Unit 9. Cibersecurity and Computer security

Contenido

1 Introducction to security

In the broadest sense security can be defined as the protection of assets. There are three main aspects to security:

- ✓ prevention
- ✓ detection
- ✓ reaction.

Consider security in the traditional sense – for example, securing your house against burglary. You may take steps to **prevent** a burglary such as locking the doors and windows and installing a burglar alarm. If a burglary did occur, you would be able to **detect** this because items would be missing and the burglar may have caused damage to your house while breaking in. You might **react** to the burglary by reporting it to the police, working out what had been stolen and making an insurance claim.

Another term for computer or system security is cybersecurity. Cyber security is the practice of defending computers, servers, mobile devices, electronic systems, networks, and data from malicious attacks. It's also known as information technology security or electronic information security. The term applies in a variety of contexts, from business to mobile computing, and can be divided into a few common categories.

- **Network security** is the practice of securing a computer network from intruders, whether targeted attackers or opportunistic malware.
- **Application security** focuses on keeping software and devices free of threats. A compromised application could provide access to the data its designed to protect. Successful security begins in the design stage, well before a program or device is deployed.
- **Information security** protects the integrity and privacy of data, both in storage and in transit.
- **Operational security** includes the processes and decisions for handling and protecting data assets. The permissions users have when accessing a network and the procedures that determine how and where data may be stored or shared all fall under this umbrella.
- **Disaster recovery and business continuity** define how an organization responds to a cyber-security incident or any other event that causes the loss of operations or data. Disaster recovery policies dictate how the organization restores its operations and information to return to the same operating capacity as before the event. Business continuity is the plan the organization falls back on while trying to operate without certain resources.
- **End-user education** addresses the most unpredictable cyber-security factor: people. Anyone can accidentally introduce a virus to an otherwise secure system by failing to follow good security practices.

Teaching users to delete suspicious email attachments, not plug in unidentified USB drives, and various other important lessons is vital for the security of any organization.

1.1 *How is information security different?*

Although the definition of security given above still applies when we are talking about information, there are some major differences between traditional security and information security.

Information can be stolen – but you still have it.

If a physical item such as a car is stolen then the thief has possession of the car and you no longer have it. If a thief steals a file from your computer, he will probably make a copy of the file for himself and leave the original on your computer. Hence you still have the file but it has also been stolen.

Confidential information may be copied and sold – but the theft might not be detected.

If your car has been stolen it is not hard to detect the fact – the car is missing! However as mentioned above, a thief who steals computer files may leave the files on your computer and only copy them for himself. Nothing appears to have changed on your computer so you may not be aware that anything untoward has happened.

The criminal may be on the other side of the world.

If a thief steals your car you at least know where he was when he stole the car. However, it is possible to hack into computer systems remotely from anywhere in the world. This makes it very hard to know who is responsible for catching a computer criminal. Is it the police in the country where the computer is, or the police in the country where the criminal is?

Although there is no single definition of computer security, we can say that:

Computer security deals with the prevention and detection of unauthorised actions by users of a computer system.
This subject deals with the theory of computer security. You should be aware that unfortunately things that are great in theory do not always work in practice. As Schneier says in *Secrets and Lies*:

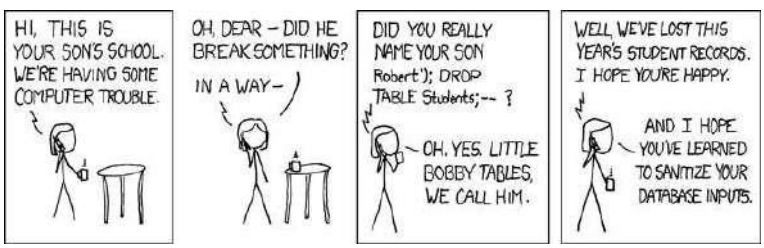
Theory works best in ideal conditions and laboratory settings. We can design idealised operating systems that are provably secure, but we can't actually build them to work securely in the real world. The real world involves design

trade-offs, unseen variables and imperfect implementations.

Schneier kept a log of 'security events' for the first week of March 2000. He recorded approximately 100 events during this time including hackers launching denial-of-service attacks, leakage of personal data from supposedly secure websites, email worms and viruses, and websites being defaced. Most of these attacks and vulnerabilities were the result of the perpetrator bypassing the security mechanism, or exploiting a weakness in the system such as an overflowing buffer.

1.2 *Learning activity*

The following cartoon by Randall Munroe is taken from xkcd.com.



Do an Internet search on SQL injection vulnerabilities to find out why this is funny!

1.3 *Features of a security system*

In order to prevent and detect unauthorised actions by its users a good security system should provide (some of) the following features:

- ✓ confidentiality
- ✓ integrity
- ✓ availability
- ✓ non-repudiation
- ✓ authentication
- ✓ access controls
- ✓ accountability.

We will look at each of these features in turn. Note that different authors on computer security disagree as to which of these features are the most important. It will depend on the main purpose of the system – is confidentiality

paramount or is the prevention of denial-of-service attacks more important? This will depend on the system in question. For example, a computer system which holds personal medical records must certainly provide access controls in order to ensure that personal information does not fall into the wrong hands, and integrity to ensure that the information stored is accurate. Other features such as non-repudiation and availability may not be so important in this case. On the other hand, it is essential for a computer system which transfers money electronically to guarantee non-repudiation and accountability in order to prevent and/or detect dishonest transactions occurring.

In this context, the term *unauthorised* implies not only malicious or criminal, but could also be accidental. For example, a breach of confidentiality arises maliciously if a spy deliberately hacks into a computer and looks at confidential material stored there. It happens accidentally if the material is left out on a desk and is seen by the office cleaner.

1.3.1 Confidentiality

Confidentiality is the prevention of unauthorised disclosure of information.
In other words, confidentiality means keeping information private or safe.
Confidentiality may be important for military, business or personal reasons. Confidentiality may also be known as *privacy* or *secrecy*.

1.3.2 Integrity

Integrity is the prevention of unauthorised writing or modification of information.
Integrity in a computer system means that there is an external consistency in the system – everything is as it is expected to be. *Data integrity* means that the data stored on the computer is the same as what is intended.

1.3.3 Availability

Availability is the prevention of unauthorised with-holding of information.
Information should be accessible and usable upon appropriate demand by an authorised user. *Denial of service* attacks are a common form of attack against computer systems whereby authorised users are denied access to the computer system. Such an attack may be orchestrated by the attacker flooding the system with requests until it cannot keep up and crashes. Authorised users are unable to access the system. Consider the damage that such an attack may cause to an electronic commerce site such as an internet shop.

1.3.4 Non-repudiation

Non-repudiation is the prevention of either the sender or the receiver denying a transmitted message.

A computer security system must be able to prove that certain messages were sent and received, who sent the message, who received the message and perhaps what the message said. For example, suppose a dishonest trader sends an electronic message to a stock broker telling him to buy £2,000 worth of shares in CryptoCom. The next day the price of CryptoCom shares soars. The trader now pretends that his original message said to buy £20,000 worth of shares. Conversely if the share price fell he might pretend that the original message said to buy shares in KryptoCom instead. Non-repudiation means that the trader is not able to deny his original message.

Non-repudiation is often implemented by using *digital signatures*

1.3.5 Authentication

Authentication is proving a claim – usually that you are who you say you are, where you say you are, at the time that you say it is.
Authentication may be obtained by the provision of a password or by a scan of your retina for example. See Chapter 2 for further methods of authentication.

1.3.6 Access controls

Access controls provide the limitation and control of access to authorised users through identification and authentication.
A system needs to be able to identify and authenticate users for access to data, applications and hardware. In a large system there may be a complex structure determining which users and applications have access to which objects. See Chapter 3 for further details on access control models.

1.3.7 Accountability

Accountability means that the system is able to provide audit trails of all transactions.
The system managers are accountable to scrutiny from outside the system and must be able to provide details of all transactions that have occurred. Audit trails must be selectively kept (and protected to maintain their integrity) so that actions affecting security can be traced back to the responsible party.

1.4 *Learning activity*

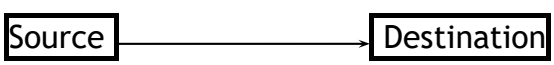
Consider the following scenario and think about the questions at the end.

A student suspects there is a vulnerability on a system in a university public access laboratory. She tests this by trying to exploit the vulnerability. She succeeds, and obtains privileges that she would not normally have. She reports both the hole and her exploiting it to the system staff, who in turn report it to the manager of the laboratory. The manager files charges of breaking into the computing system against the student. The student has to appear before the Student Judicial Authority – she is in trouble!

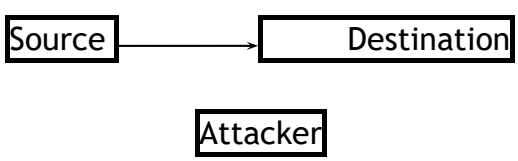
1. Did the student act ethically by testing the system for the security hole before reporting it?
2. Did the manager act ethically by filing charges against the student?
3. The manager told the system staff not to bother fixing the hole, because the action taken by the SJAwould deter any further break-ins through the hole. Was the manager's action appropriate?

1.5 *Security attacks*

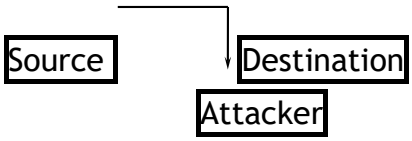
There are a **number of ways** in which an attacker can disrupt communications. Normally, information goes from the source to the destination.



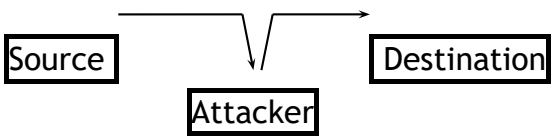
Communication is ***interrupted*** if the attacker does not allow the information to reach the destination.



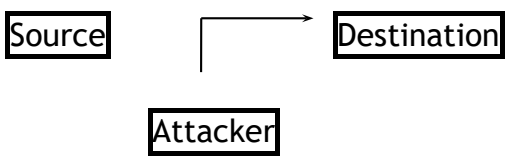
Communication is ***intercepted*** if the attacker interrupts the communication and receives the source information.



Modification occurs when the attacker intercepts the communication, alters it in some way, and then sends it on to the destination. The attacker intends to deceive the destination into thinking that the modified communication has come directly from the source. This is also known as a *Man-in-the-middle attack*.



An attacker may also make up a communication and send it to the destination pretending that it has come from the source. This is called ***fabrication***.



1.6 *Security systems*

A computer security system is not just a computer package. It also requires security conscious personnel who respect the procedures and their role in the system. For example, an access control system may be rendered worthless by employee Fred Smith who chooses user-name *Fred* and password *Smith* and therefore leaves the system open to abuse by password hackers (see section 2.3.1). Conversely, a good security system should not rely on personnel having security expertise.
Security systems

1.6.1 Risk analysis

When designing or implementing a computer security system it is very important to bear in mind the level of risk involved and the value of the information that is to be protected. As an illustration, consider that you may be willing to leave £50 in a changing room locker, but you would not be likely to leave £5,000 unattended. You would assess the risk involved before deciding whether to leave the money or not. On the other hand, it would be foolish to pay someone, say £20, to look after your £50, but this might be a good investment in the case of the £5,000 (assuming that you totally trust the person charged with keeping your money safe of course!).

In terms of computer security, the disadvantages of security systems are that they are time consuming, costly, often clumsy, and impede management and the smooth running of the system. *Risk analysis* is the study of the cost of a particular system (in terms of effort and time as well as cost) against the benefits of the system (the level of security offered).

1.7 *Learning activity*

Think about your own circumstances and where you might be affected by computer security. For example, do you use a password or PIN for any purpose? Are there medical or employment records about you? What features of a security system are involved with each example?
Consider a computer system that you are familiar with; for example, perhaps you have a networked system where you work or study, or a PC at home that is used by more than one person. How good is the security of the system? How easy is it to access other people's files or to read their emails? How difficult would it be to add extra security to the system?

1.7.1 Design considerations

There are a number of questions which need to be considered when designing a security system. We will pose five design questions here. See Gollmann, Chapter 1 for further discussion of these questions.

Does the system focus on the data, operations or users of the system?

For example, is it more important to have a data focused rule such as: *Only data of type A can be inserted in data box A* or a user focused rule such as: *Only section managers are able to access the information in data box A*?

What level should the security system operate from?

The security system may consist of a software package that runs on top of the operating system, such as Norton Internet Security which runs on top of

Windows. Alternatively, it may be part of the hardware and have physical control over the data such as where it is stored and how it is manipulated, for example Security Enhanced Linux (SELinux). ■ Should the security system be simple or sophisticated?

As discussed above, there are disadvantages to having a security system, not least in terms of time and cost. The more sophisticated a system the costlier it is likely to be. On the other hand, a system which is too simple may not provide the necessary level of security. It is obviously not a good idea to spend millions of dollars on a state of the art security system which is to be used to protect data that is not of high importance or value.

In a distributed system should the security be centralised or spread?

Should a security manager have ultimate control, for example over access control issues (this will make it easier to achieve a consistent and rigorous approach, but may cause time delays if the security manager has to be applied to for every change of access rights)? Alternatively, should individual users be allowed to choose who has access to their files? See section 3.3.3 for a description of how SELinux implements mandatory access control.

How do you secure the levels below the level of the security system?

An attacker may manage to gain access to the operating system and from there make alterations to access control limitations giving themselves access to other parts of the system. The logical access controls of the system may be by-passed by gaining direct access to the physical memory. It is therefore important to ensure that physical security measures are in place as well as the logical computer security mechanisms.

1.6 Security models

Computer security protects the computer system and the data it processes. Success depends on the implementation of security controls designed for the system. A *security model* is a means of formally expressing the rules of the security policy. The model should:

- ✓ be easy to comprehend
- ✓ be without ambiguity
- ✓ be possible to implement
- ✓ reflect the policies of the organisation.

Five popular and valuable models are as follows;

- Bell-LaPadula Model.
- Biba Model.
- Clark Wilson Model.
- Brewer and Nash Model.

- Harrison Ruzzo Ullman Model.

9 **Learning activity**

Do an Internet search for some examples of definitions of security concepts. A good starting point is the web site of the UK National Technical Authority for Information Assurance:
http://www.cesg.gov.uk. Other governments have similar sites, and many major IT companies also have pages discussing security.

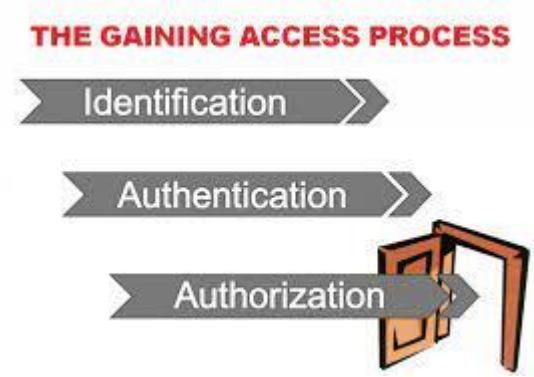
2 Identification and authentication

2.1 *Introduction*

Identification and authentication are commonly used as a two-step process, but they are distinct activities. Identification is the claiming of an identity. This only needs to occur once per authentication or access process. Any one of the three common authentication factors can be employed for identification. Once identification has been performed, the authentication process must take place. Authentication is the act of verifying or proving the claimed identity. The issue is both checking that such identity actually exists within the known accounts of the secured environment and also ensuring that the human claiming the identity is the correct, valid.

In other words, Identification is the ability to identify uniquely a user of a system or an application that is running in the system. *Authentication* is the ability to prove that a user or application is genuinely who that person or what that application claims to be.

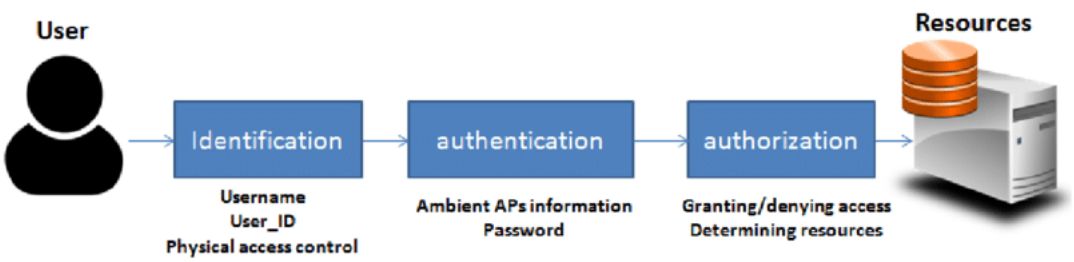
For example, consider a user who logs on to a system by entering a user ID and password. The system uses the user ID to identify the user. The system authenticates the user at the time of logon by checking that the supplied password is correct.



Two other processes complement the user activity in a system since his login until the logout when he ceases to work with the system.

