considered a special case of audit trails. Examples of keystroke monitoring would include viewing characters as they are typed by users, reading users' electronic mail, and viewing other recorded information typed by users.

Some forms of routine system maintenance may record user keystrokes. This could constitute keystroke monitoring if the keystrokes are preserved along with the user identification so that an administrator could determine the keystrokes entered by specific users. Keystroke monitoring is conducted in an effort to protect systems and data from intruders who access the systems without authority or in excess of their assigned authority. Monitoring keystrokes typed by intruders can help administrators assess and repair damage caused by intruders.

### 18.2.2 Audit Events

*System audit records* are generally used to monitor and fine-tune system performance. *Application audit trails* may be used to discern flaws in applications, or violations of security policy committed within an application. *User audits records* are generally used to hold individuals accountable for their actions. An analysis of user audit records may expose a variety

[131] The Department of Justice has advised that an ambiguity in U.S. law makes it unclear whether keystroke monitoring is considered equivalent to an unauthorized telephone wiretap. The ambiguity results from the fact that current laws were written years before such concerns as keystroke monitoring or system intruders became prevalent. Additionally, no legal precedent has been set to determine whether keystroke monitoring is legal or illegal. System administrators conducting such monitoring might be subject to criminal and civil liabilities. The Department of Justice advises system administrators to protect themselves by giving notice to system users if keystroke monitoring is being conducted. Notice should include agency/organization policy statements, training on the subject, and a banner notice on each system being monitored. [NIST, *CSL Bulletin*, March 1993]

of security violations, which might range from simple browsing to attempts to plant Trojan horses or gain unauthorized privileges.

**Sample System Log File Showing Authentication Messages**

```
Jan 27 17:14:04 host1 login: ROOT LOGIN console
Jan 27 17:15:04 host1 shutdown: reboot by root
Jan 27 17:18:38 host1 login: ROOT LOGIN console
Jan 27 17:19:37 host1 reboot: rebooted by root
Jan 28 09:46:53 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
Jan 28 09:47:35 host1 shutdown: reboot by user1
Jan 28 09:53:24 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 12 08:53:22 host1 su: 'su root' succeeded for user1 on /dev/ttyp1
Feb 17 08:57:50 host1 date: set by user1
Feb 17 13:22:52 host1 su: 'su root' succeeded for user1 on /dev/ttyp0
```

The system itself enforces certain aspects of policy (particularly *system-specific* policy) such as access to files and access to the system itself. Monitoring the alteration of systems configuration files that implement the policy is important. If special accesses (e.g., security administrator access) have to be used to alter configuration files, the system should generate audit records whenever these accesses are used.

**Application-Level Audit Record for a Mail Delivery System**

```
Apr  9 11:20:22 host1 AA06370: from=<user2@host2>, size=3355, class=0
Apr  9 11:20:23 host1 AA06370: to=<user1@host1>, delay=00:00:02,
stat=Sent
Apr  9 11:59:51 host1 AA06436: from=<user4@host3>, size=1424, class=0
Apr  9 11:59:52 host1 AA06436: to=<user1@host1>, delay=00:00:02,
stat=Sent
Apr  9 12:43:52 host1 AA06441: from=<user2@host2>, size=2077, class=0
Apr  9 12:43:53 host1 AA06441: to=<user1@host1>, delay=00:00:01,
stat=Sent
```

Sometimes a finer level of detail than system audit trails is required. *Application audit trails* can provide this greater level of recorded detail. If an application is critical, it can be desirable to record not only who invoked the application, but certain details specific to each use. For example, consider an e-mail application. It may be desirable to record who sent mail, as well as to whom they sent mail and the length of messages. Another example would be that of a database application. It may be useful to record who accessed what database as well as the individual rows or columns of a table that were read (or changed or deleted), instead of just recording the execution of the database program.

