NETWORK SECURITY

Introduction

The requirements of information security within an organization have undergone two major changes in the last several decades. Before the widespread use of data processing equipment, the security of information felt to be valuable to an organization was provided primarily by physical and administrative means. An example of the former is the use of rugged filing cabinets with a combination lock for storing sensitive documents.

With the introduction of the computer, the need for automated tools for protecting files and other information stored on the computer became evident. This is especially the case for a shared system, such as a time-sharing system, and the need is even more acute for systems that can be accessed over a public telephone network, data network, or the Internet. The generic name for the collection of tools designed to protect data and to thwart hackers is computer security.

The second major change that affected security is the introduction of distributed systems and the use of networks and communications facilities for carrying data between terminal user and computer and between computer and computer. Network security measures are needed to protect data during their transmission. In fact, the term network security is somewhat misleading, because virtually all business, government, and academic organizations interconnect their data processing equipment with a collection of interconnected networks. Such a collection is often referred to as an internet.

There are no clear boundaries between these two forms of security. For example, one of the most publicized types of attack on information systems is the computer virus. A virus may be introduced into a system physically when it arrives on a diskette or optical disk and is subsequently loaded onto a computer. Viruses may also arrive over an internet. In either case, once the virus is resident on a computer system, internal computer security tools are needed to detect and recover from the virus.

Consider the following examples of security violations:

1. User A transmits a file to user B. The file contains sensitive information (e.g., payroll records) that is to be protected from disclosure. User C, who is not authorized to read the file, is able to monitor the transmission and capture a copy of the file during its transmission.

2. A network manager, D, transmits a message to a computer, E, under its management. The message instructs computer E to update an authorization file to include the identities of a number of new users who are to be given access to that computer. User F intercepts the message, alters its contents to add or delete entries, and then forwards the message to E, which accepts the message as coming from manager D and updates its authorization file accordingly.

3. Rather than intercept a message, user F constructs its own message with the desired entries and transmits that message to E as if it had come from manager D. Computer E accepts the message as coming from manager D and updates its authorization file accordingly.

4. An employee is fired without warning. The personnel manager sends a message to a server system to invalidate the employee's account. When the invalidation is accomplished, the server is to post a notice to the employee's file as confirmation of the action. The employee is able to intercept the message and delay it long enough to make a final access to the server to retrieve sensitive information. The message is then forwarded, the action taken, and the confirmation posted. The employee's action may go unnoticed for some considerable time.

Although this list by no means exhausts the possible types of security violations, it illustrates the range of concerns of network security.

Internetwork security is both fascinating and complex. Some of the reasons follow:

1. Security involving communications and networks is not as simple as it might first appear to the novice. The requirements seem to be straightforward; indeed, most of the major requirements for security services can be given self-explanatory one-word labels: confidentiality, authentication, nonrepudiation, integrity. But the mechanisms used to meet those requirements can be quite complex, and understanding them may involve rather subtle reasoning.

2. In developing a particular security mechanism or algorithm, one must always consider potential attacks on those security features. In many cases, successful attacks are designed by looking at the problem in a completely different way, therefore exploiting an unexpected weakness in the mechanism.

3. Because of point 2, the procedures used to provide particular services are often counterintuitive: It is not obvious from the statement of a particular requirement that such elaborate measures are needed. It is only when the various countermeasures are considered that the measures used make sense.

4. Having designed various security mechanisms, it is necessary to decide where to use them. This is true both in terms of physical placement (e.g., at what points in a network are certain security mechanisms needed) and in a logical sense [e.g., at what layer or layers of an architecture such as TCP/IP (Transmission Control Protocol/Internet Protocol) should mechanisms be placed].

5. Security mechanisms usually involve more than a particular algorithm or protocol. They usually also require that participants be in possession of some secret information (e.g., an encryption key), which raises questions about the creation, distribution, and protection of that secret information. There is also a reliance on communications protocols whose behavior may complicate the task of developing the security mechanism. For example, if the proper functioning of the security mechanism requires setting time limits on the transit time of a message from sender to receiver, then any protocol or network that introduces variable, unpredictable delays may render such time limits meaningless.

ATTACKS, SERVICES, AND MECHANISMS

To assess the security needs of an organization effectively and to evaluate and choose various security products and policies, the manager responsible for security needs some systematic way of defining the requirements for security and characterizing the approaches to satisfying those requirements. One approach is to consider three aspects of information security:

- **Security attack:** Any action that compromises the security of information owned by an organization.
- **Security mechanism:** A mechanism that is designed to detect, prevent, or recover from a security attack.
- **Security service:** A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks, and they make use of one or more security mechanisms to provide the service.

Security Attacks

Attacks on the security of a computer system or network are best characterized by viewing the function of the computer system as providing information. In general, there is a flow of information from a source, such as a file or a region of main memory, to a destination, such as another file or a user. This normal flow is depicted in figure 1. The remaining parts of the figure show the following four general categories of attack:

- **Interruption:** An asset of the system is destroyed or becomes unavailable or unusable. This is an attack on **availability.** Examples include destruction of a piece of hardware, such as a hard disk, the cutting of a communication line, or the disabling of the file management system.
- **Interception:** All unauthorized party gains access to an asset. this is an attack on confidentiality. The unauthorized party could be a person, a program, or a computer. Examples include wiretapping to capture data in a network, and the illicit copying of files or programs.

- **Modification:** An unauthorized party not only gains access to but tampers with an asset. This is an attack on **integrity.** Examples include changing values in a data file, altering a program so that it performs differently, and modifying the content of messages being transmitted in a network.

- **Fabrication:** An unauthorized party inserts counterfeit objects into the system. This is an attack on **authenticity.** Examples include the insertion of spurious messages in a network or the addition of records to a file.

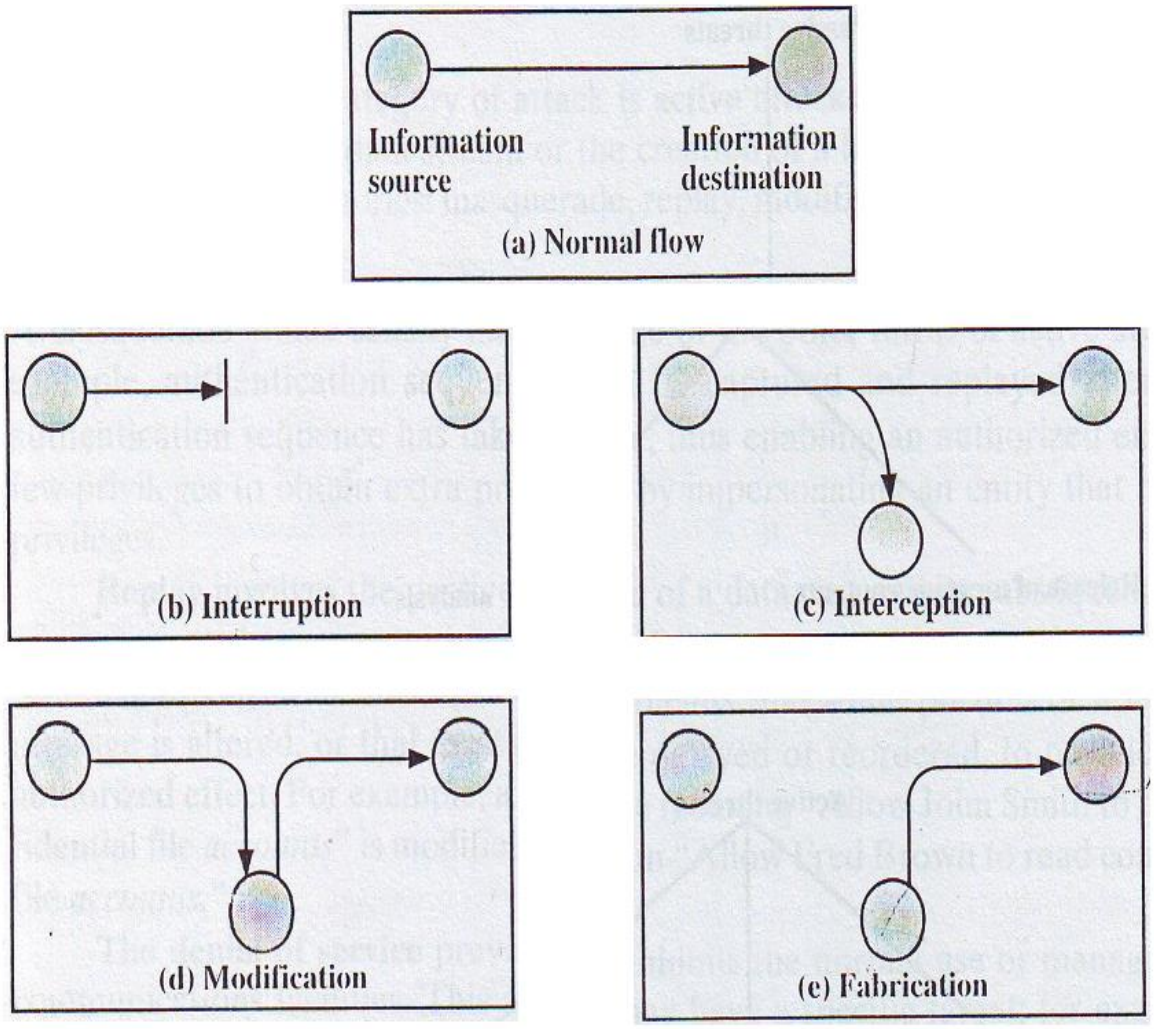


Figure 1:Security Attacks

A useful means of classifying security attacks is in terms of passive attacks and active attacks. A passive attack attempts to learn or make use of information from the system but does not affect system resources. An active attack attempts to alter system resources or affect their operation.

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Two types of passive attacks are release of message contents and traffic analysis.

The release of message contents is easily understood (Figure 2.a). A telephone conversation, an electronic mail message, and a transferred file may contain sensitive or confidential information. We would like to prevent an opponent from learning the contents of these transmissions.