Authorization

A lot of times, many people get confused with authentication and authorization. To many, it seems simple, if I'm authenticated, I'm authorized to do anything. Once the subject provides its credentials and is properly identified, the system it is trying to access needs to determine if this subject has been given the necessary rights and privileges to carry out the requested actions. Consider your mail, where you log in and provide your credentials. You will be able to compose a mail, delete a mail and do certain changes which you are authorized to do. Can you make changes to the messaging server? No, since you are not authorized to do so. Hence successful authentication does not guarantee authorization. Successful authentication only proves that your credentials exist in the system and you have successfully proved the identity you were claiming. However, to make any changes, you need authorization. The system may check these privileges through an access control matrix or a rule-based solution through you would be authorized to make the changes.



Accountability

The final piece in the puzzle is about accountability. Imagine where a user has been given certain privileges to work. What happens when he/she decides to misuse those privileges? If the audit logs are available, then you'll be able to investigate and make the subject who has misused those privileges accountable on the basis of

those logs. The subject needs to be held accountable for the actions taken within a system or domain. The only way to ensure accountability is if the subject is uniquely identified and the subject's actions are recorded. Auditing capabilities ensure users are accountable for their actions, verify that the security policies are enforced, and can be used as investigation tools. Clearly, accountability is the process of taking records of the user activity in the system followed by processing and analysing this information to obtain acountability reports.

2.2 *User-names and Passwords*

When a computer system has to verify a user's identity, there are two basic questions that have to be asked and answered appropriately. The first is:
Who are you?

The computer system has to establish somehow *who* is trying to gain access to its files. This is usually done by use of a *user-name* which, although probably unique to the user, is not a secret. The user-name is often simply produced using all or part of the user's actual name. For example, the user-name of John Smith might be *JSmith* or *johnsmith*.

When John Smith correctly enters his user-name, the computer can establish, by looking in a database of authorised user-names, that John Smith is an authorised user of the system. However, a second question now has to be asked:
How do I know that you are who you say you are?

The computer must now establish that the person logging into the system as

John Smith actually is John Smith. Since the user-name is not a secret, anyone could try to log into the system using the identity of John. The person logging in must somehow prove that they are the genuine John Smith. This is usually done by using a *password*. The password is a secret and is only known to the genuine user John Smith. By entering this secret password, in conjunction with his user-name, John proves to the computer that he is an authorised user and is allowed access to the system.

Thus there are typically two stages in the process of identification.

1) A *user-name* is used to establish identity.
2) A *password* is used to establish authentication of identity.

Your own name is an example of *something you are*. If you apply for a bank loan or a travel visa you will have to prove you are who you say you are by showing, for example, your passport. Your passport is an example of *something you have*. Another example of *something you have* might be a bank card. In order to use the bank card to withdraw money at a cash machine, you do not have to prove who you are, but you do have to prove that you are authorised to use the card.

This is done by using *something you know*. In this case the PIN number associated with the bank card. It is possible (although not recommended) for a person to lend their bank card to someone else. However, the second person will only be able to withdraw money using the bank card if they know the correct PIN number. A password is another example of *something you know*. Thus, a person can identify themselves by using *something they are, something they have*, or *something they know*.

2.3 ***Threats***

A basic identification system consists of a database of passwords indexed by user-names. This is called the *password file*. When a user logs into the system, the computer checks that the user-name and password input match an entry in the password file. If a match is found, the process is complete and the user is allowed access to the system. If not, access is denied although the user may be given another chance to enter their user-name and password.

There are various ways in which a user-name/password identification system can be abused. The simplest attacks include the hacker looking over the user's shoulder when they are typing in their password, or finding a written note that the user has made of their password. In the following sections we will consider some further possible attacks that might be used and defences that can be employed to either prevent, or detect, an attack.

2.3.1 Password guessing

Suppose that a hacker wants to access a system which is protected by a user-name/password identification system. We will assume that the hacker knows the user-name of an authorised user since this information is not generally secret. Therefore if the hacker can guess the user's password he will gain access to the system. There are several ways in which the hacker can find out the user's password. These include:

Guessing using personal knowledge of the user

Many people use passwords which relate to them personally. For example, they may use the name of their spouse or child or pet. They may use their football team or street name or birth date. If the hacker can find out personal information about the user, then they may be able to guess a personal password without too much difficulty.

This attack will fail if the user is careful not to use a password which is personally related to them in any way.

Dictionary searching

Another favourite method of generating easy to remember passwords is for the user to choose a word, usually in their own language. If the hacker cannot directly guess the user's password then he may set up a *dictionary attack*. This means that he will run a computer programme which tries every word in a dictionary as the password of the user until he finds a match.

This attack will fail if the user does not use a word which appears in a dictionary as their password.

Intelligent searching

Some user-name/password systems insist that the user's password contains a mix of letters and numbers. The most common thing for a user (who has not been educated in password security) to do is add a number onto the end of a word. For example, using a password such as banana1. An intelligent dictionary search might try all words with numbers added. Thus if the hacker knows that a particular password system insists that passwords are a minimum of six characters long and must contain at least one number, then the hacker may try all five letter words with each of the digits 0,..,9 attached. Thus apple0, apple1, apple2,....,apple9, apply0, apply1,... and so on would form part of this search.

If this attack does not succeed, the next step might be to capitalise the first letter of each word in the dictionary. Other intelligent dictionary modifications include capitalising each letter of the word in turn, including a number at the

front of the word, including a number in any position in the word or replacing letters which are similar to numbers with that number. For example, replacing the letter l with the number 1 or the letter o with the number 0.

Exhaustive searching

If the user has been clever enough to use a random, meaningless string of characters as their password, then the hacker may have to resort to trying an *exhaustive search attack*. An exhaustive search is similar to a dictionary search, but in the exhaustive case, the computer programme used by the hacker will try every possible combination of permissible characters as the password in order to find a match. Thus if searching for a six character password, the hacker might try aaaaaa, aaaaab, aaaaac, ..., aaaaaz, aaaaa0,...., aaaaa9, aaaaa*, etc. and move systematically through all possible permutations.

This attack will always succeed eventually. Since *every* possible password is tried in turn sooner or later a match will be found. However, there are ways of making an exhaustive search so time consuming for the hacker that it is not successful during the life of the password (i.e. before the exhaustive search is successful the password has been changed). Some password systems insist that the users change their passwords every three months, for example.

2.3.1.1 Learning activity

A hacker is trying to find a password in order to get access to a computer system. He does not have any personal information about the system users, but he knows that all passwords are at least eight characters long and can contain any upper or lower case letter, digit, or other keyboard character. What kind of attack do you think the hacker should attempt?

2.3.2 Number of passwords

An intelligent attacker will carry out dictionary and intelligent or modified dictionary attacks before attempting an exhaustive search. This is because, although an exhaustive search is bound to succeed eventually and a dictionary search may fail, if it succeeds, the dictionary search is much faster.
Suppose that passwords are six characters long.

If the password is made up only of lower case letters, then there are 26 choices for each character in the password. Hence there are $26^6 = 308,915,776 \approx 3^8$ possible passwords of six lower case letters.

If we include lower and capital letters, there are $52^6 \approx 2^{10}$ possible passwords. Adding in digits as well, gives a choice out of 62 for each character in the password and there are now $62^6 \approx 5.7^{10}$ possible passwords.

Finally if we allow any keyboard character including ¡ ¿ * & etc. there are approximately 100 different choices for each character in the password and hence there are $100^6 = 10^{12}$ possible passwords.

In general, if a password is *n* characters long and is made up from an alphabet of *A* different characters, then there are A^n possible different passwords.
Now suppose that a hacker has written a computer program which can try 10,000 passwords per second.

The hacker has a dictionary file which contains 1,000,000 common six letter words. First he runs a dictionary attack trying every word in his dictionary. This will only take him 1, 000, 000/ 10, 000 = 100 seconds to complete.
Making modifications to the dictionary, for example capitalising each word is easy and it only takes the hacker a few more minutes to run modified dictionary searches.

If a dictionary search is not successful then the hacker may try every combination of lowercase letters. This will take 26^6/ 10, 000 seconds, which is just over 8.5 hours to try every combination of lower case letters.
In comparison, if the hacker attempts an exhaustive search using all 100 possible characters in every combination, it will take 100^6/ 10, 000 = 10^8 seconds to complete and this is over three years!

Note that the **average time** for a hacker to find a particular kind of password is only **half** the time taken to do a complete search (i.e. if a user has chosen a dictionary word as their password, then the hacker will, on average, only have to search through half of the dictionary in order to find the password). Likewise, on average, a hacker using an exhaustive search will only have to search through half of the possible passwords before finding a match.

2.3.2.1 Learning activity

1. How many different passwords of lower case letters are there if the password is of length four? Howmany if the password is of length eight?
2. How many different alphanumeric (any letter or digit) passwords are there if the password is of lengthfour? How many if the password is of length eight?
3. On average, how long will it take a hacker to find a password of length eight which is made up entirelyof lowercase letters:
   (a) if the hacker tries only combinations of lowercase letters?
   (b) if the hacker tries all alphanumeric combinations?

Assume that the hacker can try 10,000 passwords per second.

2.3.3 Password spoofing

A *spoofing attack* is when the user is fooled into giving the hacker their password. Spoofing attacks may be very simple or very sophisticated.

Asking the user

This may sound unlikely, but it is a fact that a lot of people will tell you a password if you can convince them that you need to know it. For example, the hacker may phone the user, and tell them that he is from their office computer staff and that there is a problem with the files. All backed-up information is going to be lost so he needs the user password in order to recover the data. Sometimes an approach as simple as this will work and the user is fooled into giving the hacker their password.

This attack will fail if the user has been educated in computer security and refuses to reveal their password.

Fake log-in screens

A more sophisticated spoofing attack is when the hacker sets up a fake log-in screen which exactly resembles the genuine log-in screen for the system. The user is presented with this log-in screen and unsuspectingly enters their user-name and password. The hacker captures this information and then typically gives the user an error message saying that they have incorrectly typed in their password. The genuine log-in screen is then displayed. The user cannot be sure that they did not make a typing mistake, so they type in their user-name and password again and gain access to the system. The user may have no idea that they have been the victim of a spoofing attack.

This attack will fail if the user notices that there is something wrong with the log-in screen and so does not enter their user-name and password. Some log-in interfaces contain patterns or pictures which are impossible to replicate accurately. The attack can be detected (although not prevented) if the user is informed, at every log-in, of the time of the last failed log-in attempt. After a spoof attack, the user thinks that they had a failed log-in. If when the user successfully logs in, the system does not inform them of this failed log-in then the user is alerted to the fact that they may have been the victim of a spoof attack.

Phishing

Phishing is similar to the above. Communications such as emails or instant messages purporting to be from reliable sites such as eBay, PayPal or online banks direct users to a fake website which looks very like the genuine one. Here the user is asked to input their username, password and perhaps their bank

details.

Phishing is a growing problem and attempts to deal with it include legislation, user training, public awareness and technical security measures.

2.3.4 User and system defences

There are various things that users can do in order to minimise the risk of a hacker getting hold of their password. When a user-name/password system is implemented, it is important that the users are informed of the following measures:

- The user should always set up a password and not leave the password option as blank.
- The user should change the default password.
- The user should change their password frequently.
- The user should not use the same password for all systems.
- When changing a password, the user should not just add a digit onto the end of the old password.
- The user should not choose a password that relates to them personally such as their date of birth or the name of their child.
- The user should not choose a dictionary word as their password.
- The user should not choose a password that is too short.
- The user should choose a password that contains a mix of letters and numbers.
- The user should not write their password down or reveal it to anyone.
- Some of these measures can be enforced by the system. Things that the system can do in order to minimise the risk of attack include:
- insist that the user creates a password
- provide the user with a default password
- enforce the user to change the default password at the first log-in
- enforce the user to change their password at frequent intervals (say every three or six months depending on the security need) ▪ check password choices against a dictionary and reject weak passwords ▪ insist that passwords contain an alphanumeric mix of characters
- insist that passwords are at least a minimum length (say 6+ characters depending on the security need)
- limit log-in attempts (a maximum of three attempts is usual) after which time the system administrator will have to reset the password for the user ▪ inform users of each unsuccessful log-in attempt.

2.3.4.1 Learning activity

Suppose a user has an account on a user-name/password system and that they want to change their password. Write a protocol for a secure procedure that should be followed to enable this.

2.4 *Attacking the password file*

The *password file*, where the system stores the data for verifying passwords, is very sensitive to attack. In an insecure system, the password file will be a list of passwords indexed by user-name. A hacker with access to this file has potential knowledge of every password. It is therefore essential that the password file is protected.

There are essentially two ways in which the password file can be protected:

- using cryptographic protection
- implementing access control over the password file.

Ideally, the password file should be both encrypted and protected from unauthorised access by the implementation of access controls.

2.4.1 Cryptographic protection

A password file can be encrypted by using a *one-way function*. After encryption, the password file is just a list of garbled characters. Even if a hacker manages to view the file, it will not help him to gain access to the system.

One-way functions

A problem is said to be *one-way* if it is easy to do one way but hard to do in reverse. A non-mathematical example is making a cup of instant coffee. It is easy to put coffee granules, boiling water and milk into a mug and stir them together to make a cup of coffee. However, given a cup of coffee, it is difficult to reverse the operation and retrieve the separate components of milk, coffee granules and water.

In cryptography, the one-way problems used are mathematical functions. A good example of a mathematical one way function is multiplying/factorising.
A one-way function is a function $f : X \rightarrow Y$ which satisfies the following two properties:

- Given x in X it is easy to compute $y = f(x)$ in Y.
- Given y in Y it is very difficult to find an x in X such that $f(x) = y$.

A good example of a mathematical one-way function is multiplying/factorising. It is very easy (especially given a computer or calculator) to multiply together two integers, even if those integers are very large. However, given the resulting

number, it is very hard (even with access to a computer) to find the two numbers that were originally multiplied together. In this example, both *X* and *Y* are the set of positive integers.Encrypting the password file

The password file can be protected by using a one-way function *f* (*x*) to encrypt the stored passwords as follows:

To create a new user-name/password pair:

- The user inputs their user-name and password *x*.
- The system computes *f* (*x*).
- The password file does not store *x* but instead stores *f* (*x*) indexed by user-name.

To verify a user:

- The system asks for the user-name and password.
- The system computes f (x′) where x′ is the password entered by the user.
- The system checks to see if there is a match between the f (x) stored for the given user-name and f (x′) just computed.
- If f (x) = f (x′) then x = x′ and the user is verified. If f (x) =6 f (x′) then the password entered by the user is incorrect and access to the system is denied.

Attacking an encrypted password file

If a hacker manages to access a password file which has been encrypted using a one-way function, all he will see is the encrypted passwords, indexed by user-names. These encrypted passwords will not enable the hacker to access the system, and the actual passwords are not stored anywhere.

The function used to encrypt the passwords is not usually a secret, so the hacker may try to find an actual password by running a computer program that encrypts a dictionary list or an exhaustive list of passwords and then check to see if the result matches any of the stored encrypted passwords. If a match is found then the hacker has a password and can now gain access to the system.

This type of attack can be thwarted by using a relatively inefficient function to encrypt the passwords. Consider that the hacker may have to encrypt millions of possible passwords before a match is found. If each encryption takes one or two seconds then this will take many days. However, for a genuine individual user a time lapse of a few seconds each time they enter their user-name and password is negligible.

Rainbow tables

If a well known function, such as a secure hashing function, is used to encrypt passwords then pre-computed *rainbow tables* can be used to find passwords

very quickly.

A rainbow table is a table that stores the encryption of all possible passwords of a given format. For example, all passwords that are eight characters long and contain lower case letters and digits. These rainbow tables are huge and require a large amount of storage space and initially a lot of time to compile. However, once they are built they can be searched very quickly to find password matches. These tables are used to retrieve lost user passwords and they are very useful for this purpose. However, in the wrong hands they can obviously be used to find passwords for malicious purposes.

To avoid pre-compiled rainbow tables being used on a security system, the function used to encrypt the passwords should be somehow unique to the system. Pre-compiled tables will therefore not be available. If a user loses or forgets their password it will be irretrievable. An alternative secure method for resetting the lost password to a new value will have to be devised.

2.4.2 Password salting

Password salting is a process used to ensure that all passwords in a system are unique. Most systems insist that all user-names are unique. If a new user tries to create an account with a user-name that is already in use, they will be informed that the user-name is already used and that they should choose another. However, the system cannot inform a new user that the password they have chosen is already in use – that would be a gift for a hacker! Instead, the system adds some *salt* which is another piece of information such as the user-name to all the passwords before encryption. This ensures that every password is unique.

2.4.2.1 Learning activity

Why is it important that every password should be unique? Suppose a hacker found two encrypted passwords that were the same – how could he use this information?

2.4.3 One-time passwords

Given enough time and attempts, a *static* password (i.e. a password which remains the same) may be accessed by an unauthorised attacker. To counter this, some systems are now making use of *one-time passwords* or OTP. By constantly changing the password, the risk of the password being discovered is greatly reduced. Furthermore, an attacker who does find a password, will only be able to use it to gain access to the system once. The next time the attacker

tries to use the password it will be rejected.