**User Log Showing a Chronological List of Commands Executed by Users**

```
rcp         user1     ttyp0     0.02 secs Fri Apr  8 16:02
ls          user1     ttyp0     0.14 secs Fri Apr  8 16:01
clear       user1     ttyp0     0.05 secs Fri Apr  8 16:01
rpcinfo     user1     ttyp0     0.20 secs Fri Apr  8 16:01
nroff       user2     ttyp2     0.75 secs Fri Apr  8 16:00
sh          user2     ttyp2     0.02 secs Fri Apr  8 16:00
mv          user2     ttyp2     0.02 secs Fri Apr  8 16:00
sh          user2     ttyp2     0.03 secs Fri Apr  8 16:00
col         user2     ttyp2     0.09 secs Fri Apr  8 16:00
man         user2     ttyp2     0.14 secs Fri Apr  8 15:57
```

A *user audit trail* monitors and logs user activity in a system or application by recording events initiated by the user (e.g., access of a file, record or field, use of a modem).

Flexibility is a critical feature of audit trails. Ideally (from a security point of view), a system administrator would have the ability to monitor all system and user activity, but could choose to log only certain functions at the system level, and within certain applications. The decision of how much to log and how much to review should be a function of application/data sensitivity and should be decided by each functional manager/application owner with guidance from the system administrator and the computer security manager/officer, weighing the costs and benefits of the logging.[132]

### 18.2.2.1 System-Level Audit Trails

A system audit trail should be able to identify failed log-on attempts, especially if the system does not limit the number of failed log-on attempts. Unfortunately, some system-level audit trails cannot detect attempted log-ons, and therefore, cannot log them for later review. These audit trails can only monitor and log successful log-ons and subsequent activity. To effectively detect intrusion, a record of failed log-on attempts is required.

If a system-level audit capability exists, the audit trail should capture, at a minimum, any attempt to log on (successful or unsuccessful), the log-on ID, date and time of each log-on attempt, date and time of each log-off, the devices used, and the function(s) performed once logged on (e.g., the applications that the user tried, successfully or unsuccessfully, to invoke). System-level logging also typically includes information that is not specifically security-related, such as system operations, cost-accounting charges, and network performance.

[132] In general, audit logging can have privacy implications. Users should be aware of applicable privacy laws, regulations, and policies that may apply in such situations.

#### 18.2.2.2 Application-Level Audit Trails

System-level audit trails may not be able to track and log events *within* applications, or may not be able to provide the level of detail needed by application or data owners, the system administrator, or the computer security manager. In general, application-level audit trails monitor and log user activities, including data files opened and closed, specific actions, such as reading, editing, and deleting records or fields, and printing reports. Some applications may be sensitive enough from a data availability, confidentiality, and/or integrity perspective that a "before" and "after" picture of each modified record (or the data element(s) changed within a record) should be captured by the audit trail.

#### 18.2.2.3 User Audit Trails

User audit trails can usually log:

- all commands directly initiated by the user;
- all identification and authentication attempts; and
- files and resources accessed.

It is most useful if options and parameters are also recorded from commands. It is much more useful to know that a user tried to delete a log file (e.g., to hide unauthorized actions) than to know the user merely issued the delete command, possibly for a personal data file.

## 18.3 Implementation Issues

Audit trail data requires protection, since the data should be available for use when needed and is not useful if it is not accurate. *Also, the best planned and implemented audit trail is of limited value without timely review of the logged data.* Audit trails may be reviewed periodically, as needed (often triggered by occurrence of a security event), automatically in realtime, or in some combination of these. System managers and administrators, with

**Audit Logs for Physical Access**

Physical access control systems (e.g., a card/key entry system or an alarm system) use software and audit trails similar to general-purpose computers. The following are *examples* of criteria that may be used in selecting which events to log:

The date and time the access was attempted or made should be logged, as should the gate or door through which the access was attempted or made, and the individual (or user ID) making the attempt to access the gate or door.

Invalid attempts should be monitored and logged by noncomputer audit trails just as they are for computer-system audit trails. Management should be made aware if someone attempts to gain access during unauthorized hours.

Logged information should also include attempts to add, modify, or delete physical access privileges (e.g., granting a new employee access to the building or granting transferred employees access to their new office [and, of course, deleting their old access, as applicable]).