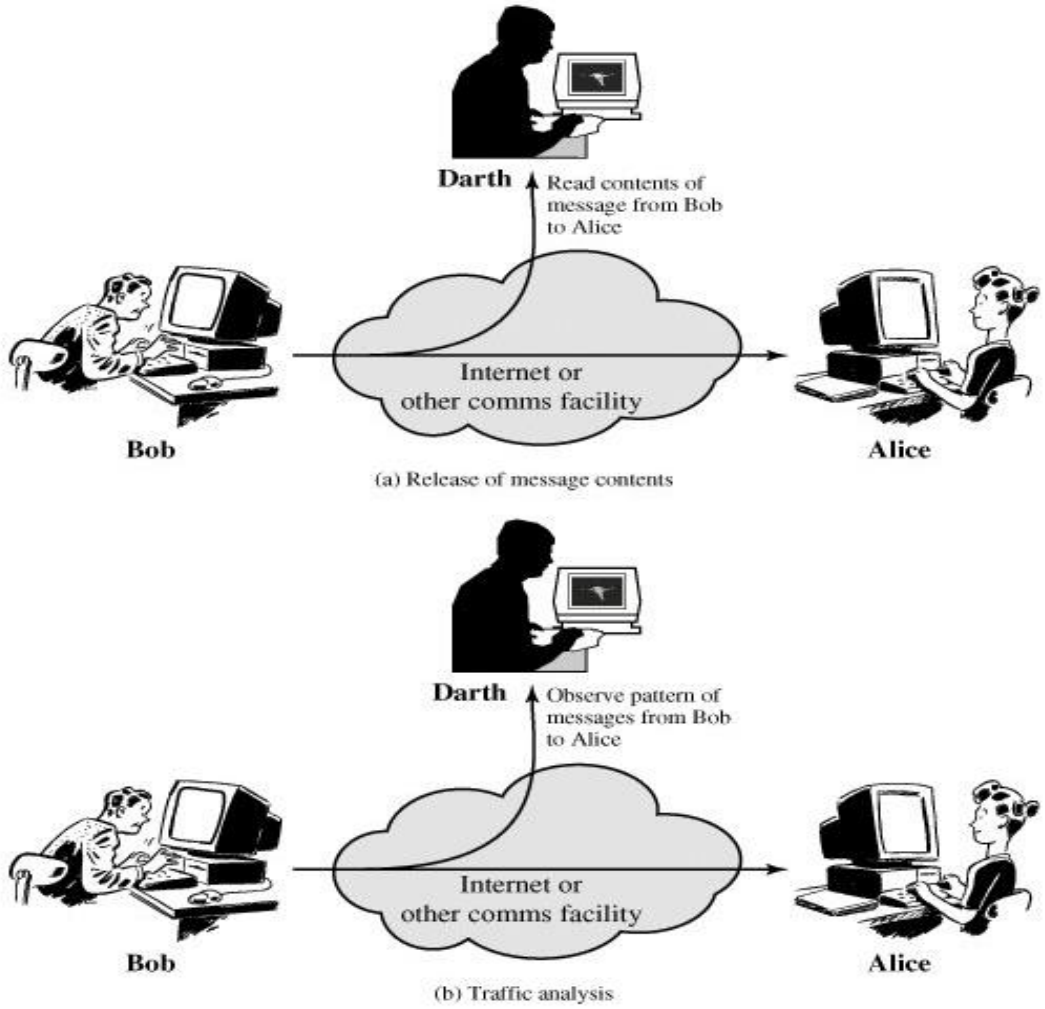


Figure 2:Passive Attacks

A second type of passive attack, **traffic analysis**, is subtler (Figure 2.b). Suppose that we had a way of masking the contents of messages or other information traffic so that opponents, even if they captured the message, could not extract the information from the message. The common technique for masking contents is encryption. If we had encryption protection in place, an opponent might still be able to observe the pattern of these messages. The opponent could determine the location and identity of communicating hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of the communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of the data. Typically, the message traffic is sent and received in an apparently normal fashion and neither the sender nor receiver is aware that a third party has read the messages or observed the traffic pattern. However, it is feasible to prevent the success of these attacks, usually by means of encryption. Thus, the emphasis in dealing with passive attacks is on prevention rather than detection.

Active Attacks

Active attacks involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: masquerade, replay, modification of messages, and denial of service.

A **masquerade** takes place when one entity pretends to be a different entity (Figure 3.a). A masquerade attack usually includes one of the other forms of active attack.

Replay involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect (Figure 3.b).

Modification of messages simply means that some portion of a legitimate message is altered, or that messages are delayed or reordered, to produce an unauthorized effect (Figure 3.c). For example, a message meaning "Allow John Smith to read confidential file accounts" is modified to mean "Allow Fred Brown to read confidential file accounts."

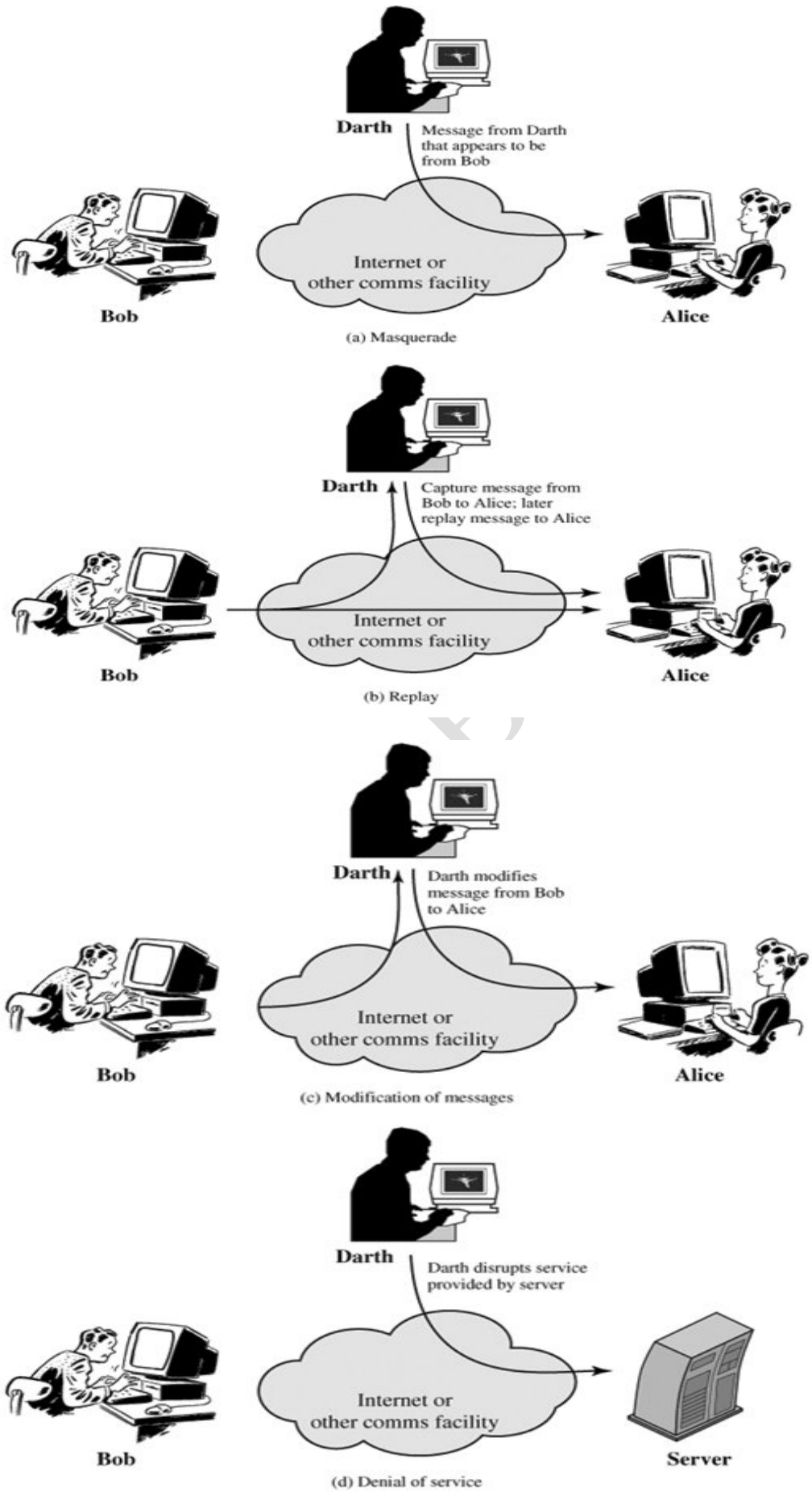


Figure 3:Active Attacks

The **denial of service** prevents or inhibits the normal use or management of communications facilities (Figure 3.d). This attack may have a specific target; for example, the disruption of an entire network, either by disabling the network or by overloading it with messages so as to degrade performance.

Active attacks present the opposite characteristics of passive attacks. Whereas passive attacks are difficult to detect, measures are available to prevent their success. On the other hand, it is quite difficult to prevent active attacks absolutely, because of the wide variety of potential physical, software, and network vulnerabilities. Instead, the goal is to detect active attacks and to recover from any disruption or delays caused by them. If the detection has a deterrent effect, it may also contribute to prevention.

Mechanisms

We can note that there is one particular element that underlies most of the security mechanisms in use :cryptographic techniques . Encryption or encryption-like Transformations of information are the most common means of providing security.

Security Services

Authentication:

The authentication service is concerned with assuring that a communication is authentic. In the case of a single message, such as a warning or alarm signal, the function of the authentication service is to assure the recipient that the message is from the source that it claims to be from. In the case of an ongoing interaction, such as the connection of a terminal to a host, two aspects are involved. First, at the time of connection initiation, the service assures that the two entities are authentic, that is, that each is the entity that it claims to be. Second, the service must assure that the connection is not interfered with in such a way that a third party can masquerade as one of the two legitimate parties for the purposes of unauthorized transmission or reception.

Access Control:

In the context of network security, access control is the ability to limit and control the access to host systems and applications via communications links. To achieve this, each entity trying to gain access must first be identified, or authenticated, so that access rights can be tailored to the individual.

Data Confidentiality:

Confidentiality is the protection of transmitted data from passive attacks. With respect to the content of a data transmission, several levels of protection can be identified. The broadest service protects all user data transmitted between two users over a period of time. For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection. Narrower forms of this service can also be defined, including the protection of a single message or even specific fields within a message. These refinements are less useful than the broad approach and may even be more complex and expensive to implement.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that an attacker not be able to observe the source and destination, frequency, length, or other characteristics of the traffic on a communications facility.

Data Integrity:

As with confidentiality, integrity can apply to a stream of messages, a single message, or selected fields within a message. Again, the most useful and straightforward approach is total stream protection.