One-time passwords typically work in one of three ways.

- A mathematical algorithm is used to generate a new password based on the previous password.
- A time synchronisation protocol is used between the authentication server and the client providing the password.
- A mathematical algorithm is used to create each new password based on a challenge such as a random number chosen by the authentication server and a counter.
-

To implement a OTP, users generally have a *token* (similar to a small electrical keyring, for example) which generates the passwords either based on a mathematical algorithm, or if the token contains a clock synchronised with the authentication server, using the current time. Work is currently being done on the use of mobile phones as tokens. This would be practical and cost effective since most Internet users also have a mobile phone.

2.4.4 Alternative methods for authentication

There are many alternative methods used for identification and authentication. Some are used when the risk is low and others where security is of paramount importance. Of course, in general, as the level of security increases so does the cost, so it is sensible to assess the risk before deciding on the level of security required. Alternative methods include:

- Answering a question that only you are likely to know the answer to such as your mother's maiden name or date of birth. This information is not that hard for a hacker to acquire so provides only a low level of security.
- Presentation of something that you have, such as a credit card or passport. These can be forged or stolen but in general are a good means of identification and authentication.
- Use of finger prints, retina patterns or palm prints. This is a high cost solution, but fingerprints, etc. are fairly hard to replicate and are not something that the genuine user can lose or forget! However, a determined attacker with adequate financial resources can replicate these physical attributes leading to a catastrophic failure of a supposedly high security identity system.

2.4.5 Authentication failure

An identification and authentication system can fail in two ways. Firstly, it can accept an unauthorised user. In this case, the security may be too weak. Secondly, it can reject an authorised user. This may be because security is too high. For example, if the system insists that user passwords are 15 characters

long and include digits and letters then it is likely that genuine users will forget or mistype their passwords leading to authentication failure.

2.5 *Sample examination questions*

Question 1

After reading a newspaper 'scare story' about password security, Walter has decided to implement strict rules regarding the passwords used by the staff in his company. Walter insists that:

- Staff passwords are of length 15 characters or more.
- Staff change their passwords at least once a week.
- Every password contains a mix of letters and digits.

a) Explain why Walter's password policy is likely to make the password system athis company less rather than more secure.

[3]

b) Write a more suitable password policy explaining the importance of each ruleyou suggest.

[10]

c) Assuming that a password cracking program can check 10,000 passwords perminute, calculate the average amount of time that it would take to find a password based on the policy that you have written in part b).

[4]

10 **Question 2**

a) What two properties are required for a *one-way* function?

[2]

b) Describe how a one-way function can be used to protect password files.

[4]

c) Explain why the one-way function used to protect a password file should not bean efficient function.

[4]

3 COMMON THREATS

Computer systems are vulnerable to many threats that can inflict various types of damage resulting in significant losses. This damage can range from errors harming database integrity to fires destroying entire computer centers. Losses can stem, for example, from the actions of supposedly trusted employees defrauding a system, from outside hackers, or from careless data entry clerks.

Precision in estimating computer security-related losses is not possible because many losses are never discovered, and others are "swept under the carpet" to

avoid unfavorable publicity. The effects of various threats varies considerably: some affect the confidentiality or integrity of data while others affect the availability of a system.

This chapter presents a broad view of the risky environment in which systems operate today. The threats and associated losses presented in this chapter were selected based on their prevalence and significance in the current computing environment and their expected growth. This list is not exhaustive, and some threats may combine elements from more than one area. This overview of many of today's common threats may prove useful to organizations studying their own threat environments; however, the perspective of this chapter is very broad.

Thus, threats against particular systems could be quite different from those discussed here. To control the risks of operating an information system, managers and users need to know the vulnerabilities of the system and the threats that may exploit them. Knowledge of the threat environment allows the system manager to implement the most cost-effective security measures.

In some cases, managers may find it more cost-effective to simply tolerate the expected losses. Such decisions should be based on the results of a risk analysis.

3.1 *Errors and Omissions*

Errors and omissions are an important threat to data and system integrity. These errors are caused not only by data entry clerks processing hundreds of transactions per day, but also by all types of users who create and edit data. Many programs, especially those designed by users for personal computers, lack quality control measures. However, even the most sophisticated programs cannot detect all types of input errors or omissions. A sound awareness and training program can help an organization reduce the number and severity of errors and omissions.

Users, data entry clerks, system operators, and programmers frequently make errors that contribute directly or indirectly to security problems. In some cases, the error is the threat, such as a data entry error or a programming error that crashes a system. In other cases, the errors create vulnerabilities. Errors can occur during all phases of the systems life cycle. A long-term survey of computer-related economic losses conducted by Robert Courtney, a computer security consultant and former member of the Computer System Security and Privacy Advisory Board, found that 65 percent of losses to organizations were the result of errors and omissions.This figure was relatively consistent between both private and public sector organizations.

Programming and development errors, often called "bugs," can range in severity from benign to catastrophic. In a 1989 study for the House Committee on Science, Space and Technology, entitled Bugs in the Program, the staff of the Subcommittee on Investigations and Oversight summarized the scope and severity of this problem in terms of government systems as follows:
As expenditures grow, so do concerns about the reliability, cost and accuracy of ever-larger and more complex software systems. These concerns are heightened as computers perform more critical tasks, where mistakes can cause financial turmoil, accidents, or in extreme cases, death.

Since the study's publication, the software industry has changed considerably, with measurable improvements in software quality. Yet software "horror stories" still abound, and the basic principles and problems analyzed in the report remain the same. While there have been great improvements in program quality, as reflected in decreasing errors per 1000 lines of code, the concurrent growth in program size often seriously diminishes the beneficial effects of these program quality enhancements.

Installation and maintenance errors are another source of security problems. For example, an audit by the President's Council for Integrity and Efficiency (PCIE) in 1988 found that every one of the ten mainframe computer sites studied had installation and maintenance errors that introduced significant security vulnerabilities.

3.2 *Fraud and Theft*

Computer systems can be exploited for both fraud and theft both by "automating" traditional methods of fraud and by using new methods. For example, individuals may use a computer to skim small amounts of money from a large number of financial accounts, assuming that small discrepancies may not be investigated. Financial systems are not the only ones at risk. Systems that control access to any resource are targets (e.g., time and attendance systems, inventory systems, school grading systems, and long-distance telephone systems).

Computer fraud and theft can be committed by insiders or outsiders. Insiders (i.e., authorized users of a system) are responsible for the majority of fraud. A 1993 InformationWeek/Emst and Young study found that 90 percent of Chief Information Officers viewed employees "who do not need to know" information as threats. The U.S. Department of Justice's Computer Crime Unit contends that "insiders constitute the greatest threat to computer systems. "^^ Since insiders have both access to and familiarity with the victim computer system (including what resources it controls and its flaws), authorized system users are in a better position to commit crimes.

Insiders can be both general users (such as clerks) or technical staff members.

An organization's former employees, with their knowledge of an organization's operations, may also pose a threat, particularly if their access is not terminated promptly.

In addition to the use of technology to commit fraud and theft, computer hardware and software may be vulnerable to theft. For example, one study conducted by Safeware Insurance found that $882 million worth of personal computers was lost due to theft in 1992.

3.3 *Employee Sabotage*

Employees are most familiar with their employer's computers and applications, including knowing what actions might cause the most damage, mischief, or sabotage. The downsizing of organizations in both the public and private sectors has created a group of individuals with organizational knowledge, who may retain potential system access (e.g., if system accounts are not deleted in a timely manner). The number of incidents of employee sabotage is believed to be much smaller than the instances of theft, but the cost of such incidents can be quite high.

Martin Sprouse, author of Sabotage in the American Workplace, reported that the motivation for sabotage can range from altruism to revenge:

As long as people feel cheated, bored, harassed, endangered, or betrayed at work, sabotage will be used as a direct method of achieving job satisfaction - the kind that never has to get the bosses' approval.

3.4 *Loss of Physical and Infrastructure Support*

The loss of supporting infrastructure includes power failures (outages, spikes, and brownouts), loss of communications, water outages and leaks, sewer problems, lack of transportation services, fire, flood, civil unrest, and strikes. These losses include such dramatic events as the explosion at the World Trade Center and the Chicago tunnel flood, as well as more common events, such as broken water pipes. A loss of infrastructure often results in system downtime, sometimes in unexpected ways. For example, employees may not be able to get to work during a winter storm, although the computer system may be functional.

3.5 *Malicious Hackers*

The term malicious hackers, sometimes called crackers, refers to those who

break into computers without authorization. They can include both outsiders and insiders. Much of the rise of hacker activity is often attributed to increases in connectivity in both government and industry. One 1992 study of a particular Internet site (i.e., one computer system) found that hackers attempted to break in at least once every other day.

The hacker threat should be considered in terms of past and potential ftiture damage. Although current losses due to hacker attacks are significantly smaller than losses due to insider theft and sabotage, the hacker problem is widespread and serious. One example of malicious hacker activity is that directed against the public telephone system.

Studies by the National Research Council and the National Security Telecommunications Advisory Committee show that hacker activity is not limited to toll fraud. It also includes the ability to break into telecommunications systems (such as switches), resulting in the degradation or disruption of system availability. While unable to reach a conclusion about the degree of threat or risk, these studies underscore the abUity of hackers to cause serious damage.

The hacker threat often receives more attention than more common and dangerous threats. The U.S. Department of Justice's Computer Crime Unit suggests three reasons for this.

- First, the hacker threat is a more recently encountered threat. Organizations have always had to worry about the actions of their own employees and could use disciplinary measures to reduce that threat. However, these measures are ineffective against outsiders who are not subject to the rules and regulations of the employer.

- Second, organizations do not know the purposes of a hacker - some hackers browse, some steal, some damage. This inability to identify purposes can suggest that hacker attacks have no limitations.

- Third, hacker attacks make people feel vulnerable, particularly because their identity is unknown. For example, suppose a painter is hired to paint a house and, once inside, steals a piece ofjewelry. Other homeowners in the neighborhood may not feel threatened by this crime and will protect themselves by not doing business with that painter. But if a burglar breaks into the same house and steals the same

3.6 *Industrial Espionage*

Industrial espionage is the act of gathering proprietary data from private

companies or the government^'* for the purpose of aiding another company(ies). Industrial espionage can be perpetrated either by companies seeking to improve their competitive advantage or by governments seeking to aid their domestic industries. Foreign industrial espionage carried out by a government is often referred to as economic espionage. Since information is processed and stored on computer systems, computer security can help protect against such threats; it can do little, however, to reduce the threat of authorized employees selling that information.

Industrial espionage is on the rise. A 1992 study sponsored by the American Society for Industrial Security (ASIS) found that proprietary business information theft had increased 260 percent since 1985. The data indicated 30 percent of the reported losses in 1991 and 1992 had foreign involvement. The study also found that 58 percent of thefts were perpetrated by current or former employees. The three most damaging types of stolen information were pricing information, manufacturing process information, and product development and specification information. Other types of information stolen included customer lists, basic research, sales data, personnel data, compensation data, cost data, proposals, and strategic plans.

Within the area of economic espionage, the Central Intelligence Agency has stated that the main objective is obtaining information related to technology, but that information on U.S. Government policy deliberations concerning foreign affairs and information on commodities, interest rates, and other economic factors is also a target. The Federal Bureau of Investigation concurs that technology-related information is the main target, but also lists corporate proprietary information, such as negotiating positions and other contracting data, as a target.

3.7 *Malicious Code*

Malicious code refers to viruses, worms, Trojan horses, logic bombs, and other "uninvited" software. Sometimes mistakenly associated only with personal computers, malicious code can attack other platforms.

A 1993 study of viruses found that while the number of known viruses is increasing exponentially, the number of virus incidents is not.The study concluded that viruses are becoming more prevalent, but only "gradually."

The rate of PC-DOS virus incidents in medium to large North American businesses appears to be approximately 1 per 1000 PCs per quarter; the number of infected machines is perhaps 3 or 4 times this figure if we assume that most such businesses are at least weakly protected against viruses.

Actual costs attributed to the presence of malicious code have resulted primarily from system outages and staff time involved in repairing the systems. Nonetheless, these costs can be significant.

Virus: A code segment that replicates by attaching copies of itself to existing executables. The new copy of the virus is executed when a user executes the new host program. The virus may include an additional "payload" that triggers when specific conditions are met. For example, some viruses display a text string on a particular date. There are many types of viruses, including variants, overwriting, resident, stealth, and polymorphic.

Trojan Horse: A program that performs a desired task, but that also includes unexpected (and undesirable) fiinctions. Consider as an example an editing program for a multiuser system. This program could be modified to randomly delete one of the users' files each time they perform a usefiil function (editing), but the deletions are unexpected and definitely undesired!

Worm: A self-replicating program that is self-contained and does not require a host program. The program creates a copy of itself and causes it to execute; no user intervention is required. Worms commonly use network services to propagate to other host systems.

3.8 *Foreign Government Espionage*

In some instances, threats posed by foreign government intelligence services may be present. In addition to possible economic espionage, foreign intelligence services may target unclassified systems to further their intelligence missions. Some unclassified information that may be of interest includes travel plans of senior officials, civil defense and emergency preparedness, manufacturing technologies, satellite data, personnel and payroll data, and law enforcement, investigative, and security files. Guidance should be sought from the cognizant security office regarding such threats.

3.9 *Threats to Personal Privacy*

The accumulation of vast amounts of electronic information about individuals by governments, credit bureaus, and private companies, combined with the ability of computers to monitor, process, and aggregate large amounts of information about individuals have created a threat to individual privacy. The possibility that all of this information and technology may be able to be linked together has arisen as a specter of the modern information age. This is often referred to as "Big Brother." To guard against such intrusion. Congress has enacted legislation, over the years, such as the Privacy Act of 1974 and the Computer Matching and Privacy Protection Act of 1988, which defines the boundaries of the legitimate uses of personal information collected by the government.

The threat to personal privacy arises from many sources. In several cases federal and state employees have sold personal information to private investigators or other "information brokers." One such case was uncovered in 1992 when the Justice Department announced the arrest of over two dozen individuals engaged in buying and selling information from Social Security Administration (SSA) computer files.'*^ During the investigation, auditors learned that SSA employees had unrestricted access to over 130 million employment records. Another investigation found that 5 percent of the employees in one region of the IRS had browsed through tax records of friends, relatives, and celebrities.'Some of the employees used the information to create fraudulent tax refunds, but many were acting simply out of curiosity.

As more of these cases come to light, many individuals are becoming increasingly concerned about threats to their personal privacy. A July 1993 special report in MacWorld cited polling data taken by Louis Harris and Associates showing that in 1970 only 33 percent of respondents were concerned about personal privacy. By 1990, that number had jumped to 79 percent.

While the magnitude and cost to society of the personal privacy threat are difficult to gauge, it is apparent that information technology is becoming powerful enough to warrant fears of both government and corporate "Big Brothers." Increased awareness of the problem is needed.

4 Computer security policy

In discussions of computer security, the term policy has more than one meaning. Policy is senior management's directives to create a computer security program, establish its goals, and assign responsibilities. The term policy is also used to refer to the specific security rules for particular systems.

Additionally, policy may refer to entirely different matters, such as the specific managerial decisions setting an organization's e-mail privacy policy or fax security policy.

In this chapter the term computer security policy is defined as the "documentation of Policy means different things to different people. The computer security decisions" - which covers term "poUcy" is used in this chapter in a broad all the types of poUcy described above.'^ In making these decisions, managers face hard choices involving resource allocation, competing objectives, and organizational strategy related to protecting both technical and information resources as well as guiding employee behavior. Managers at all levels make choices that can result in policy, with the scope of the policy's applicability varying according to the scope of the manager's authority. In this chapter we use the term policy in a broad manner to encompass all of the types

of policy described above - regardless of the level of manager who sets the particular policy.

Managerial decisions on computer security issues vary greatly. To differentiate among various kinds of policy, this chapter categorizes them into three basic types:

- Program policy is used to create an organization's computer security program.
- Issue-specific policies address specific issues of concern to the organization.
- System-specific policies focus on decisions taken by management to protect a particular system.

Procedures, standards, and guidelines are used to describe how these policies will be implemented within an organization.

Familiarity with various types and components of policy will aid managers in addressing computer security issues important to the organization. Effective policies ultimately result in the development and implementation of a better computer security program and better protection of systems and information.

These types of policy are described to aid the reader's understanding/' It is not important that one categorizes specific organizational policies into these three categories; it is more important to focus on the functions of each.

4.1 *Tools to Implement Policy:*

Standards, Guidelines, and Procedures

Because policy is written at a broad level, organizations also develop standards, guidelines, and procedures that offer users, managers, and others a clearer approach to implementing policy and meeting organizational goals.

Standards and guidelines specify technologies and methodologies to be used to secure systems. Procedures are yet more detailed steps to be followed to accomplish particular security-related tasks. Standards, guidelines, and procedures may be promulgated throughout an organization via handbooks, regulations, or manuals.