As with system and application audit trails, auditing of noncomputer functions can be implemented to send messages to security personnel indicating valid or invalid attempts to gain access to controlled spaces. In order not to desensitize a guard or monitor, all access should not result in messages being sent to a screen. Only exceptions, such as failed access attempts, should be highlighted to those monitoring access.

guidance from computer security personnel, should determine how long audit trail data will be maintained – either on the system or in archive files.

Following are examples of implementation issues that may have to be addressed when using audit trails.

### 18.3.1 Protecting Audit Trail Data

Access to on-line audit logs should be strictly controlled. Computer security managers and system administrators or managers should have access for review purposes; however, security and/or administration personnel who maintain logical access functions may have no need for access to audit logs.

It is particularly important to ensure the *integrity* of audit trail data against modification. One way to do this is to use digital signatures. (See Chapter 19.) Another way is to use write-once devices. The audit trail files needs to be protected since, for example, intruders may try to "cover their tracks" by modifying audit trail records. Audit trail records should be protected by strong access controls to help prevent unauthorized access. The integrity of audit trail information may be particularly important when legal issues arise, such as when audit trails are used as legal evidence. (This may, for example, require daily printing and signing of the logs.) Questions of such legal issues should be directed to the cognizant legal counsel.

The confidentiality of audit trail information may also be protected, for example, if the audit trail is recording information about users that may be disclosure-sensitive such as transaction data containing personal information (e.g., "before" and "after" records of modification to income tax data). Strong access controls and encryption can be particularly effective in preserving confidentiality.

### 18.3.2 Review of Audit Trails

Audit trails can be used to review what occurred after an event, for periodic reviews, and for real-time analysis. Reviewers should know what to look for to be effective in spotting unusual activity. They need to understand what normal activity looks like. Audit trail review can be easier if the audit trail function can be queried by user ID, terminal ID, application name, date and time, or some other set of parameters to run reports of selected information.

*Audit Trail Review After an Event*. Following a known system or application software problem, a known violation of existing requirements by a user, or some unexplained system or user problem, the appropriate system-level or application-level administrator should review the audit trails. Review by the application/data owner would normally involve a separate report, based upon audit trail data, to determine if their resources are being misused.

*Periodic Review of Audit Trail Data*. Application owners, data owners, system administrators, data processing function managers, and computer security managers should determine how much review of audit trail records is necessary, based on the importance of identifying unauthorized activities. This determination should have a direct correlation to the frequency of periodic reviews of audit trail data.

*Real-Time Audit Analysis*. Traditionally, audit trails are analyzed in a batch mode at regular intervals (e.g., daily). Audit records are archived during that interval for later analysis. Audit analysis tools can also be used in a real-time, or near real-time fashion. Such intrusion detection tools are based on audit reduction, attack signature, and variance techniques. Manual review of audit records in real time is almost never feasible on large multiuser systems due to the volume of records generated. However, it might be possible to view all records associated with a particular user or application, and view them in real time.[133]

### 18.3.3 Tools for Audit Trail Analysis

Many types of tools have been developed to help to reduce the amount of information contained in audit records, as well as to distill useful information from the raw data. Especially on larger systems, audit trail software can create very large files, which can be extremely difficult to analyze manually. The use of automated tools is likely to be the difference between unused audit trail data and a robust program. Some of the types of tools include:

*Audit reduction tools* are preprocessors designed to reduce the volume of audit records to facilitate manual review. Before a security review, these tools can remove many audit records known to have little security significance. (This alone may cut in half the number of records in the audit trail.) These tools generally remove records generated by specified classes of events, such as records generated by nightly backups might be removed.

*Trends/variance-detection tools* look for anomalies in user or system behavior. It is possible to construct more sophisticated processors that monitor usage trends and detect major variations. For example, if a user typically logs in at 9 a.m., but appears at 4:30 a.m. one morning, this may indicate a security problem that may need to be investigated.

*Attack signature-detection tools* look for an *attack signature*, which is a specific sequence of events indicative of an unauthorized access attempt. A simple example would be repeated failed log-in attempts.