A connection-oriented integrity service, one that deals with a stream of messages, assures that messages are received as sent, with no duplication, insertion, modification, reordering, or replays. The destruction of data is also covered under this service. Thus, the connection-oriented integrity service addresses both message stream modification and denial of service. On the other hand, a connectionless integrity service, one that deals with individual messages without regard to any larger context, generally provides protection against message modification only.

We can make a distinction between the service with and without recovery. Because the integrity service relates to active attacks, we are concerned with detection rather than prevention. If a violation of integrity is detected, then the service may simply report this violation, and some other portion of software or human intervention is required to recover from the violation. Alternatively, there are mechanisms available to recover from the loss of integrity of data, as we will review subsequently. The incorporation of automated recovery mechanisms is, in general, the more attractive alternative.

Nonrepudiation

Nonrepudiation prevents either sender or receiver from denying a transmitted message. Thus, when a message is sent, the receiver can prove that the alleged sender in fact sent the message. Similarly, when a message is received, the sender can prove that the alleged receiver in fact received the message.

Availability Service

A variety of attacks can result in the loss of or reduction in availability. Some of these attacks are amenable to automated countermeasures, such as authentication and encryption, whereas others require some sort of physical action to prevent or recover from loss of availability of elements of a distributed system.

A Model for Network Security

A model for much of what we will be discussing is captured, in very general terms, in Figure 4. A message is to be transferred from one party to another across some sort of internet. The two parties, who are the principals in this transaction, must cooperate for the exchange to take place. A logical information channel is established by defining a route through the internet from source to destination and by the cooperative use of communication protocols (e.g., TCP/IP) by the two principals.

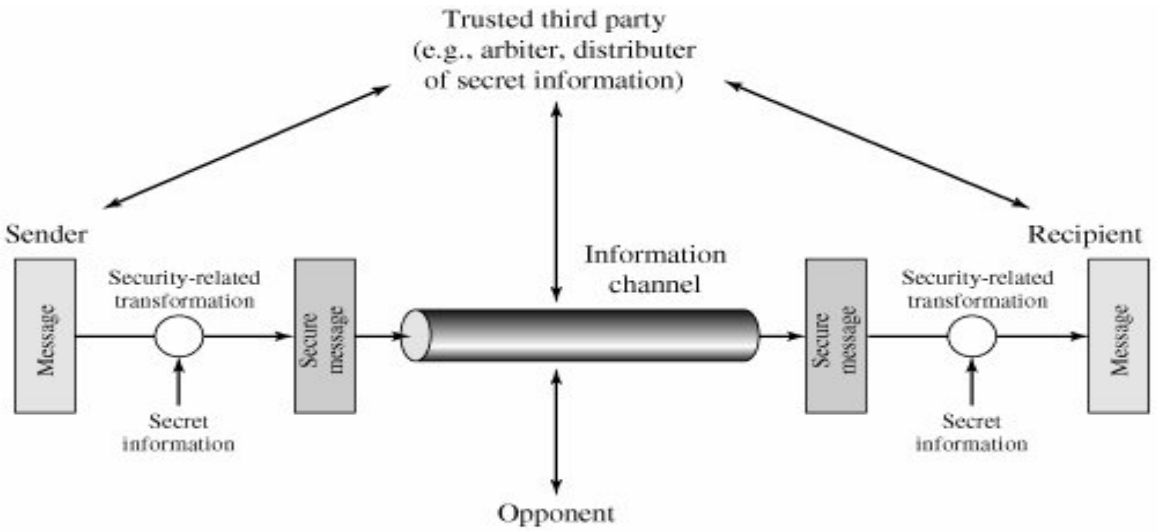


Figure 4:Model for Network Security

Security aspects come into play when it is necessary or desirable to protect the information transmission from an opponent who may present a threat to confidentiality, authenticity, and so on. All the techniques for providing security have two components:

- A security-related transformation on the information to be sent. Examples include the encryption of the message, which scrambles the message so that it is unreadable by the opponent, and the addition of a code based on the contents of the message, which can be used to verify the identity of the sender.
- Some secret information shared by the two principals and, it is hoped, unknown to the opponent. An example is an encryption key used in conjunction with the transformation to scramble the message before transmission and unscramble it on reception.

A trusted third party may be needed to achieve secure transmission. For example, a third party may be responsible for distributing the secret information to the two principals while keeping it from any opponent. Or a third party may be needed to arbitrate disputes between the two principals concerning the authenticity of a message transmission.

This general model shows that there are four basic tasks in designing a particular security service:

1. Design an algorithm for performing the security-related transformation. The algorithm should be such that an opponent cannot defeat its purpose.
2. Generate the secret information to be used with the algorithm.
3. Develop methods for the distribution and sharing of the secret information.
4. Specify a protocol to be used by the two principals that makes use of the security algorithm and the secret information to achieve a particular security service.

There are other security-related situations of interest that do not neatly fit this model but that are considered in this book. A general model of these other situations is illustrated by Figure 5, which reflects a concern for protecting an information system from unwanted access. Most readers are familiar with the concerns caused by the existence of hackers, who attempt to penetrate systems that can be accessed over a network. The hacker can be someone who, with no malign intent, simply gets satisfaction from breaking and entering a computer system. Or, the intruder can be a disgruntled employee who wishes to do damage, or a criminal who seeks to exploit computer assets for financial gain (e.g., obtaining credit card numbers or performing illegal money transfers).

Another type of unwanted access is the placement in a computer system of logic that exploits vulnerabilities in the system and that can affect application programs as well as utility programs, such as editors and compilers. Programs can present two kinds of threats:

- Information access threats intercept or modify data on behalf of users who should not have access to that data.
- Service threats exploit service flaws in computers to inhibit use by legitimate users.

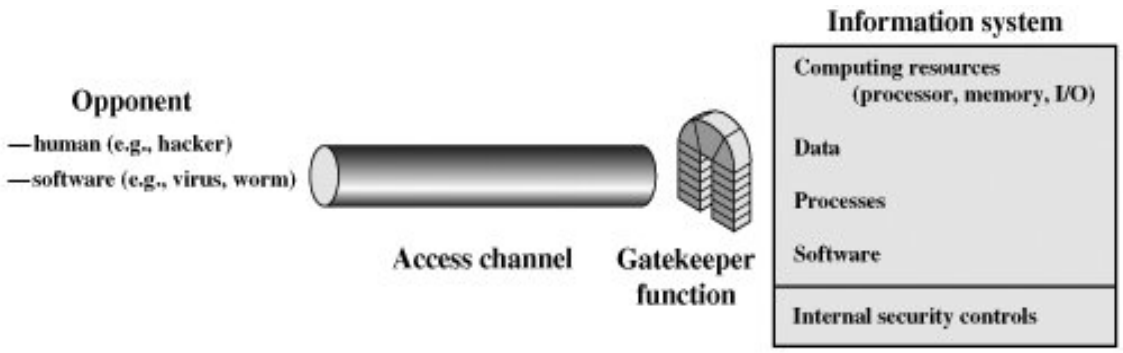


Figure 5: A Network Access Security

Viruses and worms are two examples of software attacks. Such attacks can be introduced into a system by means of a disk that contains the unwanted logic concealed in otherwise useful software. They can also be inserted into a system across a network; this latter mechanism is of more concern in network security.

The security mechanisms needed to cope with unwanted access fall into two broad categories (Figure 5). The first category might be termed a gatekeeper function. It includes password-based login procedures that are designed to deny access to all but authorized users and screening logic that is designed to detect and reject worms, viruses, and other similar attacks. Once either an unwanted user or unwanted software gains access, the second line of defense consists of a variety of internal controls that monitor activity and analyze stored information in an attempt to detect the presence of unwanted intruders.

Intruders

One of the two most publicized threats to security is the intruder (the other is viruses), generally referred to as a hacker or cracker. There are three classes of intruders:

- **Masquerader**: An individual who is not authorized to use the computer and who penetrates a system's access controls to exploit a legitimate user's account
- **Misfeasor**: A legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuses his or her privileges
- **Clandestine user**: An individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection

The masquerader is likely to be an outsider; the misfeasor generally is an insider; and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system.

Intrusion Techniques

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruder to acquire information that should have been protected. In some cases, this information is in the form of a user password. With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user.

Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways:

- One-way function: The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way

transformation (not reversible) in which the password is used to generate a key for the one-way function and in which a fixed-length output is produced.

- Access control: Access to the password file is limited to one or a very few accounts.

If one or both of these countermeasures are in place, some effort is needed for a potential intruder to learn passwords. On the basis of a survey of the literature and interviews with a number of password crackers, the following techniques for learning passwords are reported:

1. Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.
2. Exhaustively try all short passwords (those of one to three characters).
3. Try words in the system's online dictionary or a list of likely passwords. Examples of the latter are readily available on hacker bulletin boards.
4. Collect information about users, such as their full names, the names of their spouse and children, pictures in their office, and books in their office that are related to hobbies.
5. Try users' phone numbers, Social Security numbers, and room numbers.
6. Try all legitimate license plate numbers for this state.
7. Tap the line between a remote user and the host system.

Intrusion Detection

Inevitably, the best intrusion prevention system will fail. A system's second line of defense is intrusion detection, and this has been the focus of much research in recent years. This interest is motivated by a number of considerations, including the following:

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised. Even if the detection is not sufficiently timely to preempt the intruder, the sooner that the intrusion is detected, the less the amount of damage and the more quickly that recovery can be achieved.

2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.
3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified. Of course, we cannot expect that there will be a crisp, exact distinction between an attack by an intruder and the normal use of resources by an authorized user. Rather, we must expect that there will be some overlap.