Organizational standards (not to be conftised with American National Standards, FEPS, Federal Standards, or other national or international standards) specify uniform use of specific technologies, parameters, or

procedures when such uniform use will benefit an organization. Standardization of organizationwide identification badges is a typical example, providing ease of employee mobility and automation of entry/exit systems. Standards are normally compulsory within an organization.

Guidelines assist users, systems personnel, and others in effectively securing their systems. The nature of guidelines, however, immediately recognizes that systems vary considerably, and imposition of standards is not always achievable, appropriate, or cost-effective. For example, an organizational guideline may be used to help develop system-specific standard procedures.

Guidelines are often used to help ensure that specific security measures are not overlooked, although they can be implemented, and correctly so, in more than one way.

Procedures normally assist in complying with applicable security policies, standards, and guidelines. They are detailed steps to be followed by users, system operations personnel, or others to accomplish a particular task (e.g., preparing new user accounts and assigning the appropriate privileges).

Some organizations issue overall computer security manuals, regulations, handbooks, or similar documents. These may mix policy, guidelines, standards, and procedures, since diey are closely linked. While manuals and regulations can serve as important tools, it is often useful if they clearly distinguish between policy and its implementation. This can help in promoting flexibility and cost-effectiveness by offering alternative implementation approaches to achieving policy goals.

4.2 *Program Policy*

A management official, normally the head of the organization or the senior administration official, issues program policy to establish (or restructure) the organization's computer security program and its basic structure. This high-level policy defines the purpose of the program and its scope within the organization; assigns responsibilities (to the computer security organization) for direct program implementation, as well as other responsibilities to related offices (such as the Information Resources Management [IRM] organization); and addresses compliance issues.

Program policy sets organizational strategic directions for security and assigns resources for its implementation.

4.2.1 Basic Components of Program Policy

Components of program policy should address:

Purpose. Program policy normally includes a statement describing why the program is being established. This may include defining the goals of the program. Security-related needs, such as integrity, availability, and confidentiality, can form the basis of organizational goals established in policy.

For instance, in an organization responsible for maintaining large mission-critical databases, reduction in errors, data loss, data corruption, and recovery might be specifically stressed. In an organization responsible for maintaining confidential personal data, however, goals might emphasize stronger protection against unauthorized disclosure.

Scope. Program policy should be clear as to which resources ~ including facilities, hardware, and software, information, and personnel — the computer security program covers. In many cases, the program will encompass aU systems and organizational personnel, but this is not always true. In some instances, it may be appropriate for an organization's computer security program to be more limited in scope.

Responsibilities. Once the computer security program is established, its management is Program policy which establishes the security program and normally assigned to either a newly created or assigns program management and supporting existing office.

The responsibilities of officials and offices throughout the organization also need to be addressed, including line managers, applications owners, users, and the data processing or IRM organizations. This section of the policy statement, for example, would distinguish between the responsibilities of computer services providers and those of the managers of applications using the provided services. The policy could also establish operational security offices for major systems, particularly those at high risk or most critical to organizational operations. It also can serve as the basis for establishing employee

Accountability. At the program level, responsibilities should be specifically assigned to those organizational elements and officials responsible for the implementation and continuity of the computer security policy.

Compliance. Program policy typically will address two compliance issues:

1) General compliance to ensure meeting the requirements to establish a program and the responsibilities assigned therein to various organizational components. Often an oversight office (e.g., the Inspector General) is assigned responsibility for monitoring compliance, including how well the organization is implementing management's priorities for the program.

2) The use of specified penalties and disciplinary actions. Since the security policy is a high-level document, specific penalties for various infractions are normally not detailed here; instead, the policy may authorize the creation of compliance structures that include violations and specific disciplinary action(s).

Those developing compliance policy should remember that violations of policy can be unintentional on the part of employees. For example, nonconformance can often be due to a lack of knowledge or training.

4.3 *Issue-Specific Policy*

Whereas program policy is intended to address the broad organizationwide computer security program, issue-specific policies are developed to focus on areas of current relevance and concern (and sometimes controversy) to an organization. Management may find it appropriate, for example, to issue a policy on how the organization will approach contingency planning (centralized vs. decentralized) or the use of a particular methodology for managing risk to systems.

A policy could also be issued, for example, on the appropriate use of a cutting-edge technology (whose security vulnerabilities are still largely unknown) within the organization.

Issue-specific policies may also be appropriate when new issues arise, such as when implementing a recently passed law requiring additional protection of particular information. Program policy is usually broad enough that it does not require much modification over time, whereas issue-specific policies are likely to require more frequent revision as changes in technology and related factors take place.

In general, for issue- specific and system-specific policy, the issuer is a senior official; the more global, controversial, or resource-intensive, the more senior the issuer.

4.3.1 Example Topics for Issue-Specific

There are many areas for which issue-specific policy may be appropriate. Two examples are explained below.

Internet Access. Many organizations are looking at the Internet as a means for expanding their research opportunities and communications. Unquestionably, connecting to the Internet yields many benefits - and some disadvantages. Some issues an Internet access policy may address include who will have access,

which types of systems may be connected to the network, what types of information may be transmitted via the network, requirements for user authentication for Internet-connected systems, and the use of firewalls and secure gateways.

E-Mail Privacy. Users of computer e-mail systems have come to rely upon that service for informal communication with colleagues and others. However, since the system is typically owned by the employing organization, from time-to-time, management may wish to monitor the employee's e-mail for various reasons (e.g., to be sure that it is used for business purposes only or if they are suspected of distributing viruses, sending offensive e-mail, or disclosing organizational secrets.) On the other hand, users may have an expectation of privacy, similar to that accorded U.S. mail. Policy in this area addresses what level of privacy will be accorded e-mail and the circumstances under which it may or may not be read.

4.3.2 Basic Components of Issue-Specific Policy

As suggested for program policy, a useful structure for issue-specific policy is to break the policy into its basic components.

Issue Statement. To formulate a policy on an issue, managers first must define the issue with any relevant terms, distinctions, and conditions included. It is also often usefial to specify the goal or justification for the policy - which can be helpfiil in gaining compliance with the policy. For example, an organization might want to develop an issue-specific policy on the use of "unofficial software," which might be defined to mean any software not approved, purchased, screened, managed, and owned by the organization. Additionally, the applicable distinctions and conditions might then need to be included, for instance, for software privately owned by employees but approved for use at work, and for software owned and used by other businesses under contract to the organization.

Statement of the Organization's Position. Once the issue is stated and related terms and conditions are discussed, this section is used to clearly state the organization's position (i.e., management's decision) on the issue. To continue the previous example, this would mean stating whether use of unofficial software as defined is prohibited in all or some cases, whether there are further guidelines for approval and use, or whether case-by-case exceptions will be granted, by whom, and on what basis.

Applicability. Issue-specific policies also need to include statements of applicability. This means clarifying where, how, when, to whom, and to what a particular policy applies. For example, it could be that the hypothetical policy on unofficial software is intended to apply only to the organization's own on-site resources and employees and not to contractors with offices at other locations. Additionally, the policy's applicability to employees travelling among

different sites and/or working at home who need to transport and use disks at multiple sites might need to be clarified.

Roles and Responsibilities. The assignment of roles and responsibilities is also usually included in issue-specific policies. For example, if the policy permits unofficial software privately owned by employees to be used at work with the appropriate approvals, then the approval authority granting such permission would need to be stated. (Policy would stipulate, who, by position, has such authority.) Likewise, it would need to be clarified who would be responsible for ensuring that only approved software is used on organizational computer resources and, perhaps, for monitoring users in regard to unofficial software.

Compliance. For some types of policy, it may be appropriate to describe, in some detail, the infractions that are unacceptable, and the consequences of such behavior. Penalties may be explicitly stated and should be consistent with organizational personnel policies and practices.

When used, they should be coordinated with appropriate officials and offices and, perhaps, employee bargaining units. It may also be desirable to task a specific office within the organization to monitor compliance.

Points of Contact and Supplementary Information. For any issue-specific policy, the appropriate individuals in the organization to contact for further information, guidance, and compliance should be indicated. Since positions tend to change less often than the people occupying them, specific positions may be preferable as the point of contact. For example, for some issues the point of contact might be a line manager; for other issues it might be a facility manager, technical support person, system administrator, or security program representative. Using the above example once more, employees would need to know whether the point of contact for questions and procedural information would be their immediate superior, a system administrator, or a computer security official.

Guidelines and procedures often accompany policy. The issue-specific policy on unofficial software, for example, might include procedural guidelines for checking disks brought to work that had been used by employees at other locations.

4.3.3 Some Helpful Hints on Policy

To be effective, policy requires visibility. Visibility aids implementation of policy by helping to ensure policy is fully communicated throughout the organization. Management presentations, videos, panel discussions, guest speakers, question/answer forums, and newsletters increase visibility. The organization's computer security training and awareness program can effectively notify users of new policies. It also can be used to familiarize new employees with the organization's policies.

Computer security policies should be introduced in a manner that ensures that management's unqualified support is clear, especially in environments where employees feel inundated with policies, directives, guidelines, and procedures.

The organization's policy is the vehicle for emphasizing management's commitment to computer security and making clear their expectations for employee performance, behavior, and accountability. To be effective, policy should be consistent with other existing directives, laws, organizational culture, guidelines, procedures, and the organization's overall mission. It should also be integrated into and consistent with other organizational poUcies (e.g., personnel policies). One way to help ensure this is to coordinate policies during development with other organizational offices.

4.4 *Examples of security policies*

EXAMPLE: A university disallows cheating, which is defined to include copying another student's homework assignment (with or without permission). A computer science class requires the students to do their homework on the department's computer.

One student notices that a second student has not read protected the file containing her homework and copies it. Has either student (or have both students) breached security?

The second student has not, despite her failure to protect her homework. The security policy requires no action to prevent files from being read. Although she may have been too trusting, the policy does not ban this; hence, the second student has not breached security.

The first student has breached security. The security policy disallows the copying of homework, and the student has done exactly that. Whether the security policy specifically states that "files containing homework shall not be copied" or simply says that "users are bound by the rules of the university" is irrelevant; in the latter case, one of those rules bans cheating. If the security policy is silent on such matters, the most reasonable interpretation is that the policy disallows actions that the university disallows, because the computer science department is part of the university.

The retort that the first user could copy the files, and therefore the action is allowed, confuses *mechanism* with *policy*. The distinction is sharp:
A *security mechanism* is an entity or procedure that enforces some part of the security policy.

EXAMPLE: In the preceding example, the policy is the statement that no student may copy another student's homework. One mechanism is the file access controls; if the second student had set permissions to prevent the first student

from reading the file containing her homework, the first student could not have copied that file.

EXAMPLE: Another site's security policy states that information relating to a particular product is proprietary and is not to leave the control of the company. The company stores its backup tapes in a vault in the town's bank (this is common practice in case the computer installation is completely destroyed).

The company must ensure that only authorized employees have access to the backup tapes even when the tapes are stored off-site; hence, the bank's controls on access to the vault, and the procedures used to transport the tapes to and from the bank, are considered security mechanisms. Note that these mechanisms are not technical controls built into the computer. Procedural, or operational, controls also can be security mechanisms.

Security policies are often implicit rather than explicit. This causes confusion, especially when the policy is defined in terms of the mechanisms. This definition may be ambiguous—for example, if some mechanisms prevent a specific action and others allow it. Such policies lead to confusion, and sites should avoid them.

EXAMPLE: The UNIX operating system, initially developed for a small research group, had mechanisms sufficient to prevent users from accidentally damaging one another's files; for example, the user *ken* could not delete the user *dmr*'s files (unless *dmr* had set the files and the containing directories appropriately). The implied security policy for this friendly environment was "do not delete or corrupt another's files, and any file not protected may be read."

When the UNIX operating system moved into academic institutions and commercial and government environments, the previous security policy became inadequate; for example, some files had to be protected from individual users (rather than from groups of users). Not surprisingly, the security mechanisms were inadequate for those environments.

The difference between a policy and an abstract description of that policy is crucial to the analysis that follows.

A *security model* is a model that represents a particular policy or set of policies. A model abstracts details relevant for analysis. Analyses rarely discuss particular policies; they usually focus on *specific characteristics* of policies, because many policies exhibit these characteristics; and the more policies with those characteristics, the more useful the analysis. By the HRU result (see Theorem 3–2), no single nontrivial analysis can cover all policies, but restricting the class of security policies sufficiently allows meaningful analysis of that class of policies.

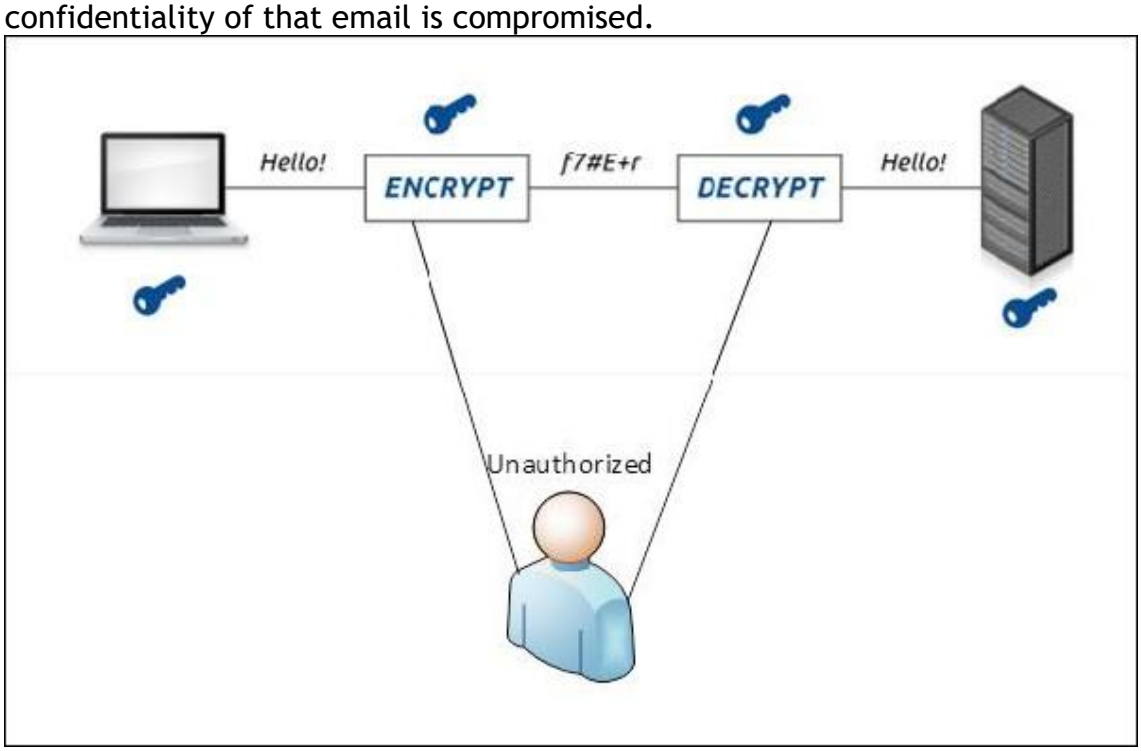
4.5 *Types of Security Policies*

4.5.1 Terms explained

Confidentiality

Confidentiality is the concealment of information or resources. Also, there is a need to keep information secret from other third parties that want to have access to it, so just the right people can access it.

Example in real life – Let's say there are two people communicating via an encrypted email they know the decryption keys of each other and they read the email by entering these keys into the email program. If someone else can read these decryption keys when they are entered into the program, then the confidentiality of that email is compromised.



Integrity

Integrity is the trustworthiness of data in the systems or resources by the point of view of preventing unauthorized and improper changes. Generally, Integrity is composed of two sub-elements – data-integrity, which it has to do with the content of the data and authentication which has to do with the origin of the data as such information has values only if it is correct.

Example in real life – Let's say you are doing an online payment of 5 USD, but your information is tampered without your knowledge in a way by sending to the seller 500 USD, this would cost you too much.

In this case cryptography plays a very major role in ensuring data integrity. Commonly used methods to protect data integrity includes hashing the data you receive and comparing it with the hash of the original message. However, this

means that the hash of the original data must be provided in a secure way.

Availability

Availability refers to the ability to access data of a resource when it is needed, as such the information has value only if the authorized people can access at right time. Denying access to data nowadays has become a common attack. Imagine a downtime of a live server how costly it can be.
Example in real life – Let's say a hacker has compromised a webserver of a bank and put it down. You as an authenticated user want to do an e-banking transfer but it is impossible to access it, the undone transfer is a money lost for the bank.

4.5.2 Policies

Each site has its own requirements for the levels of confidentiality, integrity, and availability, and the site policy states these needs for that particular site.

A *military security policy* (also called a *governmental security policy*) is a security policy developed primarily to provide confidentiality. The name comes from the military's need to keep information, such as the date that a troop ship will sail, secret. Although integrity and availability are important, organizations using this class of policies can overcome the loss of either—for example, by using orders not sent through a computer network. But the compromise of confidentiality would be catastrophic, because an opponent would be able to plan countermeasures (and the organization may not know of the compromise).