[133] This is similar to keystroke monitoring, though, and may be legally restricted.

## 18.4 Interdependencies

The ability to audit supports many of the controls presented in this handbook. The following paragraphs describe some of the most important interdependencies.

*Policy*. The most fundamental interdependency of audit trails is with policy. Policy dictates who is authorized access to what system resources. Therefore it specifies, directly or indirectly, what violations of policy should be identified through audit trails.

*Assurance*. System auditing is an important aspect of operational assurance. The data recorded into an audit trail is used to support a system audit. The analysis of audit trail data and the process of auditing systems are closely linked; in some cases, they may even be the same thing. In most cases, the analysis of audit trail data is a critical part of maintaining operational assurance.

*Identification and Authentication*. Audit trails are tools often used to help hold users accountable for their actions. To be held accountable, the users must be known to the system (usually accomplished through the identification and authentication process). However, as mentioned earlier, audit trails record events and associate them with the *perceived* user (i.e., the user ID). If a user is impersonated, the audit trail will establish events but *not* the identity of the user.

*Logical Access Control*. Logical access controls restrict the use of system resources to authorized users. Audit trails complement this activity in two ways. First, they may be used to identify breakdowns in logical access controls or to verify that access control restrictions are behaving as expected, for example, if a particular user is erroneously included in a group permitted access to a file. Second, audit trails are used *to audit use of resources by those who have legitimate access*. Additionally, to protect audit trail files, access controls are used to ensure that audit trails are not modified.

*Contingency Planning*. Audit trails assist in contingency planning by leaving a record of activities performed on the system or within a specific application. In the event of a technical malfunction, this log can be used to help reconstruct the state of the system (or specific files).

*Incident Response*. If a security incident occurs, such as hacking, audit records and other intrusion detection methods can be used to help determine the extent of the incident. For example, was just one file browsed, or was a Trojan horse planted to collect passwords?

*Cryptography*. Digital signatures can be used to protect audit trails from undetected modification. (This does not prevent deletion or modification of the audit trail, but will provide an alert that the audit trail has been altered.) Digital signatures can also be used in conjunction with adding secure time stamps to audit records. Encryption can be used if confidentiality of audit trail information is important.

## 18.5 Cost Considerations

Audit trails involve many costs. First, some system overhead is incurred recording the audit trail. Additional system overhead will be incurred storing and processing the records. The more detailed the records, the more overhead is required. Another cost involves human and machine time required to do the analysis. This can be minimized by using tools to perform most of the analysis. Many simple analyzers can be constructed quickly (and cheaply) from system utilities, but they are limited to audit reduction and identifying particularly sensitive events. More complex tools that identify trends or sequences of events are slowly becoming available as off-the-shelf software. (If complex tools are not available for a system, development may be prohibitively expensive. Some intrusion detection systems, for example, have taken years to develop.)

The final cost of audit trails is the cost of investigating anomalous events. If the system is identifying too many events as suspicious, administrators may spend undue time reconstructing events and questioning personnel.

## References

Fites, P., and M. Kratz. *Information Systems Security: A Practitioner's Reference*. New York: Van Nostrand Reinhold, 1993, (especially Chapter 12, pp. 331 - 350).

Kim, G., and E. Spafford, "Monitoring File System Integrity on UNIX Platforms." *Infosecurity News*. 4(4), 1993. pp. 21-22.

Lunt, T. "Automated Audit Trail Analysis for Intrusion Detection," *Computer Audit Update*, April 1992. pp. 2-8.

National Computer Security Center. *A Guide to Understanding Audit in Trusted Systems*. NCSC-TG-001, Version-2. Ft. Meade, MD, 1988.

National Institute of Standards and Technology. "Guidance on the Legality of Keystroke Monitoring." *CSL Bulletin*. March 1993.

Phillips, P. W. "New Approach Identifies Malicious System Activity." *Signal*. 46(7), 1992. pp. 65-66.

Ruthberg, Z., et al. *Guide to Auditing for Controls and Security: A System Development Life Cycle Approach*. Special Publication 500-153. Gaithersburg, MD: National Bureau of Standards, 1988.