Figure 16 suggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

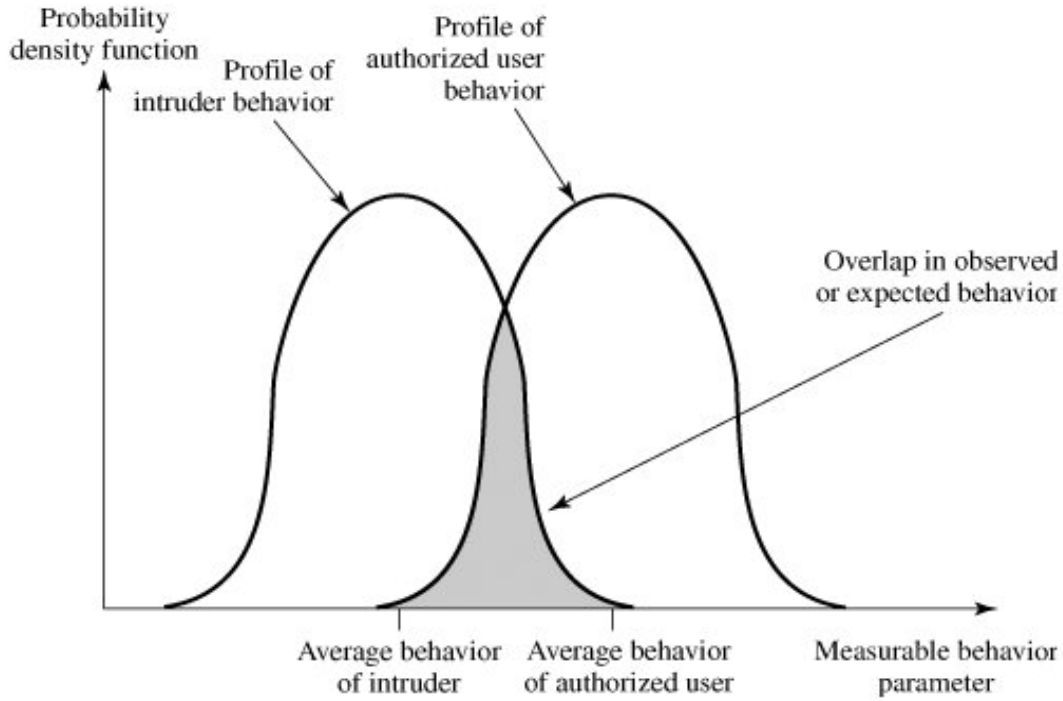


Figure 16. Profiles of Behavior of Intruders and Authorized Users

The following approaches to intrusion detection has been identified:

1. **Statistical anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.
    a. Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
    b. Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.
2. **Rule-based detection**: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.
    a. Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
    b. Penetration identification: An expert system approach that searches for suspicious behavior.

In a nutshell, statistical approaches attempt to define normal, or expected, behavior, whereas rule-based approaches attempt to define proper behavior.

Audit Records

A fundamental tool for intrusion detection is the audit record. Some record of ongoing activity by users must be maintained as input to an intrusion detection system. Basically, two plans are used:

- **Native audit records**: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.
- **Detection-specific audit records**: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system.

Statistical Anomaly Detection

As was mentioned, statistical anomaly detection techniques fall into two broad categories: threshold detection and profile-based systems. Threshold detection involves counting the number of occurrences of a specific event type over an interval of time. If the count surpasses what is considered a reasonable number that one might expect to occur, then intrusion is assumed.

Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined. Because of the variability across users, such thresholds are likely to generate either a lot of false positives or a lot of false negatives. However, simple threshold detectors may be useful in conjunction with more sophisticated techniques.

Profile-based anomaly detection focuses on characterizing the past behavior of individual users or related groups of users and then detecting significant deviations. A profile may consist of a set of parameters, so that deviation on just a single parameter may not be sufficient in itself to signal an alert.

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior. Second, current audit records are the input used to detect intrusion. That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior.

Examples of metrics that are useful for profile-based intrusion detection are the following:

- **Counter:** A nonnegative integer that may be incremented but not decremented until it is reset by management action. Typically, a count of certain event types is kept over a particular period of time. Examples include the number of logins by a single user during an hour, the number of times a given command is executed during a single user session, and the number of password failures during a minute.

- **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.
- **Interval timer:** The length of time between two related events. An example is the length of time between successive logins to an account.
- **Resource utilization:** Quantity of resources consumed during a specified period. Examples include the number of pages printed during a user session and total time consumed by a program execution.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. The following approaches may be taken:

- Mean and standard deviation
- Multivariate
- Markov process
- Time series
- Operational

The simplest statistical test is to measure the **mean and standard deviation** of a parameter over some historical period. This gives a reflection of the average behavior and its variability. The use of mean and standard deviation is applicable to a wide variety of counters, timers, and resource measures. But these measures, by themselves, are typically too crude for intrusion detection purposes.

A **multivariate** model is based on correlations between two or more variables. Intruder behavior may be characterized with greater confidence by considering such correlations (for example, processor time and resource usage, or login frequency and session elapsed time).

A **Markov process** model is used to establish transition probabilities among various states. As an example, this model might be used to look at transitions between certain commands.

A **time series** model focuses on time intervals, looking for sequences of events that happen too rapidly or too slowly. A variety of statistical tests can be applied to characterize abnormal timing.

Finally, an **operational model** is based on a judgment of what is considered abnormal, rather than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits. This type of approach works best where intruder behavior can be deduced from certain types of activities. For example, a large number of login attempts over a short period suggests an attempted intrusion.

Rule-Based Intrusion Detection

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious. In very general terms, we can characterize all approaches as focusing on either anomaly detection or penetration identification, although there is some overlap in these approaches.

Rule-based anomaly detection is similar in terms of its approach and strengths to statistical anomaly detection. With the rule-based approach, historical audit records are analyzed to identify usage patterns and to generate automatically rules that describe those patterns. Rules may represent past behavior patterns of users, programs, privileges, time slots, terminals, and so on. Current behavior is then observed, and each transaction is matched against the set of rules to determine if it conforms to any historically observed pattern of behavior.

As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past. In order for this approach to be effective, a rather large database of rules will be needed.

Rule-based penetration identification takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses. Rules can also be defined that identify suspicious behavior, even when the behavior is within the bounds of established patterns of usage. Typically, the rules used in these systems are specific to the machine and operating system. Also, such rules are generated by "experts" rather than by means of an automated analysis of audit records. The normal procedure is to interview system administrators and security analysts to collect a suite of known penetration scenarios and key events that threaten the security of

the target system. Thus, the strength of the approach depends on the skill of those involved in setting up the rules.

Early system used heuristic rules that can be used to assign degrees of suspicion to activities. Example heuristics are the following:

1. Users should not read files in other users' personal directories.
2. Users must not write other users' files.
3. Users who log in after hours often access the same files they used earlier.
4. Users do not generally open disk devices directly but rely on higher-level operating system utilities.
5. Users should not be logged in more than once to the same system.
6. Users do not make copies of system programs.

Distributed Intrusion Detection

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN or internetwork. Although it is possible to mount a defense by using stand-alone intrusion detection systems on each host, a more effective defense can be achieved by coordination and cooperation among intrusion detection systems across the network.

the following are the major issues in the design of a distributed intrusion detection system:

- A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.
- One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data. Integrity is required to prevent an intruder from masking his or her activities by altering the transmitted audit information. Confidentiality is required because the transmitted audit information could be valuable.
- Either a centralized or decentralized architecture can be used. With a centralized architecture, there is a single central point of

collection and analysis of all audit data. This eases the task of correlating incoming reports but creates a potential bottleneck and single point of failure. With a decentralized architecture, there are more than one analysis centers, but these must coordinate their activities and exchange information.

A good example of a distributed intrusion detection system is one developed at the University of California at Davis. Figure 17 shows the overall architecture, which consists of three main components:

- Host agent module: An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.
- LAN monitor agent module: Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.
- Central manager module: Receives reports from LAN monitor and host agents and processes and correlates these reports to detect intrusion.

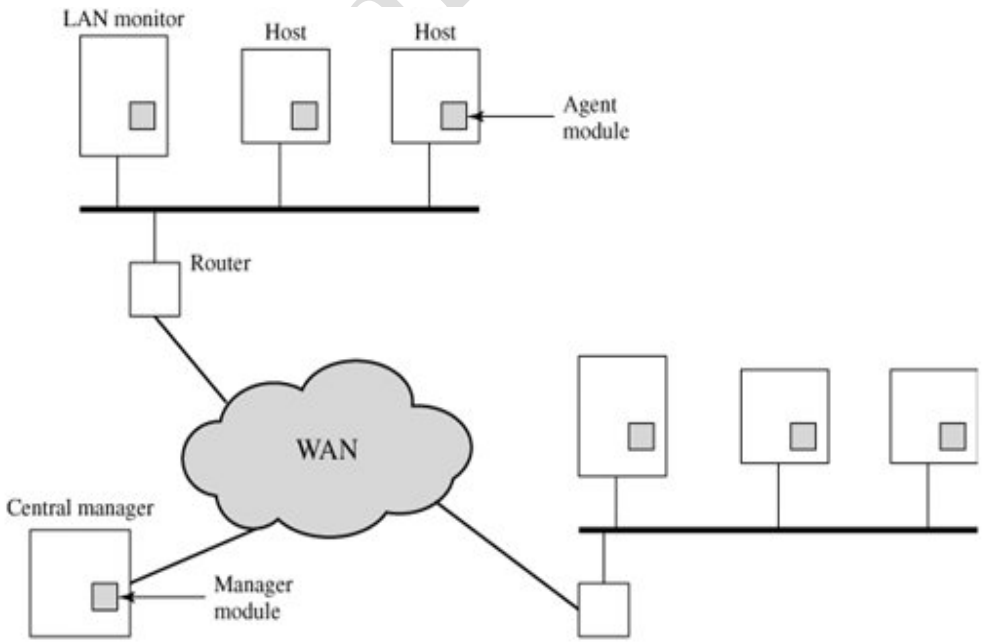


Figure 17. Architecture for Distributed Intrusion Detection

Password Management

Password Protection

The front line of defense against intruders is the password system. Virtually all multiuser systems require that a user provide not only a name or identifier (ID) but also a password. The password serves to authenticate the ID of the individual logging on to the system. In turn, the ID provides security in the following ways:

- The ID determines whether the user is authorized to gain access to a system. In some systems, only those who already have an ID filed on the system are allowed to gain access.
- The ID determines the privileges accorded to the user. A few users may have supervisory or "superuser" status that enables them to read files and perform functions that are especially protected by the operating system. Some systems have guest or anonymous accounts, and users of these accounts have more limited privileges than others.
- The ID is used in what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

Selection Strategies

Many users choose a password that is too short or too easy to guess. At the other extreme, if users are assigned passwords consisting of eight randomly selected printable characters, password cracking is effectively impossible. But it would be almost as impossible for most users to remember their passwords. Fortunately, even if we limit the password universe to strings of characters that are reasonably memorable, the size of the universe is still too large to permit practical cracking. Our goal, then, is to eliminate guessable passwords while allowing the user to select a password that is memorable. Four basic techniques are in use:

- User education
- Computer-generated passwords
- Reactive password checking
- Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations,

particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines. Others may not be good judges of what is a strong password. For example, many users (mistakenly) believe that reversing a word or capitalizing the last letter makes a password unguessable.