Confidentiality is one of the factors of privacy, an issue recognized in the laws of many government entities (such as the Privacy Act of the United States and similar legislation in Sweden). Aside from constraining what information a government entity can legally obtain from individuals, such acts place constraints on the disclosure and use of that information. Unauthorized disclosure can result in penalties that include jail or fines; also, such disclosure undermines the authority and respect that individuals have for the government and inhibits them from disclosing that type of information to the agencies so compromised.

A *commercial security policy* is a security policy developed primarily to provide integrity.The name comes from the need of commercial firms to prevent tampering with their data, because they could not survive such compromises. For example, if the confidentiality of a bank's computer is compromised, a customer's account balance may be revealed. This would certainly embarrass the bank and possibly cause the customer to take her business elsewhere. But the loss to the bank's "bottom line" would be minor.

However, if the integrity of the computer holding the accounts were compromised, the balances in the customers' accounts could be altered, with financially ruinous effects.

Some integrity policies use the notion of a transaction; like database specifications, they require that actions occur in such a way as to leave the database in a consistent state. These policies, called *transaction-oriented integrity security policies*, are critical to organizations that require consistency of databases.

EXAMPLE: When a customer moves money from one account to another, the bank uses a well-formed transaction. This transaction has two distinct parts: money is first debited to the original account and then credited to the second account. Unless both parts of the transaction are completed, the customer will lose the money. With a wellformed transaction, if the transaction is interrupted, the state of the database is still consistent—either as it was before the transaction began or as it would have been when the transaction ended. Hence, part of the bank's security policy is that all transactions must be well-formed.

The role of trust in these policies highlights their difference. Confidentiality policies place no trust in objects; so far as the policy is concerned, the object could be a factually correct report or a tale taken from *Aesop's Fables*. The policy statement dictates whether that object can be disclosed. It says nothing about whether the object should be believed.

Integrity policies, to the contrary, indicate how much the object can be trusted. Given that this level of trust is correct, the policy dictates what a subject can do with that object. But the crucial question is how the level of trust is assigned. For example, if a site obtains a new version of a program, should that program have high integrity (that is, the site trusts the new version of that program) or low integrity (that is, the site does not yet trust the new program), or should the level of trust be somewhere in between (because the vendor supplied the program, but it has not been tested at the local site as thoroughly as the old version)? This makes integrity policies considerably more nebulous than confidentiality policies. The assignment of a level of confidentiality is based on what the classifier wants others to know, but the assignment of a level of integrity is based on what the classifier subjectively believes to be true about the trustworthiness of the information.

Two other terms describe policies related to security needs; because they appear elsewhere, we define them now.

A *confidentiality policy* is a security policy dealing only with confidentiality.

An *integrity policy* is a security policy dealing only with integrity.

Both confidentiality policies and military policies deal with confidentiality; however, a confidentiality policy does not deal with integrity at all, whereas a military policy may. A similar distinction holds for integrity policies and commercial policies.

4.6 *The Role of Trust*

The role of trust is crucial to understanding the nature of computer security. This notes presents theories and mechanisms for analyzing and enhancing computer security, but any theories or mechanisms rest on certain assumptions. When someone understands the assumptions her security policies, mechanisms, and procedures rest on, she will have a very good understanding of how effective those policies, mechanisms, and procedures are. Let us examine the consequences of this maxim.

A system administrator receives a security patch for her computer's operating system. She installs it. Has she improved the security of her system? She has indeed, given the correctness of certain assumptions:

1) She is assuming that the patch came from the vendor and was not tampered with in transit, rather than from an attacker trying to trick her into installing a bogus patch that would actually open security holes. Winkler describes a penetration test in which this technique enabled attackers to gain direct access to the computer systems of the target.

2) She is assuming that the vendor tested the patch thoroughly. Vendors are often under considerable pressure to issue patches quickly and sometimes test them only against a particular attack. The vulnerability may be deeper, however, and other attacks may succeed. When someone released an exploit of one vendor's operating system code, the vendor released a correcting patch in 24 hours. Unfortunately, the patch opened a second hole, one that was far easier to exploit. The next patch (released 48 hours later) fixed both problems correctly.

3) She is assuming that the vendor's test environment corresponds to her environment. Otherwise, the patch may not work as expected. As an example, a vendor's patch once reset ownerships of executables to the user *root*. At some installations, maintenance procedures required that these executables be owned by the user *bin*. The vendor's patch had to be undone and fixed for the local configuration. This assumption also covers possible conflicts between different patches, as well as patches that conflict with one another (such as patches from different vendors of software that the system is using).

4) She is assuming that the patch is installed correctly. Some patches are

simple to install, because they are simply executable files. Others are complex, requiring the system administrator to reconfigure networkoriented properties, add a user, modify the contents of a registry, give rights to some set of users, and then reboot the system. An error in any of these steps could prevent the patch from correcting the problems, as could an inconsistency between the environments in which the patch was developed and in which the patch is applied. Furthermore, the patch may claim to require specific privileges, when in reality the privileges are unnecessary and in fact dangerous.

These assumptions are fairly high-level, but invalidating any of them makes the patch a potential security problem.

Assumptions arise also at a much lower level. Consider formal verification, an oft-touted panacea for security problems. The important aspect is that formal verification provides a formal mathematical proof that a given program *P* is correct—that is, given any set of inputs *i*, *j*, *k*, the program *P* will produce the output *x* that its specification requires. This level of assurance is greater than most existing programs provide, and hence makes *P* a desirable program. Suppose a security-related programs have been formally verified for the operating system *O*. What assumptions would be made when it was installed?

1) The formal verification of *S* is correct—that is, the proof has no errors. Because formal verification relies on automated theorem provers as well as human analysis, the theorem provers must be programmed correctly.

2) The assumptions made in the formal verification of *S* are correct; specifically, the preconditions hold in the environment in which the program is to be executed. These preconditions are typically fed to the theorem provers as well as the program *S*. An implicit aspect of this assumption is that the version of *O* in the environment in which the program is to be executed is the same as the version of *O* used to verify *S*.

3) The program will be transformed into an executable whose actions correspond to those indicated by the source code; in other words, the compiler, linker, loader, and any libraries are correct. An experiment with one version of the UNIX operating system demonstrated how devastating a rigged compiler could be, and attackers have replaced libraries with others that performed additional functions, thereby increasing security risks.

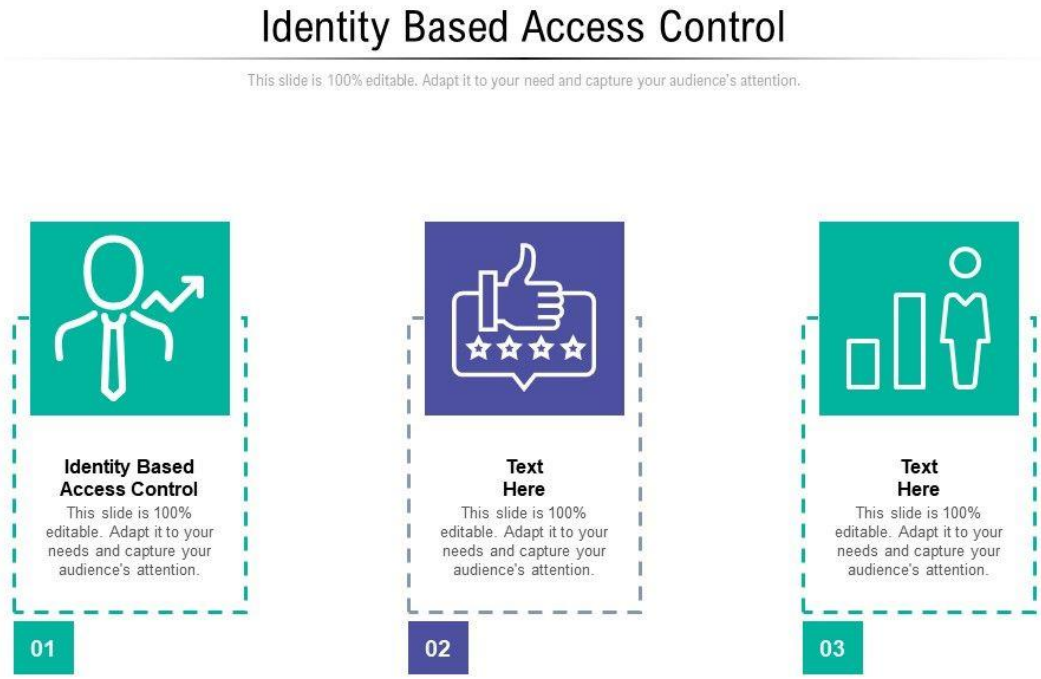
4) The hardware will execute the program as intended. A program that relies on floating point calculations would yield incorrect results on some computer CPU chips, regardless of any formal verification of the program, owing to a flaw in these chips. Similarly, a program that relies on inputs from hardware assumes that specific conditions cause those inputs.

The point is that *any* security policy, mechanism, or procedure is based on assumptions that, if incorrect, destroy the superstructure on which it is built. Analysts and designers (and users) must bear this in mind, because unless they understand what the security policy, mechanism, or procedure is based on, they jump from an unwarranted assumption to an erroneous conclusion.

4.7 *Types of Access Control*

A security policy may use two types of access controls, alone or in combination. In one, access control is left to the discretion of the owner. In the other, the operating system controls access, and the owner cannot override the controls. The first type is based on user identity and is the most widely known:

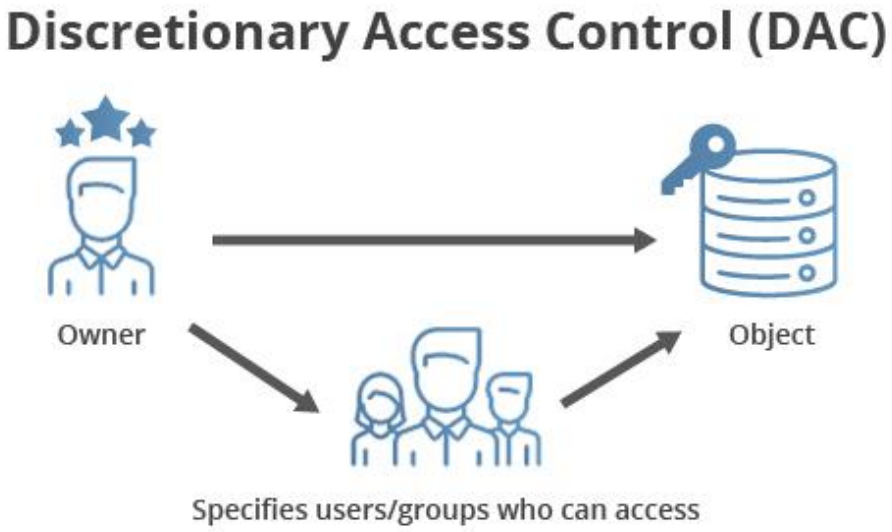
If an individual user can set an access control mechanism to allow or deny access to an object, that mechanism is a *discretionary access control* (DAC), also called an *identity-based access control* (IBAC).



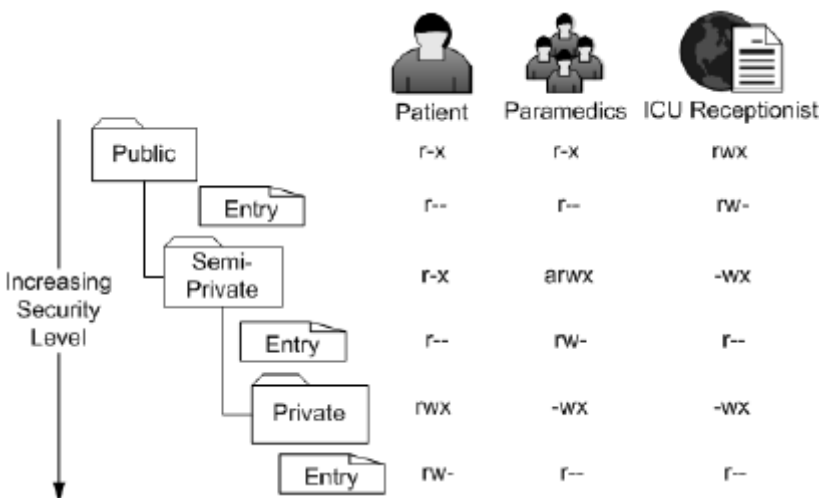
Discretionary access controls base access rights on the identity of the subject and the identity of the object involved. Identity is the key; the owner of the object constrains who can access it by allowing only particular subjects to have access. The owner states the constraint in terms of the identity of the subject, or the owner of the subject.

EXAMPLE: Suppose a child keeps a diary. The child controls access to the diary, because she can allow someone to read it (grant read access) or not allow someone to read it (deny read access). The child allows her mother to read it, but no one else.

This is a discretionary access control because access to the diary is based on the identity of the subject (mom) requesting read access to the object (the diary).



The second type of access control is based on fiat, and identity is irrelevant: When a system mechanism controls access to an object and an individual user cannot alter that access, the control is a *mandatory access control* (MAC), occasionally called a *rule-based access control*.



The operating system enforces mandatory access controls. Neither the subject nor the owner of the object can determine whether access is granted. Typically, the system mechanism will check information associated with both the subject and the object to determine whether the subject should access the object. Rules describe the conditions under which access is allowed.

EXAMPLE: The law allows a court to access driving records without the owners' permission. This is a mandatory control, because the owner of the record has no control over the court's accessing the information.
An *originator controlled access control* (ORCON or ORGCON) bases access on the creator of an object (or the information it contains).

5 COMPUTER SECURITY PROGRAM MANAGEMENT

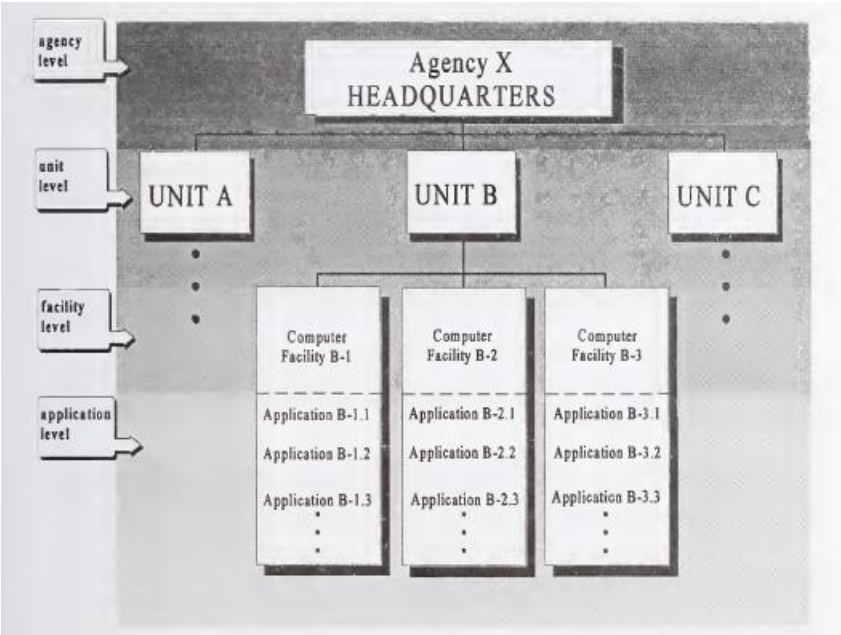
Computers and the information they process are critical to many organizations' ability to perform their mission and business functions. It therefore makes sense that executives view computer security as a management issue and seek to protect their organization's computer resources as they would any other valuable asset. To do this effectively requires developing of a comprehensive management approach.

This chapter presents an organizationwide approach to computer security and discusses its important management function." Because Information Resources," management styles, and culture, it is not security program. However, this chapter does describe some of the features and issues common to many organizations.