Stoll, Clifford. *The Cuckoo's Egg*. New York, NY: Doubleday, 1989.

# Chapter 19

# CRYPTOGRAPHY

Cryptography is a branch of mathematics based on the transformation of data. It provides an important tool for protecting information and is used in many aspects of computer security. For example, cryptography can help provide data confidentiality, integrity, electronic signatures, and advanced user authentication. Although modern cryptography relies upon advanced mathematics, users can reap its benefits without understanding its mathematical underpinnings.

This chapter describes cryptography as a tool for satisfying a wide spectrum of computer security needs and requirements. It describes fundamental aspects of the basic cryptographic technologies and some specific ways cryptography can be applied to improve security. The chapter also explores some of the important issues that should be considered when incorporating cryptography into computer systems.

> Cryptography is traditionally associated only with keeping data secret. However, modern cryptography can be used to provide many security services, such as electronic signatures and ensuring that data has not been modified.

## 19.1 Basic Cryptographic Technologies

Cryptography relies upon two basic components: an *algorithm* (or cryptographic methodology) and a *key*. In modern cryptographic systems, algorithms are complex mathematical formulae and keys are strings of bits. For two parties to communicate, they must use the same algorithm (or algorithms that are designed to work together). In some cases, they must also use the same key. Many cryptographic keys must be kept secret; sometimes algorithms are also kept secret.

> There are two basic types of cryptography: "secret key" and "public key."

There are two basic types of cryptography: *secret key systems* (also called symmetric systems) and *public key systems* (also called asymmetric systems). Table 19.1 compares some of the distinct features of secret and public key systems. Both types of systems offer advantages and disadvantages. Often, the two are combined to form a *hybrid system* to exploit the strengths of each type. To determine which type of cryptography best meets its needs, an organization first has to identify its security requirements and operating environment.

| DISTINCT FEATURES | SECRET KEY CRYPTOGRAPHY | PUBLIC KEY CRYPTOGRAPHY |
|---|---|---|
| NUMBER OF KEYS | Single key. | Pair of keys. |
| TYPES OF KEYS | Key is secret. | One key is private, and one key is public. |
| PROTECTION OF KEYS | Disclosure and modification. | Disclosure and modification for private keys and modification for public keys. |
| RELATIVE SPEEDS | Faster. | Slower. |

Table 19.1

### 19.1.1 Secret Key Cryptography

In secret key cryptography, two (or more) parties share the same key, and that key is used to encrypt and decrypt data. As the name implies, secret key cryptography relies on keeping the key secret. If the key is compromised, the security offered by cryptography is severely reduced or eliminated. Secret key cryptography assumes that the parties who share a key rely upon each other not to disclose the key and protect it against modification.

Secret key cryptography has been in use for centuries. Early forms merely transposed the written characters to hide the message.

The best known secret key system is the *Data Encryption Standard* (DES), published by NIST as Federal Information Processing Standard (FIPS) 46-2. Although the adequacy of DES has at times been questioned, these claims remain unsubstantiated, and DES remains strong. It is the most widely accepted, publicly available cryptographic system today. The American National Standards Institute (ANSI) has adopted DES as the basis for encryption, integrity, access control, and key management standards.

The *Escrowed Encryption Standard*, published as FIPS 185, also makes use of a secret key system. (See the discussion of Key Escrow Encryption in this chapter.)

### 19.1.2 Public Key Cryptography

Public key cryptography is a modern invention and requires the use of advanced mathematics.

Whereas secret key cryptography uses a single key shared by two (or more) parties, public key cryptography uses a pair of keys for *each* party. One of the keys of the pair is "public" and the other is "private." The public key can be made known to other parties; the private key must be kept confidential and must be known only to its owner. Both keys, however, need to be protected against modification.

Public key cryptography is particularly useful when the parties wishing to communicate cannot rely upon each other or do not share a common key. There are several public key cryptographic systems. One of the first public key systems is RSA, which can provide many different security services. The Digital Signature Standard (DSS), described later in the chapter, is another example of a public key system.