Computer-generated passwords also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down. In general, computer-generated password schemes have a history of poor acceptance by users.

A **reactive password checking** strategy is one in which the system periodically runs its own password cracker to find guessable passwords. The system cancels any passwords that are guessed and notifies the user. This tactic has a number of drawbacks. First, it is resource intensive if the job is done right. Because a determined opponent who is able to steal a password file can devote full CPU time to the task for hours or even days, an effective reactive password checker is at a distinct disadvantage. Furthermore, any existing passwords remain vulnerable until the reactive password checker finds them.

The most promising approach to improved password security is a **proactive password checker**. In this scheme, a user is allowed to select his or her own password. However, at the time of selection, the system checks to see if the password is allowable and, if not, rejects it. Such checkers are based on the philosophy that, with sufficient guidance from the system, users can select memorable passwords from a fairly large password space that are not likely to be guessed in a dictionary attack.

The first approach is a simple system for rule enforcement. For example, the following rules could be enforced:

- All passwords must be at least eight characters long.
- In the first eight characters, the passwords must include at least one each of uppercase, lowercase, numeric digits, and punctuation marks.

These rules could be coupled with advice to the user. Although this approach is superior to simply educating users, it may not be sufficient to thwart password crackers. This scheme alerts crackers as to which passwords not to try but may still make it possible to do password cracking.

Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list. There are two problems with this approach:

- Space: The dictionary must be very large to be effective. For example, the dictionary used in the Purdue study occupies more than 30 megabytes of storage.
- Time: The time required to search a large dictionary may itself be large. In addition, to check for likely permutations of dictionary words, either those words most be included in the dictionary, making it truly huge, or each search must also involve considerable processing.

Message Authentication

Perhaps the most confusing area of network security is that of message authentication and the related topic of digital signatures. Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness.

Authentication Requirements:

In the context of communications across a network, the following attacks can be identified:

1. **Disclosure**: Release of message contents to any person or process not possessing the appropriate cryptographic key.
2. **Traffic analysis**: Discovery of the pattern of traffic between parties. In a connection-oriented application, the frequency and duration of connections could be determined. In either a connection-oriented or connectionless environment, the number and length of messages between parties could be determined.
3. **Masquerade**: Insertion of messages into the network from a fraudulent source. This includes the creation of messages by an opponent that are purported to come from an authorized entity. Also included are fraudulent acknowledgments of message receipt or non receipt by someone other than the message recipient.
4. **Content modification**: Changes to the contents of a message, including insertion, deletion, transposition, and modification.
5. **Sequence modification**: Any modification to a sequence of messages between parties, including insertion, deletion, and reordering.
6. **Timing modification**: Delay or replay of messages. In a connection-oriented application, an entire session or sequence of messages could be a replay of some previous valid session, or individual messages in the sequence could be delayed or replayed. In a connectionless application, an individual message (e.g., datagram) could be delayed or replayed.
7. **Source repudiation**: Denial of transmission of message by source.
8. **Destination repudiation**: Denial of receipt of message by destination.

Authentication Functions

Any message authentication or digital signature mechanism has two levels of functionality. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lower-level function is then used as a primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message.

Types of functions that may be used to produce an authenticator may be grouped into three classes, as follows:

- **Message encryption:** The ciphertext of the entire message serves as its authenticator
- **Message authentication code (MAC)**: A function of the message and a secret key that produces a fixed-length value that serves as the authenticator
- **Hash function:** A function that maps a message of any length into a fixed-length hash value, which serves as the authenticator

Message Encryption

Message encryption by itself can provide a measure of authentication. The analysis differs for symmetric and public-key encryption schemes.

Symmetric Encryption

Consider the straightforward use of symmetric encryption (Figure13a). A message M transmitted from source A to destination B is encrypted using a secret key K shared by A and B. If no other party knows the key, then confidentiality is provided: No other party can recover the plaintext of the message.

In addition, we may say that B is assured that the message was generated by A. Why? The message must have come from A because A is the only other party that possesses K and therefore the only other party with the information necessary to construct ciphertext that can be decrypted with K. Furthermore, if M is recovered, B knows that none of the bits of M have been altered, because an opponent that does not know K would not know how to alter bits in the ciphertext to produce desired changes in the plaintext.

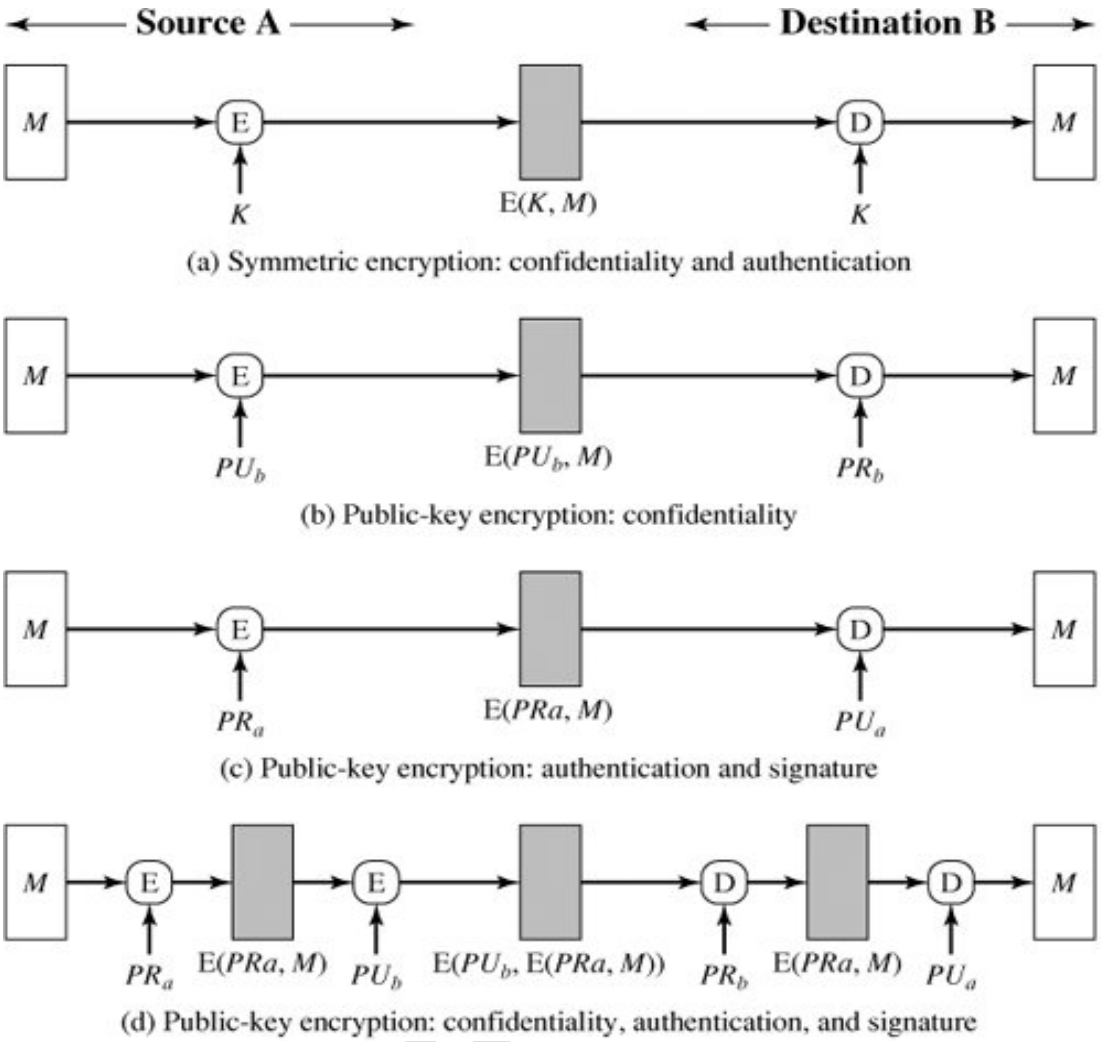


Figure 13. Basic Uses of Message Encryption

So we may say that symmetric encryption provides authentication as well as confidentiality.

Public-Key Encryption

The straightforward use of public-key encryption (Figure 13b) provides confidentiality but not authentication. The source (A) uses the public key PU_b of the destination (B) to encrypt M. Because only B has the corresponding private key PR_b, only B can decrypt the message. This scheme provides no authentication because any opponent could also use B's public key to encrypt a message, claiming to be A.

To provide authentication, A uses its private key to encrypt the message, and B uses A's public key to decrypt (Figure 13c). This provides

authentication using the same type of reasoning as in the symmetric encryption case: The message must have come from A because A is the only party that possesses PR_a and therefore the only party with the information necessary to construct ciphertext that can be decrypted with PU_a. Again, the same reasoning as before applies: There must be some internal structure to the plaintext so that the receiver can distinguish between well-formed plaintext and random bits.

Assuming there is such structure, then the scheme of Figure 13c does provide authentication. It also provides what is known as digital signature.[1] Only A could have constructed the ciphertext because only A possesses PR_a. Not even B, the recipient, could have constructed the ciphertext. Therefore, if B is in possession of the ciphertext, B has the means to prove that the message must have come from A. In effect, A has "signed" the message by using its private key to encrypt. Note that this scheme does not provide confidentiality. Anyone in possession of A's public key can decrypt the ciphertext.

To provide both confidentiality and authentication, A can encrypt M first using its private key, which provides the digital signature, and then using B's public key, which provides confidentiality (Figure 13d). The disadvantage of this approach is that the public-key algorithm, which is complex, must be exercised four times rather than two in each communication.

Message Authentication Code

An alternative authentication technique involves the use of a secret key to generate a small fixed-size block of data, known as a cryptographic checksum or MAC that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key K. When A has a message to send to B, it calculates the MAC as a function of the message and the key:MAC = C(K, M), where

M = input message

C = MAC function

K = shared secret key

MAC = message authentication code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the same secret key, to generate a new MAC. The received MAC is compared to the calculated MAC (Figure 14). If we assume that only the receiver and the sender know the identity of the secret key, and if the received MAC matches the calculated MAC, then

1. The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the MAC, then the receiver's calculation of the MAC will differ from the received MAC. Because the attacker is assumed not to know the secret key, the attacker cannot alter the MAC to correspond to the alterations in the message.
2. The receiver is assured that the message is from the alleged sender. Because no one else knows the secret key, no one else could prepare a message with a proper MAC.
3. If the message includes a sequence number, then the receiver can be assured of the proper sequence because an attacker cannot successfully alter the sequence number.