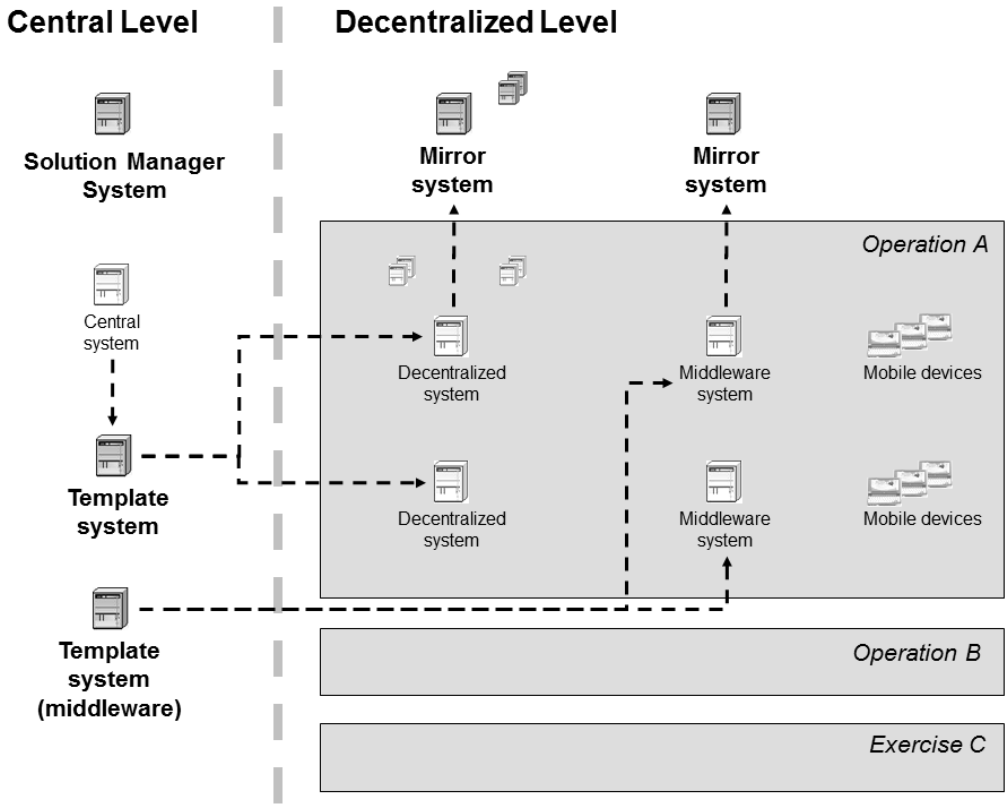
5.1 *Structure of a Computer Security Program*

Many computer-security programs that are distributed throughout the organization have different elements performing various functions. While this approach has benefits, the distribution of the computer security function in many organizations is haphazard, usually based upon history (i.e., who was available in the organization to do what when the need arose). Ideally, the distribution of computer security functions should result from a planned and integrated management philosophy.

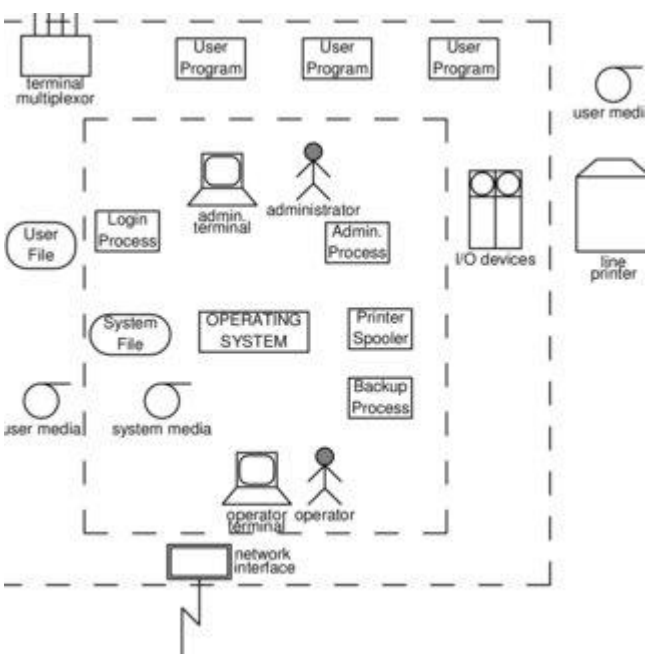
Managing computer security at multiple levels brings many benefits. Each level contributes to the overall computer security program with different types of expertise, authority, and resources. In general, higher-level officials (such as those at the headquarters or unit levels in the agency described above) better understand the organization as a whole and have more authority. On the other hand, lower-level officials (at the computer facility and applications levels) are more familiar with the specific requirements, both technical and procedural, and problems of the systems and the users. The levels of computer security program management should be complementary; each can help the other be more effective.



Since many organizations have at least two levels of computer security management, this chapter divides computer security program management into two levels: the central level and the system level. (Each organization, though, may have its own unique structure.) The central computer security program can be used to address the overall management of computer security within an organization or a major component of an organization. The system-level computer security program addresses the management of computer security for a particular system.

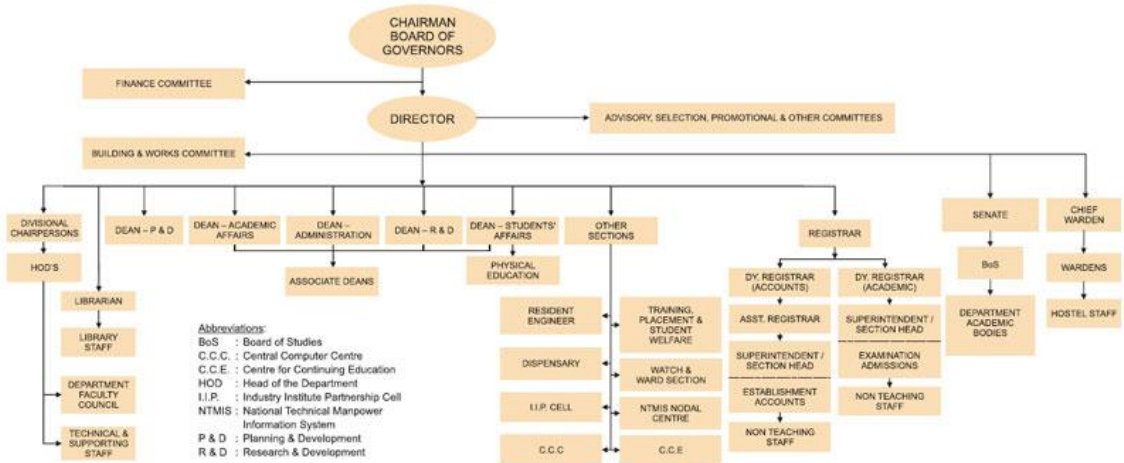


Each member of a system hold a role in the organization but also in the system. Administrators, users, executives and so on should access to the system with different permissions granted



Also it members of the organization would have different levels of access to comply with security policies. Imagine a complex organization as in the next

picture



5.2 *Central Computer Security Programs*

The purpose of a central computer security program is to address the overall management of computer security within an organization. In the federal government, the organization could consist of a department, agency, or other major operating unit.

As with the management of all resources, central computer security management can be performed in many practical and cost-effective ways. The importance of sound management cannot be overemphasized. There is also a downside to centrally managed computer security programs. Specifically, they present greater risk that errors in judgement will be more widely propagated throughout the organization. As they strive to meet their objectives, managers need to consider the full impact of available options when establishing their computer security programs.

5.2.1 Benefits of Central Computer Security Programs

A central security program should provide two quite distinct types of benefits:

Increased efficiency and economy of security throughout the organization, and the ability to provide centralized enforcement and oversight.

It should be agreed to perform organization information management activities in an efficient, effective, and economical manner... .

Organizations shall assure an adequate level of security for all agency

automated information systems, whether maintained in-house or commercially.

5.2.2 Efficient, Economic Coordination of Information

A central computer security program helps to coordinate and manage effective use of security related resources throughout the organization. The most important of these resources are normally information financial resources.

Sound and timely information is necessary for managers to accomplish their tasks effectively. However, most organizations have trouble collecting information from myriad sources and effectively processing and distributing it within the organization. This section discusses some of the sources and efficient uses of computer security information.

Computer security-related information is also available from private and federal professional societies and groups. These groups wiU often provide the information as a public service, although some private groups charge a fee for it. However, even for information that is free or inexpensive, the costs associated with personnel gathering the information can be high.

Internal security-related information, such as which procedures were effective, virus infections, security problems, and solutions, need to be shared within an organization. Often this information is specific to the operating environment and culture of the organization.

A computer security program administered at the organization level can provide a way to collect the internal security-related information and distribute it as needed throughout the organization. Sometimes an organization can also share this information with external groups.

Another use of an effective conduit of information is to increase the central computer security program's ability to influence external and internal policy decisions. If the central computer security program office can represent the entire organization, then its advice is more likely to be heeded by upper management and external organizations. However, to be effective, there should be excellent communication between the system-level computer security programs and the organization level. For example, if an organization were considering consolidating its mainframes into one site (or considering distributing the processing currently done at one site), personnel at the central program could provide initial opinions about the security implications.

However, to speak authoritatively, central program personnel would have to actually know the security impacts of the proposed change - information that would have to be obtained from the system level computer security program.

Personnel at the central computer security program level can also develop their own areas of expertise. For example, they could sharpen their skills could in contingency planning and risk analysis to help the entire organization perform these vital security functions.

Besides allowing an organization to share expertise and, therefore, save money, a central computer security program can use its position to consolidate requirements so the organization can negotiate discounts based on volume purchasing of security hardware and software. It also facilitates such activities as strategic planning and organizationwide incident handling and security trend analysis.

5.2.3 Central Enforcement and Oversight

Besides helping an organization improve the economy and efficiency of its computer security program, a centralized program can include an independent evaluation or enforcement fianction to ensure that organizational subunits are cost-effectively securing resources and following applicable policy. While the Office of the Inspector General (OIG) and external organizations, such as the General Accounting Office (GAO), also perform a valuable evaluation role, they operate outside the regular management channels.

There are several reasons for having an oversight function within the regular management channel. First, computer security is an important component in the management of organizational resources. This is a responsibility that cannot be transferred or abandoned. Second, maintaining an internal oversight function allows an organization to find and correct problems without the potential embarrassment of an IG or GAO audit or investigation. Third, the organization may find different problems fi^om those that an outside organization may find. The organization understands its assets, threats, systems, and procedures better than an external organization; additionally, people may have a tendency to be more candid with insiders.

6 A model for computer security

6.1 *Vocabulary*

Adversary (threat agent)
An entity that attacks, or is a threat to, a system.

Attack

An assault on system security that derives from an intelligent threat; that is, an intelligent act that is a deliberate attempt (especially in the sense of a method or technique) to evade security services and violate the security policy of a system.

Countermeasure

An action, device, procedure, or technique that reduces a threat, a vulnerability, or an attack by eliminating or preventing it, by minimizing the harm it can cause, or by discovering and reporting it so that corrective action can be taken.

Risk

An expectation of loss expressed as the probability that a particular threat will exploit a particular vulnerability with a particular harmful result.

Security Policy

A set of rules and practices that specify or regulate how a system or organization provides security services to
protect sensitive and critical system resources.

System Resource (Asset)

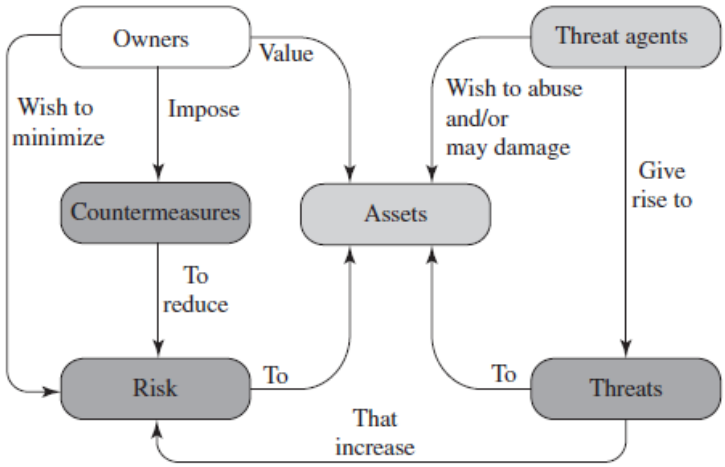
Data contained in an information system; or a service provided by a system; or a system capability, such as processing power or communication bandwidth; or an item of system equipment (i.e., a system component— hardware, firmware, software, or documentation); or a facility that houses system operations and equipment.

Threat

A potential for violation of security, which exists when there is a circumstance, capability, action, or event, that could breach security and cause harm. That is, a threat is a possible danger that might exploit a vulnerability.

Vulnerability

A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy.



This concepts such as a **system resource**, or **asset**, that users and owners wish to protect are the center of a security policy.

The assets of a computer system can be categorized as follows:

- **Hardware:** Including computer systems and other data processing, data storage, and data communications devices
- **Software:** Including the operating system, system utilities, and applications.
- **Data:** Including files and databases, as well as security-related data, such as password files.
- **Communication facilities and networks:** Local and wide area network communication links, bridges, routers, and so on.

In the context of security, our concern is with the **vulnerabilities** of system resources. Here we list the following general categories of vulnerabilities of a computer system or network asset:

- It can be **corrupted**, so that it does the wrong thing or gives wrong answers. For example, stored data values may differ from what they should be because they have been improperly modified.
- It can become **leaky**. For example, someone who should not have access to some or all of the information available through the network obtains such access.
- It can become **unavailable** or very slow. That is, using the system or network becomes impossible or impractical.

These three general types of vulnerability correspond to the concepts of integrity, confidentiality, and availability, enumerated earlier in this section.

Corresponding to the various types of vulnerabilities to a system resource are **threats** that are capable of exploiting those vulnerabilities. A threat represents a potential security harm to an asset. An **attack** is a threat that is carried out (threat action) and, if successful, leads to an undesirable violation of security, or threat consequence.

The agent carrying out the attack is referred to as an attacker, or **threat agent**.

We can distinguish two types of attacks:

- **Active attack**: An attempt to alter system resources or affect their operation.
- **Passive attack**: An attempt to learn or make use of information from the system that does not affect system resources.

We can also classify attacks based on the origin of the attack:

- **Inside attack**: Initiated by an entity inside the security perimeter (an " insider"). The insider is authorized to access system resources but uses them in a way not approved by those who granted the authorization.
- **Outside attack:** Initiated from outside the perimeter, by an unauthorized or illegitimate user of the system (an "outsider"). On the Internet, potential outside attackers range from amateur pranksters to organized criminals, international terrorists, and hostile governments.

Finally, a **countermeasure** is any means taken to deal with a security attack. Ideally, a countermeasure can be devised to **prevent** a particular type of attack from succeeding. When prevention is not possible, or fails in some instance, the goal is to **detect** the attack and then **recover** from the effects of the attack.

A **countermeasure** may itself introduce new vulnerabilities. In any case, residual vulnerabilities may remain after the imposition of countermeasures. Such vulnerabilities may be exploited by threat agents representing a residual level of **risk** to the assets. Owners will seek to minimize that risk given other constraints.

6.2 *Threats, Attacks, and Assets*

We now turn to a more detailed look at threats, attacks, and assets. First, we look at the types of security threats that must be dealt with, and then give some examples of the types of threats that apply to different categories of assets.

6.2.1 Threats and Attacks

Table 1.2, based on RFC 4949, describes four kinds of threat consequences and lists the kinds of attacks that result in each consequence.

- ✓ **Unauthorized disclosure** is a threat to confidentiality. The following types of attacks can result in this threat consequence:
- ✓ **Exposure:** This can be deliberate, as when an insider intentionally releases sensitive information, such as credit card numbers, to an outsider. It can also be the result of a human, hardware, or software error, which results in an entity gaining unauthorized knowledge of sensitive data. There have been numerous instances of this, such as universities accidentally posting student confidential information on the Web.

Threat Consequence Threat Action (Attack)	**Threat Action (Attack)**
Unauthorized Disclosure A circumstance or event whereby an entity gains access to data for which the entity is not authorized.	**Exposure:** Sensitive data are directly released to an unauthorized entity. **Interception:** An unauthorized entity directly accesses sensitive data traveling between authorized sources and destinations. **Inference:** A threat action whereby an unauthorized entity indirectly accesses sensitive data (but not necessarily the data contained in the communication) by reasoning from characteristics or by-products of communications. **Intrusion:** An unauthorized entity gains access to sensitive data by circumventing a system's security protections.
Deception A circumstance or event that may result in an authorized entity receiving false data and believing it to be true.	**Masquerade:** An unauthorized entity gains access to a system or performs a malicious act by posing as an authorized entity. **Falsification:** False data deceive an authorized entity. **Repudiation:** An entity deceives another by falsely denying responsibility for an act.
Disruption A circumstance or event that interrupts or prevents the correct operation of system services and functions.	**Incapacitation:** Prevents or interrupts system operation by disabling a system component.

	Corruption: Undesirably alters system operation by adversely modifying system functions or data. **Obstruction:** A threat action that interrupts delivery of system services by hindering system operation.
Usurpation A circumstance or event that results in control of system services or functions by an unauthorized entity.	**Misappropriation:** An entity assumes unauthorized logical or Physical control of a system resource. **Misuse:** Causes a system component to perform a function or Service that is detrimental to system security.

- **Interception:** Interception is a common attack in the context of communications. On a shared local area network (LAN), such as a wireless LAN or a broadcast Ethernet, any device attached to the LAN can receive a copy of packets intended for another device. On the Internet, a determined hacker can gain access to e-mail traffic and other data transfers. All of these situations create the potential for unauthorized access to data.
- **Inference:** An example of inference is known as traffic analysis, in which an adversary is able to gain information from observing the pattern of traffic on a network, such as the amount of traffic between particular pairs of hosts on the network. Another example is the inference of detailed information from a database by a user who has only limited access; this is accomplished by repeated queries whose combined results enable inference.
- **Intrusion:** An example of intrusion is an adversary gaining unauthorized access to sensitive data by overcoming the system's access control protections.