### 19.1.3 Hybrid Cryptographic Systems

Public and secret key cryptography have relative advantages and disadvantages. Although public key cryptography does not require users to share a common key, secret key cryptography is much faster: equivalent implementations of secret key cryptography can run 1,000 to 10,000 times faster than public key cryptography.

Secret key systems are often used for bulk data encryption and public key systems for automated key distribution.

To maximize the advantages and minimize the disadvantages of both secret and public key cryptography, a computer system can use both types in a complementary manner, with each performing different functions. Typically, the speed advantage of secret key cryptography means that it is used for encrypting data. Public key cryptography is used for applications that are less demanding to a computer system's resources, such as encrypting the keys used by secret key cryptography (for distribution) or to sign messages.

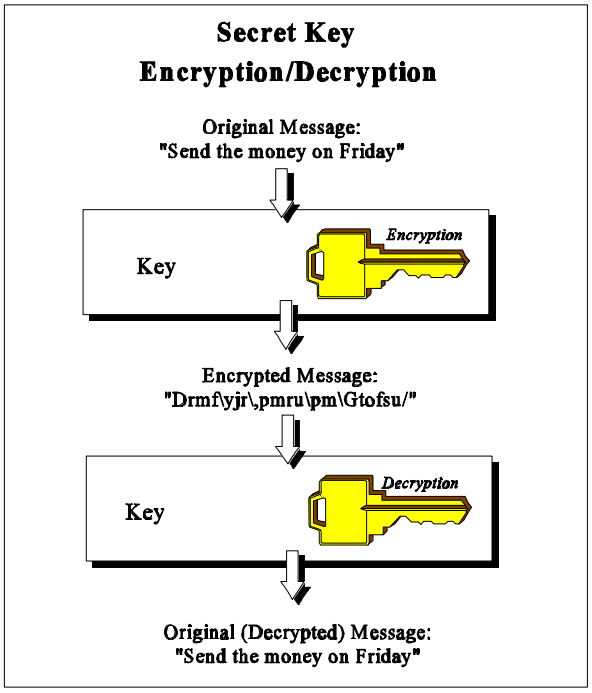
### 19.1.4 Key Escrow

Because cryptography can provide extremely strong encryption, it can thwart the government's efforts to lawfully perform electronic surveillance. For example, if strong cryptography is used to encrypt a phone conversation, a court-authorized wiretap will not be effective. To meet the needs of the government *and* to provide privacy, the federal government has adopted voluntary key escrow cryptography. This technology allows the use of strong encryption, but also allows the government when legally authorized to obtain decryption keys held by escrow agents. NIST has published the *Escrowed Encryption Standard* as FIPS 185. Under the Federal Government's

voluntary key escrow initiative, the decryption keys are split into parts and given to separate escrow authorities. Access to one part of the key does *not* help decrypt the data; both keys must be obtained.

## 19.2 Uses of Cryptography

Cryptography is used to protect data *both* inside and outside the boundaries of a computer system. Outside the computer system, cryptography is sometimes the *only* way to protect data. While in a computer system, data is normally protected with logical and physical access controls (perhaps supplemented by cryptography). However, when in transit across communications lines or resident on someone else's computer, data cannot be protected by the originator's[134] logical or physical access controls. Cryptography provides a solution by protecting data even when the data is no longer in the control of the originator.




### 19.2.1 Data Encryption

One of the best ways to obtain cost-effective data confidentiality is through the use of encryption. Encryption transforms intelligible data, called *plaintext*,[135] into an unintelligible form, called *ciphertext*. This process is reversed through the process of decryption. Once data is encrypted, the ciphertext does not have to be protected against disclosure. However, if ciphertext is modified, it will not decrypt correctly.

Both secret key and public key cryptography can be used for data encryption although not all public key algorithms provide for data encryption.

To use a secret key algorithm, data is encrypted using a key. The same key must be used to

[134] The originator does not have to be the original creator of the data. It can also be a guardian or custodian of the data.

[135] Plaintext can be intelligible to a human (e.g., a novel) or to a machine (e.g., executable code).