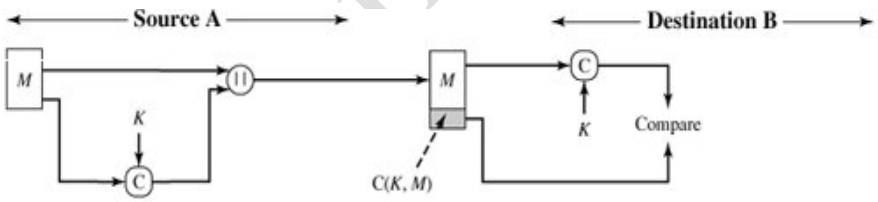


Figure 14. Basic Uses of Message Authentication Code (MAC)

Hash Function

A variation on the message authentication code is the one-way hash function. As with the message authentication code, a hash function accepts a variable-size message M as input and produces a fixed-size output, referred to as a **hash code** H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a **message digest** or hash value. The hash code is a function of all the bits of the message and provides an error-detection capability: A change to any bit or bits in the message results in a change to the hash code.

Figure 15 illustrates a variety of ways in which a hash code can be used to provide message authentication, as follows:

a. The message plus concatenated hash code is encrypted using symmetric encryption. The same line of reasoning applies: Because only A and B share the secret key, the message must have come from A and has not been altered. The hash code provides the structure or redundancy required to achieve authentication. Because encryption is applied to the entire message plus hash code, confidentiality is also provided.
b. Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.

   Note that the combination of hashing and encryption results in an overall function that is, in fact, a MAC (Figure 14). That is, E(K, H(M)) is a function of a variable-length message M and a secret key K, and it produces a fixed-size output that is secure against an opponent who does not know the secret key.

c. Only the hash code is encrypted, using public-key encryption and using the sender's private key. As with (b), this provides authentication. It also provides a digital signature, because only the sender could have produced the encrypted hash code. In fact, this is the essence of the digital signature technique.
d. If confidentiality as well as a digital signature is desired, then the message plus the private-key-encrypted hash code can be encrypted using a symmetric secret key. This is a common technique.
e. It is possible to use a hash function but no encryption for message authentication. The technique assumes that the two communicating parties share a common secret value S. A computes the hash value over the concatenation of M and S and appends the resulting hash value to M. Because B possesses S, it can recompute the hash value to verify. Because the secret value itself is not sent, an opponent cannot modify an intercepted message and cannot generate a false message.
f. Confidentiality can be added to the approach of (e) by encrypting the entire message plus the hash code.

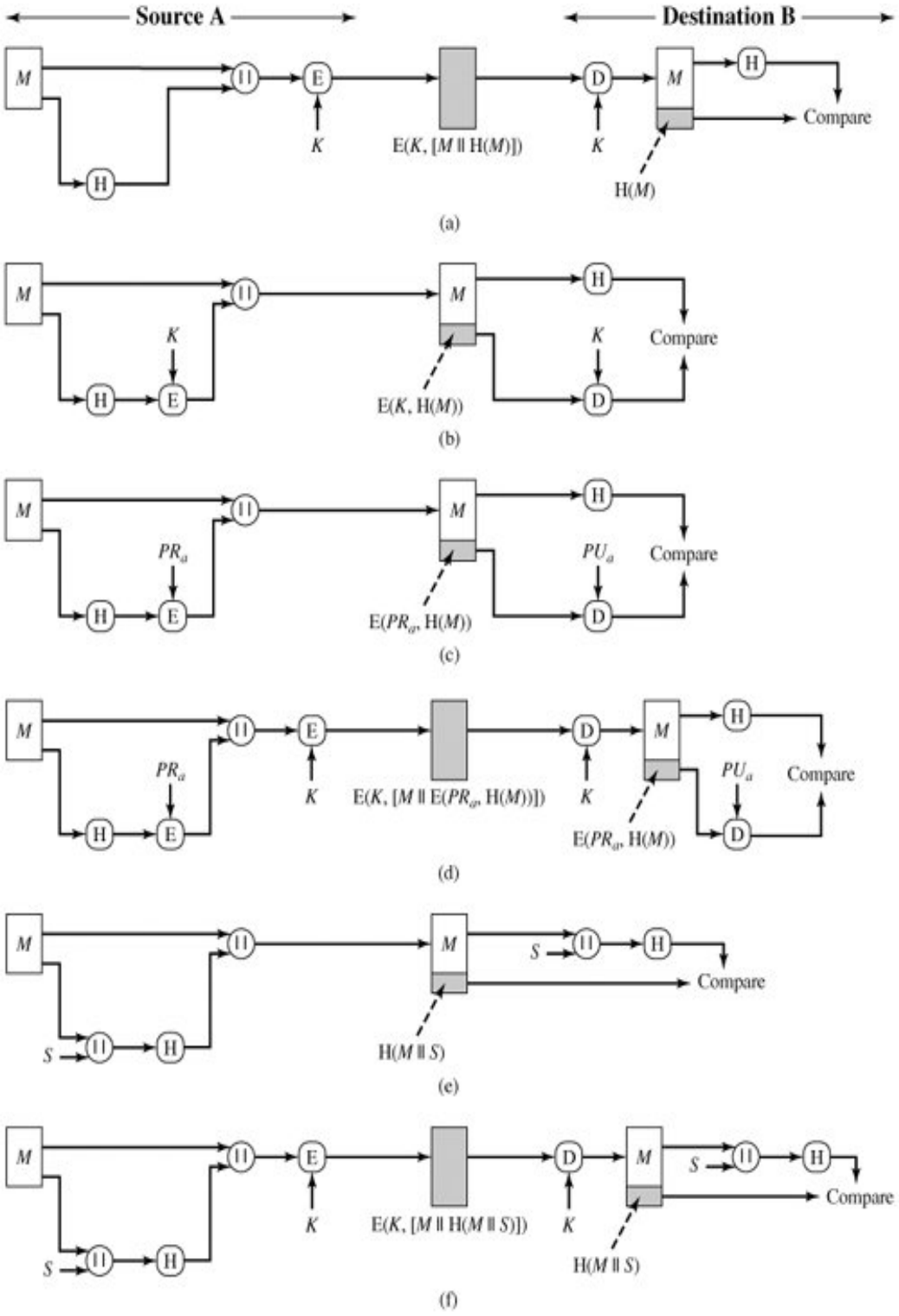


Figure 15. Basic Uses of Hash Function

A hash value h is generated by a function H of the form

h = H(M)

where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (Figure 15).

Requirements for a Hash Function

The purpose of a hash function is to produce a "fingerprint" of a file, message, or other block of data. To be useful for message authentication, a hash function H must have the following properties:

1. H can be applied to a block of data of any size.
2. H produces a fixed-length output.
3. H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.
4. For any given value h, it is computationally infeasible to find x such that H(x) = h. This is sometimes referred to in the literature as the one-way property.
5. For any given block x, it is computationally infeasible to find y x such that H(y) = H(x). This is sometimes referred to as **weak collision resistance**.
6. It is computationally infeasible to find any pair (x, y) such that H(x) = H(y). This is sometimes referred to as **strong collision resistance**.

Simple Hash Functions

All hash functions operate using the following general principles. The input (message, file, etc.) is viewed as a sequence of n-bit blocks. The input is processed one block at a time in an iterative fashion to produce an n-bit hash function.

One of the simplest hash functions is the bit-by-bit exclusive-OR (XOR) of every block. This can be expressed as follows:

$$C_i = b_{i1} \oplus b_{i1} \oplus \ldots \oplus b_{im}$$

where

C_i = ith bit of the hash code, $1 \leq i \leq n$

m = number of n-bit blocks in the input

b_{ij} = ith bit in jth block

$\oplus$ = XOR operation

This operation produces a simple parity for each bit position and is known as a longitudinal redundancy check. It is reasonably effective for random data as a data integrity check.

What Are Computer Viruses?

Computer viruses are small software programs that are designed to spread from one computer to another and to interfere with computer operation.

This definition is rather broad, because it contains everything from small viruses that duplicate your files and are just a mere annoyance to the dangerous ones that put a lock on your files and refuse to give you access to them until you pay a certain amount of money to its creator.

Virus attaches itself to files stored on floppy disks, USBs, email attachments and hard disks. A file containing a virus is called infected file. If this file is copied to a computer, virus is also copied to the computer.

Besides stealing information, computer viruses oftentimes:

- *delete information off of your computer,*
- *corrupt files and make your computer act wonky and weird,*
- *use your e-mail address to spread itself to other users,*
- *take your files hostage until you pay a certain amount of money to the creator of the virus in order to release them.*

In short: Computer viruses are small software programs created for malicious purposes that involve stealing information. They can easily hide themselves into small files such as images and attachments and infect your computer, after which they can multiply and send themselves as attachments using your e-mail address to other people in your list. The damages each virus can do to your computer varies according to the category it fits in. While some may corrupt your files, multiply them or delete them, others go as far as deleting everything from your hard disk or taking your data hostage until you pay a certain fee.

Activation of Viruses

When the computer virus starts working, it is called the activation of virus. A virus normally runs all the time in the computer. Different viruses are activated in different ways. Many viruses are activated on a certain date. For example, a popular virus "Friday, the 13th" is activated only if the date is 13 and the day is Friday.

Causes of Computer Viruses:

The following are the main causes of a Computer Virus.

Infected Flash Drives or Disks

Flash drives and disks are the main cause of spreading viruses. Flash drives and disks are used to transfer data from one computer to other. A virus can also be copied from one computer to other when the user copies infected files using flash drives and disks.

Email Attachments

Most of the viruses spread through emails. Email attachment is a file that is sent along with an email. An email may contain an infected file attachment. Virus can spread if the users opens and downloads an email attachment. It may harm the computer when it is activated. It may destroy files on the hard disk or may send the virus automatically to all email addresses saved in the address book.

Infected websites

Thousands of insecure websites can infect computer with viruses. Most of the websites with suspicion materials are infected, so by visiting these websites the user's computer also gets infected by virus. These websites are developed to spread viruses. The virus is transferred to the user's computer when this material is downloaded. These websites may access the computer automatically when the users visit them.