Deception is a threat to either system integrity or data integrity. The following types of attacks can result in this threat consequence:

- **Masquerade:** One example of masquerade is an attempt by an unauthorized user to gain access to a system by posing as an authorized user; this could happen if the unauthorized user has learned another user's logon ID and password.Another example is malicious logic, such as a Trojan horse, that appears to perform a useful or desirable function but actually gains unauthorized access to system resources or tricks a user into executing other malicious logic.
- **Falsification:** This refers to the altering or replacing of valid data or the introduction of false data into a file or database. For example, a student may alter his or her grades on a school database.
- **Repudiation:** In this case, a user either denies sending data or a user denies receiving or possessing the data.

Disruption is a threat to availability or system integrity. The following types of attacks can result in this threat consequence:

- **Incapacitation:** This is an attack on system availability. This could occur as a result of physical destruction of or damage to system hardware. More typically, malicious software, such as Trojan horses, viruses, or worms, could operate in such a way as to disable a system or some of its services.
- **Corruption:** This is an attack on system integrity. Malicious software in this context could operate in such a way that system resources or services function in an unintended manner. Or a user could gain unauthorized access to a system and modify some of its functions. An example of the latter is a user placing backdoor logic in the system to provide subsequent access to a system and its resources by other than the usual procedure.
- **Obstruction:** One way to obstruct system operation is to interfere with communications by disabling communication links or altering communication control information. Another way is to overload the system by placing excess burden on communication traffic or processing resources.

Usurpation is a threat to system integrity. The following types of attacks can result in this threat consequence:

- **Misappropriation:** This can include theft of service. An example is a distributed denial of service attack, when malicious software is installed on a number of hosts to be used as platforms to launch traffic at a target host. In this case, the malicious software makes unauthorized use of processor and operating system resources.
- **Misuse:** Misuse can occur by means of either malicious logic or a hacker that has gained unauthorized access to a system. In either case, security functions can be disabled or thwarted.

Threats and Assets

The assets of a computer system can be categorized as hardware, software, data, and communication lines and networks. In this subsection, we briefly describe these four categories and relate these to the concepts of integrity, confidentiality, and availability introduced in previous sections

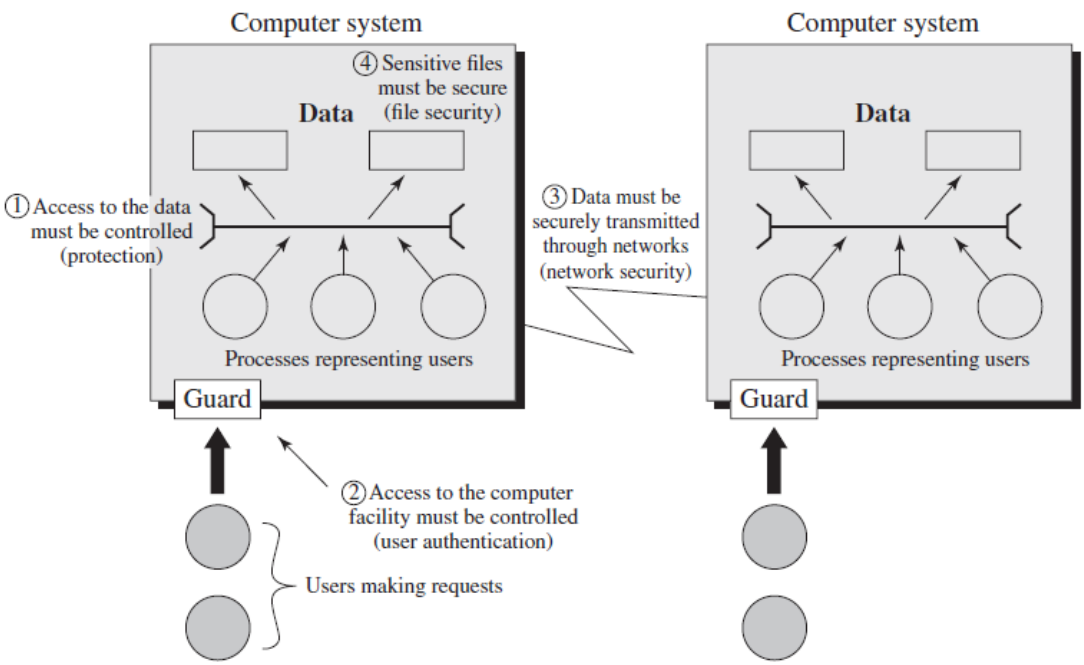
Hardware

A major threat to computer system hardware is the threat to availability.

Hardware is the most vulnerable to attack and the least susceptible to automated controls. Threats include accidental and deliberate damage to equipment as well as theft. The proliferation of personal computers and workstations and the widespread use of LANs increase the potential for losses in this area. Theft of CD-ROMs and DVDs can lead to loss of confidentiality. Physical and administrative security measures are needed to deal with these threats.

Software

Software includes the operating system, utilities, and application programs. A key threat to software is an attack on availability. Software, especially application software, is often easy to delete. Software can also be altered or damaged to render it useless. Careful software configuration management, which includes making backups of the most recent version of software, can maintain high availability. A more difficult problem to deal with is software modification that results in a program that still functions but that behaves differently than before, which is a threat to integrity/authenticity. Computer viruses and related attacks fall into this category. A final problem is protection against software piracy. Although certain countermeasures are available, by and large the problem of unauthorized copying of software has not been solved.



	Availability	Confidentiality	Integrity
Hardware	Equipment is stolen or disabled, thus denying service.	An unencrypted CD-ROM or DVD is stolen.	
Software	Programs are deleted, denying access to users.	An unauthorized copy of software is made.	A working program is modified, either to cause it to fail during execution or to cause

			it to do some unintended task.
Data	Files are deleted, denying access to users.	An unauthorized read of data is performed. An analysis of statistical data reveals underlying data.	Existing files are modified or new files are fabricated.
Communication Lines and Networks	Messages are destroyed or deleted. Communication lines or networks are rendered unavailable.	Messages are read. The traffic pattern of messages is observed.	Messages are modified, delayed, reordered, or duplicated. False messages are fabricated.

Data

Hardware and software security are typically concerns of computing center professionals or individual concerns of personal computer users. A much more widespread problem is data security, which involves files and other forms of data controlled by individuals, groups, and business organizations.

Security concerns with respect to data are broad, encompassing availability, secrecy, and integrity. In the case of availability, the concern is with the destruction of data files, which can occur either accidentally or maliciously.

The obvious concern with secrecy is the unauthorized reading of data files or databases, and this area has been the subject of perhaps more research and effort than any other area of computer security. A less obvious threat to secrecy involves the analysis of data and manifests itself in the use of so-called statistical databases, which provide summary or aggregate information. Presumably, the existence of aggregate information does not threaten the privacy of the individuals involved.

However, as the use of statistical databases grows, there is an increasing potential for disclosure of personal information. In essence, characteristics of constituent individuals may be identified through careful analysis. For example, if one table records the aggregate of the incomes of respondents A, B, C, and D and another records the aggregate of the incomes of A, B, C, D, and E, the difference between the two aggregates would be the income of E. This problem is exacerbated by the increasing desire to combine data sets. In many cases, matching several sets of data for consistency at different levels of aggregation requires access to individual units. Thus, the individual units, which are the subject of privacy concerns, are available at various stages in the processing of data sets.

Finally, data integrity is a major concern in most installations. Modifications to data files can have consequences ranging from minor to disastrous.

Communication Lines and Networks

Network security attacks can be classified as *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 are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the attacker is to obtain information that is being transmitted. Two types of passive attacks are the release of message contents and traffic analysis.

- ✓ The **release of message contents** is easily understood. 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.

- ✓ A second type of passive attack, **traffic analysis**, is subtler. 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 involve some modification of the data stream or the creation of a false stream and can be subdivided into four categories: replay, masquerade, modification of messages, and denial of service.

- ✓ **Replay** involves the passive capture of a data unit and its subsequent retransmission to produce an unauthorized effect.
- ✓ A **masquerade** takes place when one entity pretends to be a different entity. A masquerade attack usually includes one of the other forms of active attack. For example, authentication sequences can be captured and replayed after a valid authentication sequence has taken place, thus enabling an authorized entity with few privileges to obtain extra privileges by impersonating an entity that has those privileges.
- ✓ **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. For example, a message stating, “Allow John Smith to read confidential file accounts” is modified to say, “Allow Fred Brown to read confidential file accounts.”

- ✓ **The denial of service** prevents or inhibits the normal use or management of communication facilities. This attack may have a specific target; for example, an entity may suppress all messages directed to a particular destination (e.g., the security audit service). Another form of service denial is 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 to do so would require continouos physical protection of all communication facilities and paths. Instead, the goal is to detect them and to recover from any disruption or delays caused by them. Because the detection has a deterrent effect, it may also contribute to prevention.

6.3 *Security Functional Requirements*

There are a number of ways of classifying and characterizing the countermeasures that may be used to reduce vulnerabilities and deal with threats to system assets.

It will be useful for the presentation in the remainder of the book to look at several approaches, which we do in this and the next two sections. In this section, we view countermeasures in terms of functional requirements, and we follow the classification defined in FIPS 200 (*Minimum Security Requirements for Federal Information and Information Systems*). This standard enumerates 17 security-related areas with regard to protecting the confidentiality, integrity, and availability of information systems and the information processed, stored, and transmitted by those systems.

The areas are defined in next section.
The requirements listed in FIPS 200 encompass a wide range of countermeasures to security vulnerabilities and threats. Roughly, we can divide these countermeasures into two categories: those that require computer security technical measures (covered in this book in Parts One and Two), either hardware or software, or both; and those that are fundamentally management issues (covered in Part Three).

Each of the functional areas may involve both computer security technical measures and management measures. Functional areas that primarily require computer security technical measures include access control, identification and authentication, system and communication protection, and system and information integrity. Functional areas that primarily involve management controls and procedures include awareness and

training; audit and accountability; certification, accreditation, and security assessments; contingency planning; maintenance; physical and environmental protection; planning; personnel security; risk assessment; and systems and services acquisition. Functional areas that overlap computer security technical
measures and management controls include configuration management, incident response, and media protection.

Note that the majority of the functional requirements areas in FIPS 200 are either primarily issues of management or at least have a significant management component, as opposed to purely software or hardware solutions. This may be new to some readers and is not reflected in many of the books on computer and information security. But as one computer security expert observed, "If you think technology can solve your security problems, then you don't understand the problems and you don't technology" [SCHN00]. This book reflects the need to combine technical and managerial approaches to achieve effective computer security.
FIPS 200 provides a useful summary of the principal areas of concern, both technical and managerial, with respect to computer security. This book attempts to
cover all of these areas.

6.3.1 Security requirements

Access Control: Limit information system access to authorized users, processes acting on behalf of authorized users, or devices (including other information systems) and to the types of transactions and functions that authorized users are permitted to exercise.

Awareness and Training: (i) Ensure that managers and users of organizational information systems are made aware of the security risks associated with their activities and of the applicable laws, regulation, and policies related to the security of organizational information systems; and (ii) ensure that personnel are adequately trained to carry out their assigned information security-related duties and responsibilities.

Audit and Accountability: (i) Create, protect, and retain information system audit records to the extent needed to enable the monitoring, analysis, investigation, and reporting of unlawful, unauthorized, or inappropriate information system activity; and (ii) ensure that the actions of individual information system users can be uniquely traced to those users so they can be held accountable for their actions.

Certification, Accreditation, and Security Assessments: (i) Periodically assess the security controls in organizational information systems to determine if the controls are effective in their application; (ii) develop and

implement plans of action designed to correct deficiencies and reduce or eliminate vulnerabilities in organizational information systems; (iii) authorize the operation of organizational information systems and any associated information system connections; and (iv) **monitor information system security controls on an ongoing basis to ensure the continued effectiveness of the controls.**

Configuration Management: **(**i) Establish and maintain baseline configurations and inventories of organizational information systems (including hardware, software, firmware, and documentation) throughout the respective system development life cycles; and (ii) establish and enforce security configuration settings for information technology products employed in organizational information systems.

Contingency Planning: Establish, maintain, and implement plans for emergency response, backup operations, and postdisaster recovery for organizational information systems to ensure the availability of critical information resources and continuity of operations in emergency situations.

Identification and Authentication: Identify information system users, processes acting on behalf of users, or devices, and authenticate (or verify) the identities of those users, processes, or devices, as a prerequisite to allowing access to organizational information systems.

Incident Response: (i) Establish an operational incident-handling capability for organizational information systems that includes adequate preparation, detection, analysis, containment, recovery, and user-response activities; and (ii) track, document, and report incidents to appropriate organizational officials and/or authorities.

Maintenance: (i) Perform periodic and timely maintenance on organizational information systems; and (ii) provide effective controls on the tools, techniques, mechanisms, and personnel used to conduct information system maintenance.

Media Protection: (i) Protect information system media, both paper and digital; (ii) limit access to information on information system media to authorized users; and (iii) sanitize or destroy information system media before disposal or release for reuse.

Physical and Environmental Protection: (i) Limit physical access to information systems, equipment, and the respective operating environments to authorized individuals; (ii) protect the physical plant and support infrastructure for information systems; (iii) provide supporting utilities for information systems; (iv) protect information systems against environmental hazards; and (v) provide appropriate environmental controls in facilities containing information systems.

Planning: Develop, document, periodically update, and implement security plans for organizational information systems that describe the security controls in place or planned for the information systems and the rules of behavior for individuals accessing the information systems.

Personnel Security: (i) Ensure that individuals occupying positions of responsibility within organizations (including third-party service providers) are trustworthy and meet established security criteria for those positions; (ii) ensure that organizational information and information systems are protected during and after personnel actions such as terminations and transfers; and (iii) employ formal sanctions for personnel failing to comply with organizational security policies and procedures.

Risk Assessment: Periodically assess the risk to organizational operations (including mission, functions, image, or reputation), organizational assets, and individuals, resulting from the operation of organizational information systems and the associated processing, storage, or transmission of organizational information.

Systems and Services Acquisition: (i) Allocate sufficient resources to adequately protect organizational information systems; (ii) employ system development life cycle processes that incorporate information security considerations; (iii) employ software usage and installation restrictions; and (iv) ensure that thirdparty providers employ adequate security measures to protect information, applications, and/or services outsourced from the organization.

System and Communications Protection: (i) Monitor, control, and protect organizational communications (i.e., information transmitted or received by organizational information systems) at the external boundaries and key internal boundaries of the information systems; and (ii) employ architectural designs, software development techniques, and systems engineering principles that promote effective information security within organizational information systems.

System and Information Integrity: (i) Identify, report, and correct information and information system flaws in a timely manner; (ii) provide protection from malicious code at appropriate locations within organizational information systems; and (iii) monitor information system security alerts and advisories and take appropriate actions in response.

6.4 *Fundamental Security Design Principles*

Despite years of research and development, it has not been possible to develop security design and implementation techniques that systematically exclude security flaws and prevent all unauthorized actions. In the absence of such foolproof techniques, it is useful to have a set of widely agreed design principles that can guide the development of protection mechanisms. The National Centers of Academic Excellence in Information Assurance/Cyber Defense, which is jointly sponsored by the U.S. National Security Agency and the U. S. Department of Homeland Security, list the following as fundamental security design principles [NCAE13]:

- ✓ Economy of mechanism
- ✓ Fail-safe defaults
- ✓ Complete mediation
- ✓ Open design
- ✓ Separation of privilege
- ✓ Least privilege
- ✓ Least common mechanism
- ✓ Psychological acceptability
- ✓ Isolation
- ✓ Encapsulation
- ✓ Modularity
- ✓ Layering
- ✓ Least astonishment

The first eight listed principles were first proposed in [SALT75] and have withstood the test of time. In this section, we briefly discuss each principle.

Economy of mechanism means that the design of security measures embodied in both hardware and software should be as simple and small as possible. The motivation for this principle is that relatively simple, small design is easier to test and verify thoroughly. With a complex design, there are many more opportunities for an adversary to discover subtle weaknesses to exploit that may be difficult to spot ahead of time. The more complex the mechanism, the more likely it is to possess exploitable flaws. Simple mechanisms tend to have fewer exploitable flaws and require less maintenance. Furthermore, because configuration management issues are simplified, updating or replacing a simple mechanism becomes a less intensive process. In practice, this is perhaps the most difficult principle to honor. There is a constant demand for new features in both hardware and software, complicating the security design task. The best that can be done is to keep this principle in mind during system design to try to eliminate unnecessary complexity.

Fail-safe default means that access decisions should be based on permission rather than exclusion. That is, the default situation is lack of access, and the protection scheme identifies conditions under which access is permitted. This approach exhibits a better failure mode than the alternative approach, where

the default is to permit access. A design or implementation mistake in a mechanism that gives explicit permission tends to fail by refusing permission, a safe situation that can be quickly detected. On the other hand, a design or implementation mistake in a mechanism that explicitly excludes access tends to fail by allowing access, a failure that may long go unnoticed in normal use. For example, most file access systems work on this principle and virtually all protected services on client/server systems work this way.