Networks

Virus can spread if an infected computer is connected to a network. The internet is an example of such network. When a user downloads a file infected with virus from the internet, the virus is copied to the computer. It may infect the files stored on the computer.

Pirated Software

An illegal copy of software is called pirated software. Virus can spread if user installs pirated software that contains a virus. A variety of pirated software is available in CDs and from the internet. Some companies intentionally add virus in the software. The virus is automatically activated if the user uses the software without purchasing license.

Types of computer viruses

Basic types of viruses:

Boot viruses: Boot viruses attack the boot sectors on your hard drive and interfere with your computer's basic operation, making your operating system run strangely or even corrupt it all together .

Macro viruses: Macro viruses tend to attack data files, like word documents and spreadsheets, causing you to loose files or cause your word or excel software to not work properly .

Trojan viruses: Trojan viruses pretend to be other software, hence their name as in the Trojan horse. Trojan viruses pretend to be a legitimate piece of software, but in reality can attack your hard drives, deleting files and re-writing system files, causing your computer to become unstable, particular when operating system files are deleted .

As a general rule, computer viruses only attack files in your computer. They do not attack your computer's hardware, like the monitor, mouse or keyboard .

However, some viruses will attack the files that operate your computer's hardware, causing hard drives to reformat, video drivers to be deleted or your operating system to stop running. While this may cause your monitor to stop working properly, it doesn't mean you need to get a new monitor .

Structure of a Virus

A computer virus has three parts:

Infection mechanism: How a virus spreads, by modifying other code to contain a (possibly altered) copy of the virus. The exact means through which a virus spreads is referred to as its infection vector. This doesn't have to be unique - a virus that infects in multiple ways is called multipartite.
Trigger: The means of deciding whether to deliver the payload or not.
Payload: What the virus does, besides spread. The payload may involve damage, either intentional or accidental. Accidental damage may result from bugs in the virus, encountering an unknown type of system, or perhaps
unanticipated multiple viral infections.

In pseudocode, a virus would have the structure below. The trigger function would return a Boolean, whose value would indicate whether or not the trigger conditions were met. The payload could be anything, of course.

```
def virus():
    infect()
    if trigger() is true:
        payload()
```

Infection is done by selecting some target code and infecting it, as shown below. The target code is locally accessible to the machine where the virus runs. Locally accessible targets may include code in shared network directories, though, as these directories are made to appear locally accessible.
Generally, k targets may be infected each time the infection code below is run.
The exact method used to select targets varies, and may be trivial, as in

the case of the boot-sector infectors. The tricky part of select_target is that the virus doesn't want to repeatedly re-infect the same code; that would be a waste of effort, and may reveal the presence of the virus. Select_target has to have some way to detect whether or not some potential target code is already infected, which is a double-edged sword. If the virus can detect itself, then so can anti-virus software. The infect _code routine performs the actual infection by placing some version of the virus' code in the target.

```
def infect():
    repeat k times:
        target = select_target()
        if no target:
            return
        infect_code(target)
```

File Infectors

Operating systems have a notion of files that are executable. A file infector is a virus that infects files which the operating system considers to be executable; Where is the virus placed?

Beginning of a File

Very simple executable file formats like the .COM MS-DOS format would treat the entire file as a combination of code and data. When executed, the entire file would be loaded into memory, and execution would start by jumping to the beginning of the loaded file. In this case, a virus that places itself at the start of the file gets control first when the infected file is run, as illustrated in Figure 1. This is called a prepending virus.

End of a File

Appending code onto the end of a file is extremely easy. A virus that places itself at the end of a file is called an appending virus. How does the virus get control? There are two basic possibilities:
• The original instruction(s) in the code can be saved, and replaced by a jump to the viral code. Later, the virus will transfer control back to the

code it infected. The virus may try to run the original instructions directly in their saved location, or the virus may restore the infected code back to its original state and run it.

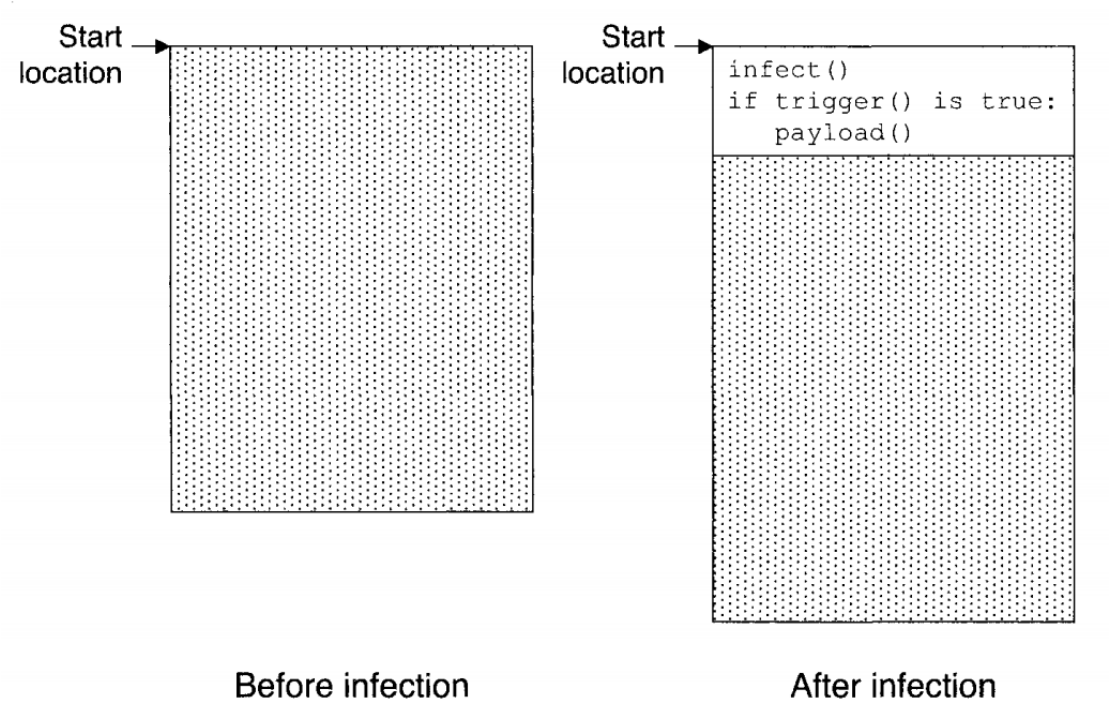


Figure1. Prepending Virus

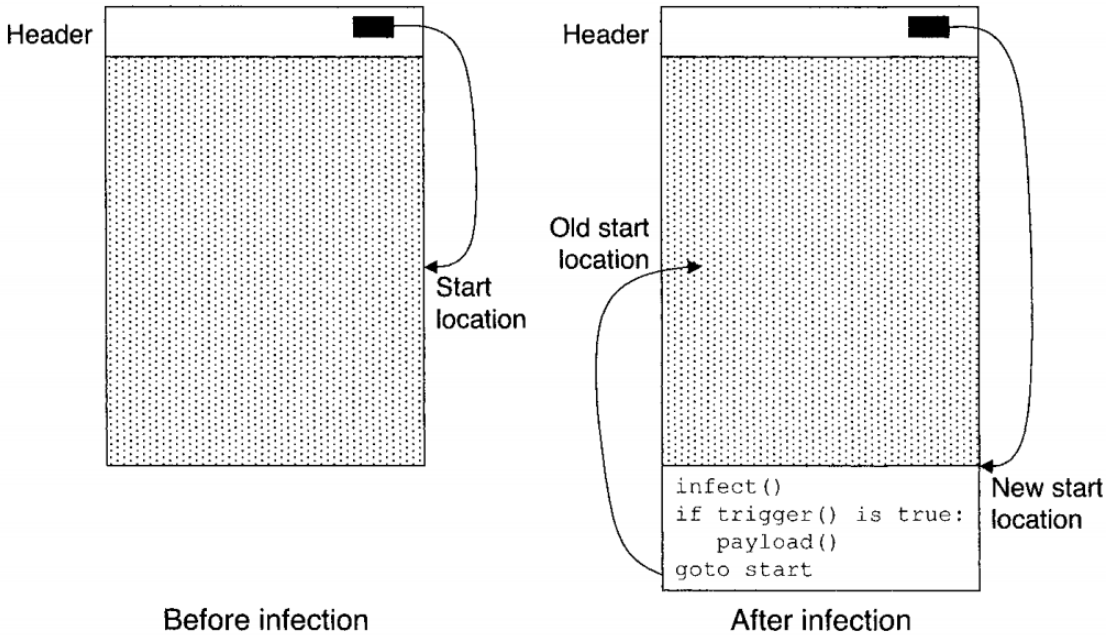


Figure2. Appending Virus

• Many executable file formats specify the start location in a file header. The virus can change this start location to point to its own code, then jump to the original start location when done. Figure 2 shows an appending virus using the latter scheme.

Well known Computer Viruses:

The following are some well-known viruses:

CodeRed

It is a worm that infects a computer running Microsoft IIS server. This virus launched DOS attack on White House's website. It allows the hacker to access the infected computer remotely.

Nimba

It is spreads itself using different methods. IT damages computer in different ways. It modified files, alters security settings and degrades performance.

SirCam

It is distributed as an email attachment. It may delete files, degrade performance and send the files to anyone.

Melisa

It is a virus that is distributed as an email attachment. IT disables different safeguards in MS Word. It sends itself to 50 people if Microsoft Outlook is installed.

Ripper. It corrupts data from the hard disk.

MDMA. It is transferred from one MS Word file to other if both files are in memory.

Concept

It is also transferred as an email attachment. It saves the file in template directory instead of its original location.

One_Half

It encrypts hard disk so only the virus may read the data. It displays One_Half on the screen when the encryption is half completed.

Anti-virus programs:

The broad definition for anti-virus programs is that they are one or multiple programs that have been designed with the specific purpose of preventing and destroying the malicious software (also known as the computer viruses) they detect on the user's computer. These programs are very smart and they use all sorts of tricks in order to catch and destroy the malicious software on the user's computer. They go as far as creating 'bait files' that the viruses will attack and thus get destroyed or
quarantined.

Things to remember when purchasing an anti-virus program:

- **Make sure it's a full paid version** – the free ones just search your computer for patterns of malicious software they are already familiar with and they don't do anything about the viruses they find on your computer. The full and paid version of the anti-virus program will use a variety of techniques to catch, destroy or quarantine the viruses on your computer.
- **Make sure you have an Internet connection** –the anti-virus programs need a constant internet connection in order to stay up to date on the virus definitions and make sure that they catch all of the malicious software / programs that can be found in your computer.
- **Don't download anything that you don't trust to be virus free**. No matter how pretty that wallpaper looks like or how much you want that new song, unless you know and trust the website, don't ever download anything off of the internet. Oftentimes, those that create such malicious software hide it in plain sight – in .mp3 or .jpg files and once it reaches your computer, it's very difficult to get rid of it.
- **Do some research before purchasing a certain anti-virus program**. As you probably know by now, the competition is fierce

in the anti-virus industry, do some research, see what other people experienced with the same program and then decide whether to buy it or not.

How do anti-virus programs work?

The science behind anti-virus programs is not that difficult to understand. As mentioned previously, they were created with the sole purpose of stopping the malicious software from damaging the computers of the users. They employ a variety of techniques in order to detect, quarantine and eventually destroy the viruses, some of which include:

- **Scanning the downloaded files;** This means that the anti-virus program starts to scan the documents you are downloading into your computer as soon as they appear in your computer or as soon as you click the "Download" button on the website you want to download the said file from.
- **Scanning the programs before you execute them;** When you double click a program in order to start it up, even if you don't notice it because it happens so fast, the anti-virus program on your computer will scan it very quick in order to make sure that there is no malicious software attached to it and that you can safely open it. If it is, then the anti-virus program will stop the program from starting up in order to protect your computer.
- **Scanning the entire computer;** If you tell your anti-virus program to scan your entire computer, it will take each file, one by one and run it through a series of programs in order to determine whether the said file contains malware or malicious software. This type of scanning takes much longer than usual obviously because the amount of files the anti-virus must go through is huge.

DIGITAL SIGNATURE

Properties

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two parties are possible.

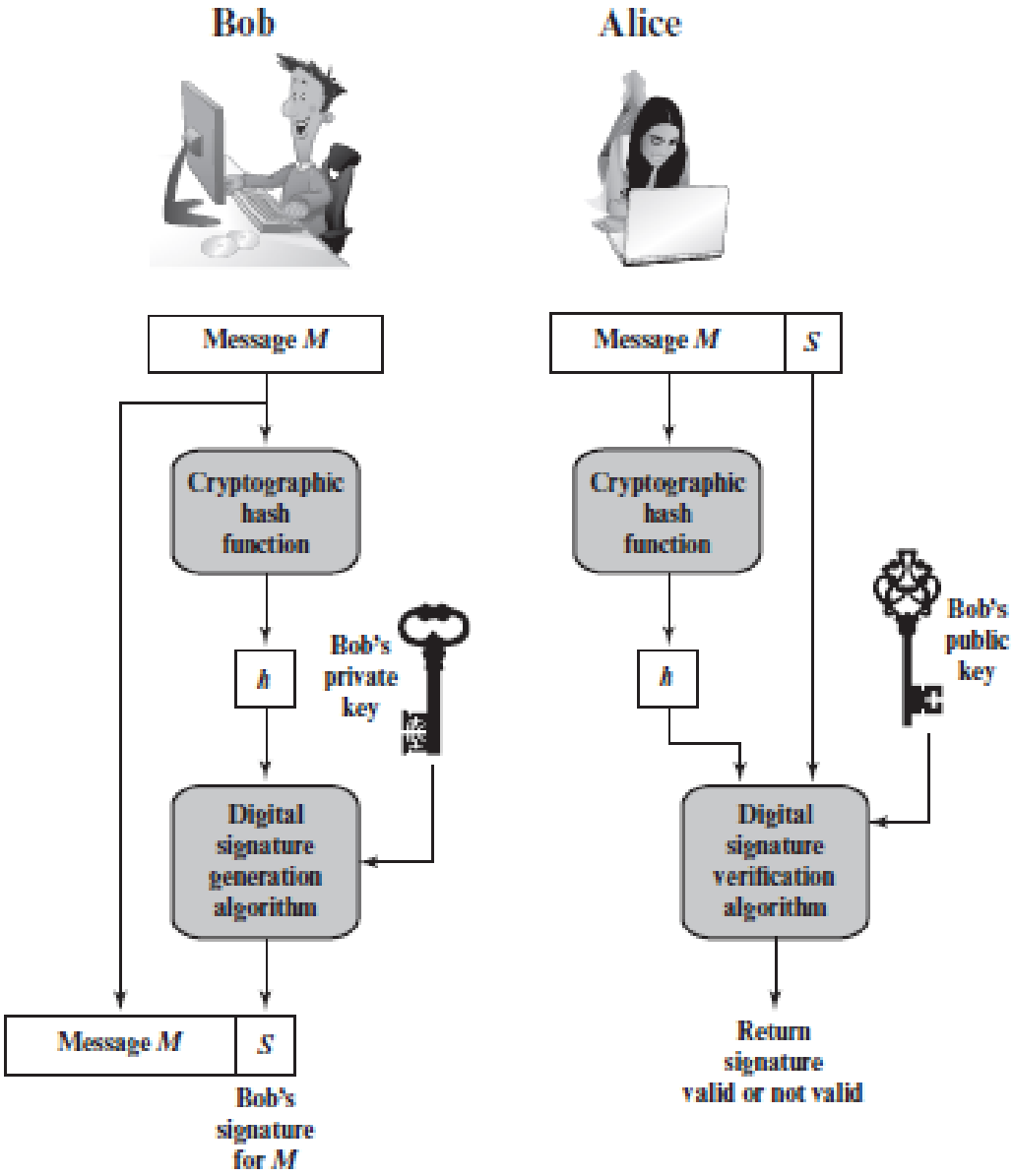
For example, suppose that John sends an authenticated message to Mary. Consider the following disputes that could arise.

1. Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
2. John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message.

Both scenarios are of legitimate concern. Here is an example of the first scenario: An electronic funds transfer takes place, and the receiver increases the amount of funds transferred and claims that the larger amount had arrived from the sender. An example of the second scenario is that an electronic mail message contains instructions to a stockbroker for a transaction that subsequently turns out badly. The sender pretends that the message was never sent.
In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature must have the following properties:

- It must verify the author and the date and time of the signature.
- It must authenticate the contents at the time of the signature.
- It must be verifiable by third parties, to resolve disputes.

Thus, the digital signature function includes the authentication function.


Bob
Alice
Message M
Cryptographic hash function
h
Bob's private key
Digital signature generation algorithm
Message M
S
Bob's signature for M
(a) Bob signs a message
Message M
S
Cryptographic hash function
h
Bob's public key
Digital signature verification algorithm
Return signature valid or not valid
(b) Alice verifies the signature

Attacks and Forgeries

Following are types of attacks. Here A denotes the user whose signature method is being attacked, and C denotes the attacker.

- Key-only attack: C only knows A's public key.
- Known message attack: C is given access to a set of messages and their signatures.
- Generic chosen message attack: C chooses a list of messages before attempting to breaks A's signature scheme, independent of A's public key. C then obtains from A valid signatures for the chosen messages. The attack is generic, because it does not depend on A's public key; the same attack is used against everyone.
- Directed chosen message attack: Similar to the generic attack, except that the list of messages to be signed is chosen after C knows A's public key but before any signatures are seen.
- Adaptive chosen message attack: C is allowed to use A as an "oracle." This means that C may request from A signatures of messages that depend on previously obtained message-signature pairs.

Digital signature Requirements

On the basis of the properties and attacks just discussed, we can formulate the following requirements for a digital signature.

- The signature must be a bit pattern that depends on the message being signed.
- The signature must use some information only known to the sender to prevent both forgery and denial.
- It must be relatively easy to produce the digital signature.
- It must be relatively easy to recognize and verify the digital signature.
- It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
- It must be practical to retain a copy of the digital signature in storage.

A secure hash function provides a basis for satisfying these requirements.

Direct Digital Signature

The term direct digital signature refers to a digital signature scheme that involves only the communicating parties (source, destination). It is assumed that the destination knows the public key of the source.

Confidentiality can be provided by encrypting the entire message plus signature with a shared secret key (symmetric encryption). Note that it is important to perform the signature function first and then an outer confidentiality function.

In case of dispute, some third party must view the message and its signature. If the signature is calculated on an encrypted message, then the third party also needs access to the decryption key to read the original message. However, if the signature is the inner operation, then the recipient can store the plaintext message and its signature for later use in dispute resolution.

The validity of the scheme just described depends on the security of the sender's private key. If a sender later wishes to deny sending a particular message, the sender can claim that the private key was lost or stolen and that someone else forged his or her signature. Administrative controls relating to the security of private keys can be employed to thwart or at least weaken this ploy, but the threat is still there, at least to some degree. One example is to require every signed message to include a timestamp (date and time) and to require prompt reporting of compromised keys to a central authority.

Another threat is that a private key might actually be stolen from X at time T.

The opponent can then send a message signed with X's signature and stamped with

a time before or equal to T.

The universally accepted technique for dealing with these threats is the use of a digital certificate (digital signature algorithm).

Schnoor Digital Signature Scheme:

The scheme is based on using a prime modulus p. Typically, we use $p \approx 2^{1024}$ and $q \approx 2^{160}$. Thus, p is a 1024-bit number, and q is a 160-bit number, which is also the length of the SHA-1 hash value.

The first part of this scheme is the generation of a private/public key pair, which consists of the following steps.

1. Choose primes p and q, such that q is a prime factor of p - 1.
2. Choose an integer a, such that $a^q = 1 \bmod p$. The values a, p, and q comprise a global public key that can be common to a group of users.
3. Choose a random integer s with $0 < s < q$. This is the user's private key.
4. Calculate $v = a^{-s} \bmod p$. This is the user's public key.

A user with private key s and public key v generates a signature as follows.

1. Choose a random integer r with $0 < r < q$ and compute $x = a^r \bmod p$. This computation is a preprocessing stage independent of the message M to be signed.
2. Concatenate the message with x and hash the result to compute the value e:

$e = H(M \| x)$

3. Compute $y = (r + se) \bmod q$. The signature consists of the pair (e, y).

Any other user can verify the signature as follows.

1. Compute $x' = a^y v^e \bmod p$.
2. Verify that $e = H(M \| x')$.

To see that the verification works, observe that

$x' \equiv a^y v^e \equiv a^y a^{-se} \equiv a^{y-se} \equiv a^r \equiv x \pmod p$

Hence, $H(M \| x') = H(M \| x)$.