Complete mediation means that every access must be checked against the access control mechanism. Systems should not rely on access decisions retrieved from a cache. In a system designed to operate continuously, this principle requires that, if access decisions are remembered for future use, careful consideration be given to how changes in authority are propagated into such local memories. File access systems appear to provide an example of a system that complies with this principle. However, typically, once a user has opened a file, no check is made to see of permissions change. To fully implement complete mediation, every time a user reads a field or record in a file, or a data item in a database, the system must exercise access control. This resource-intensive approach is rarely used.

Open design means that the design of a security mechanism should be open rather than secret. For example, although encryption keys must be secret, encryption algorithms should be open to public scrutiny. The algorithms can then be reviewed by many experts, and users can therefore have high confidence in them.

This is the philosophy behind the National Institute of Standards and Technology (NIST) program of standardizing encryption and hash algorithms, and has led to the widespread adoption of NIST-approved algorithms.

Separation of privilege is defined in [SALT75] as a practice in which multiple privilege attributes are required to achieve access to a restricted resource. A good example of this is multifactor user authentication, which requires the use of multiple techniques, such as a password and a smart card, to authorize a user. The term is also now applied to any technique in which a program is divided into parts that are limited to the specific privileges they require in order to perform a specific task. This is used to mitigate the potential damage of a computer security attack.

One example of this latter interpretation of the principle is removing high privilege operations to another process and running that process with the higher privileges required to perform its tasks. Day-to-day interfaces are executed in a lower privileged process.

Least privilege means that every process and every user of the system should operate using the least set of privileges necessary to perform the task. A good example of the use of this principle is role-based access control, described in Chapter 4.

The system security policy can identify and define the various roles of users or processes. Each role is assigned only those permissions needed to perform its functions.

Each permission specifies a permitted access to a particular resource (such as read and write access to a specified file or directory, and connect access to a given host and port). Unless permission is granted explicitly, the user or process should not be able to access the protected resource. More generally, any access control system should allow each user only the privileges that are authorized for that user. There is also a temporal aspect to the least privilege principle. For example, system programs or administrators who have special privileges should have those privileges only when necessary; when they are doing ordinary activities the privileges should be withdrawn. Leaving them in place just opens the door to accidents.

Least common mechanism means that the design should minimize the functions shared by different users, providing mutual security. This principle helps reduce the number of unintended communication paths and reduces the amount of hardware and software on which all users depend, thus making it easier to verify if there are any undesirable security implications.

Psychological acceptability implies that the security mechanisms should not interfere unduly with the work of users, while at the same time meeting the needs of those who authorize access. If security mechanisms hinder the usability or accessibility of resources, users may opt to turn off those mechanisms. Where possible, security mechanisms should be transparent to the users of the system or at most introduce minimal obstruction. In addition to not being intrusive or burdensome, security procedures must reflect the user's mental model of protection. If the protection procedures do not make sense to the user or if the user must translate his image of protection into a substantially different protocol, the user is likely to make errors.

Isolation is a principle that applies in three contexts. First, public access systems should be isolated from critical resources (data, processes, etc.) to prevent disclosure or tampering. In cases where the sensitivity or criticality of the information is high, organizations may want to limit the number of systems on which that data are stored and isolate them, either physically or logically. Physical isolation may include ensuring that no physical connection exists between an organization's public access information resources and an organization's critical information. When implementing logical isolation solutions, layers of security services and mechanisms should be established between public systems and secure systems responsible for protecting critical resources. Second, the processes and files of individual users should be isolated from one another except where it is explicitly desired. All modern operating systems provide facilities for such isolation, so that individual users have separate, isolated process space, memory space, and file space, with protections for preventing unauthorized access. And finally, security mechanisms should be isolated in the sense of preventing access to those

mechanisms. For example, logical access control may provide a means of isolating cryptographic software from other parts of the host system and for protecting cryptographic software from tampering and the keys from replacement or disclosure.

Encapsulation can be viewed as a specific form of isolation based on objectoriented functionality. Protection is provided by encapsulating a collection of procedures and data objects in a domain of its own so that the internal structure of a data object is accessible only to the procedures of the protected subsystem and the procedures may be called only at designated domain entry points.

Modularity in the context of security refers both to the development of security functions as separate, protected modules and to the use of a modular architecture for mechanism design and implementation. With respect to the use of separate security modules, the design goal here is to provide common security functions and services, such as cryptographic functions, as common modules. For example, numerous protocols and applications make use of cryptographic functions. Rather than implementing such functions in each protocol or application, a more secure design is provided by developing a common cryptographic module that can be invoked by numerous protocols and applications. The design and implementation effort can then focus on the secure design and implementation of a single cryptographic module, including mechanisms to protect the module from tampering. With respect to the use of a modular architecture, each security mechanism should be able to support migration to new technology or upgrade of new features without requiring an entire system redesign. The security design should be modular so that individual parts of the security design can be upgraded without the requirement to modify the entire system.

Layering refers to the use of multiple, overlapping protection approaches addressing the people, technology, and operational aspects of information systems. By using multiple, overlapping protection approaches, the failure or circumvention of any individual protection approach will not leave the system unprotected.

We will see throughout this book that a layering approach is often used to provide multiple barriers between an adversary and protected information or services. This technique is often referred to as *defense in depth*.

Least astonishment means that a program or user interface should always respond in the way that is least likely to astonish the user. For example, the mechanism for authorization should be transparent enough to a user that the user has a good intuitive understanding of how the security goals map to the provided security mechanism.

intuitive
understanding of how the security goals map to the provided security mechanism.

6.5 *Attack Surfaces And Attack Trees*

In previous sections, we provided an overview of the spectrum of security threats and attacks facing computer and network systems. Section 8.1 goes into more detail about the nature of attacks and the types of adversaries that present security threats.

In this section, we elaborate on two concepts that are useful in evaluating and classifying threats: attack surfaces and attack trees.

6.5.1 Attack Surfaces

An attack surface consists of the reachable and exploitable vulnerabilities in a system Examples of attack surfaces are the following:

- Open ports on outward facing Web and other servers, and code listening on those ports
- Services available on the inside of a firewall
- Code that processes incoming data, email, XML, office documents, and industryspecific custom data exchange formats
- Interfaces, SQL, and Web forms
- An employee with access to sensitive information vulnerable to a social engineering attack

Attack surfaces can be categorized in the following way:

- Network attack surface: This category refers to vulnerabilities over an enterprise network, wide-area network, or the Internet. Included in this category are network protocol vulnerabilities, such as those used for a denial-of-service attack, disruption of communications links, and various forms of intruder attacks.
- Software attack surface: This refers to vulnerabilities in application, utility, or operating system code. A particular focus in this category is Web server software.
- Human attack surface: This category refers to vulnerabilities created by personnel or outsiders, such as social engineering, human error, and trusted insiders.

An attack surface analysis is a useful technique for assessing the scale and severity of threats to a system. A systematic analysis of points of vulnerability

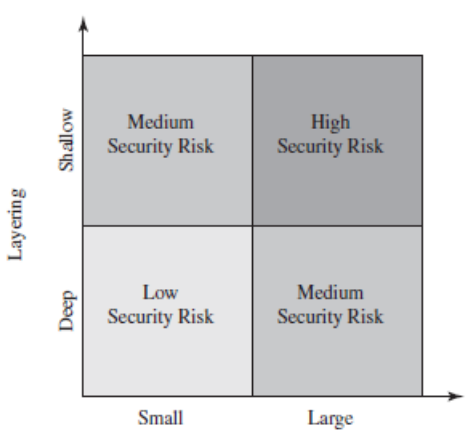
makes developers and security analysts aware of where security mechanisms are required. Once an attack surface is defined, designers may be able to find ways to make the surface smaller, thus making the task of the adversary more difficult. The attack surface also provides guidance on setting priorities for testing, strengthening security measures, or modifying the service or application.

6.5.2 Attack Trees

An attack tree is a branching, hierarchical data structure that represents a set of potential techniques for exploiting security vulnerabilities
. The security incident that is the goal of the attack is represented as the root node of the tree, and the ways that an attacker could reach that goal are iteratively and incrementally represented as branches and subnodes of the tree. Each subnode defines a subgoal, and each subgoal may have its own set of further subgoals, etc. The final nodes on the paths outward from the root, i.e., the leaf nodes, represent different ways to initiate an attack. Each node other than a leaf is either an AND-node or an OR-node. To achieve the goal represented by an AND-node, the subgoals represented by all of that node's subnodes must be achieved; and for an OR-node, at least one of the subgoals must be achieved. Branches can be labeled with values representing difficulty, cost, or other attack attributes, so that alternative attacks can be compared.

The motivation for the use of attack trees is to effectively exploit the information available on attack patterns. Organizations such as CERT publish security advisories that have enabled the development of a body of knowledge about both general attack strategies and specific attack patterns. Security analysts can use the attack tree to document security attacks in a structured form that reveals key vulnerabilities.

The attack tree can guide both the design of systems and applications, and the choice and strength of countermeasures.



which communications or processing resources are consumed so that they are unavailable to legitimate users.

• **Response:** If security mechanisms detect an ongoing attack, such as a denial-of-service attack, the system may be able to respond in such a way as to halt the attack and prevent further damage.
• **Recovery:** An example of recovery is the use of backup systems, so that if data integrity is compromised, a prior, correct copy of the data can be reloaded.

Assurance and Evaluation

Those who are "consumers" of computer security services and mechanisms (e.g., system managers, vendors, customers, and end users) desire a belief that the security measures in place work as intended. That is, security consumers want to feel that the security infrastructure of their systems meet security requirements and enforce security policies. These considerations bring us to the concepts of assurance and evaluation.

The NIST Computer Security Handbook defines **assurance** as the degree of confidence one has that the security measures, both technical and operational, work as intended to protect the system and the information it processes. This encompasses both system design and system implementation. Thus, assurance deals with the questions, "Does the security system design meet its requirements?" and "Does the security system implementation meet its specifications?"

Note that assurance is expressed as a degree of confidence, not in terms of a formal proof that a design or implementation is correct. The state of the art in proving designs and implementations is such that it is not possible to provide absolute proof.

Much work has been done in developing formal models that define requirements and characterize designs and implementations, together with logical and mathematical techniques for addressing these issues. But assurance is still a matter of degree.

Evaluation is the process of examining a computer product or system with respect to certain criteria. Evaluation involves testing and may also involve formal analytic or mathematical techniques. The central thrust of work in this area is the development of evaluation criteria that can be applied to any security system (encompassing security services and mechanisms) and that are broadly supported for making product comparisons.

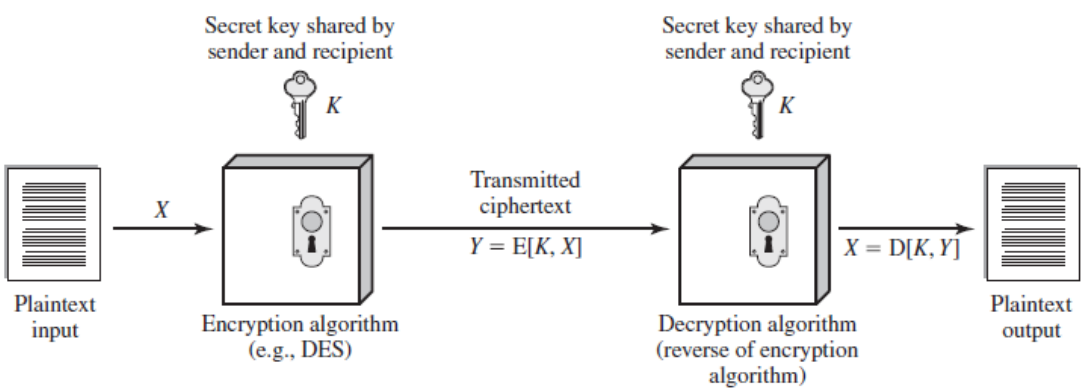
7 Cryptographic Tools

7.1 *Confidentiality with Symmetric Encryption*

The universal technique for providing confidentiality for transmitted or stored data is symmetric encryption. This section introduces the basic concept of symmetric encryption. This is followed by an overview of the two most important symmetric encryption algorithms: the Data Encryption Standard (DES) and the Advanced Encryption Standard (AES), which are block encryption algorithms. Finally, this section introduces the concept of symmetric stream encryption algorithms.

7.1.1 Symmetric Encryption

Symmetric encryption, also referred to as conventional encryption or single-key encryption, was the only type of encryption in use prior to the introduction of public-key encryption in the late 1970s. Countless individuals and groups, from Julius Caesar to the German U-boat force to present-day diplomatic, military, and commercial users, have used symmetric encryption for secret communication. It remains the more widely used of the two types of encryption.



A symmetric encryption scheme has five ingredients:

- **Plaintext: This is the original message or data that is fed into the algorithm as input.**
- **Encryption algorithm: The encryption algorithm performs various substitutions and transformations on the plaintext.**
- **Secret key: The secret key is also input to the encryption algorithm. The exact substitutions and transformations performed by the algorithm depend on the key.**
- **Ciphertext:** This is the scrambled message produced as output. It depends on the plaintext and the secret key. For a given message, two different keys will produce two different ciphertexts.

- **Decryption algorithm:** This is essentially the encryption algorithm run in reverse. It takes the ciphertext and the secret key and produces the original plaintext.

There are two requirements for secure use of symmetric encryption:

1) We need a strong encryption algorithm. At a minimum, we would like the Algorithm to be such that an opponent who knows the algorithm and has Access to one or more ciphertexts would be unable to decipher the ciphertext or figure out the key. This requirement is usually stated in a stronger form:The opponent should be unable to decrypt ciphertext or discover the key even if he or she is in possession of a number of ciphertexts together with the plaintext that produced each ciphertext.
2) Sender and receiver must have obtained copies of the secret key in a secure fashion and must keep the key secure. If someone can discover the key and knows the algorithm, all communication using this key is readable.

There are two general approaches to attacking a symmetric encryption scheme. The first attack is known as **cryptanalysis**. Cryptanalytic attacks rely on the nature of the algorithm plus perhaps some knowledge of the general characteristics of the plaintext or even some sample plaintext-ciphertext pairs. This type of attack exploits the characteristics of the algorithm to attempt to deduce a specific plaintext or to deduce the key being used. If the attack succeeds in deducing the key, the effect is catastrophic: All future and past messages encrypted with that key are compromised.

The second method, known as the **brute-force attack**, is to try every possible key on a piece of ciphertext until an intelligible translation into plaintext is obtained.

On average, half of all possible keys must be tried to achieve success. That is, if there are *x* different keys, on average an attacker would discover the actual key after *x*/2 tries. It is important to note that there is more to a brute-force attack than simply running through all possible keys. Unless known plaintext is provided, the analyst must be able to recognize plaintext as plaintext. If the message is just plain text in English, then the result pops out easily, although the task of recognizing English would have to be automated. If the text message has been compressed before encryption, then recognition is more difficult. And if the message is some more general type of data, such as a numerical file, and this has been compressed, the problem becomes even more difficult to automate. Thus, to supplement the brute-force approach, some degree of knowledge about the expected plaintext is needed, and some means of automatically distinguishing plaintext from garble is also needed.

Symmetric Block Encryption Algorithms

The most commonly used symmetric encryption algorithms are block ciphers. A block cipher processes the plaintext input in fixed-size blocks and produces a block of ciphertext of equal size for each plaintext block. The algorithm processes longer plaintext amounts as a series of fixed-size blocks. The most important symmetric algorithms, all of which are block ciphers, are the Data Encryption Standard (DES), triple DES, and the Advanced Encryption Standard (AES); see Table. This subsection provides an overview of these algorithms.

Concerns about the strength of DES fall into two categories: concerns about the algorithm itself and concerns about the use of a 56-bit key. The first concern refers to the possibility that cryptanalysis is possible by exploiting the characteristics of the DES algorithm. Over the years, there have been numerous attempts to find and exploit weaknesses in the algorithm, making DES the most-studied encryption algorithm in existence. Despite numerous approaches, no one has so far reported a fatal weakness in DES.

A more serious concern is key length. With a key length of 56 bits, there are 256 possible keys, which is approximately 7.2 * 1016 keys. Given the speed of commercial, off-the-shelf processors this key length is woefully inadequate. A paper from Seagate Technology suggests that a rate of one billion (109) key combinations per second is reasonable for today's multicore computers. Recent offerings confirm this. Both Intel and AMD now offer hardware based instructions to accelerate the use of AES. Tests run on a contemporary multicore Intel machine resulted in an encryption rate of about half a billion encryptions per second. Another recent analysis suggests that with contemporary supercomputer technology, a rate of 1013 encryptions/s is reasonable.

With these results in mind, Table shows how much time is required for a brute-force attack for various key sizes. As can be seen, a single PC can break DES in about a year; if multiple PCs work in parallel, the time is drastically shortened.

And today's supercomputers should be able to find a key in about an hour. Key sizes of 128 bits or greater are effectively unbreakable using simply a bruteforce approach. Even if we managed to speed up the attacking system by a factor of 1 trillion (1012), it would still take over 100,000 years to break a code using a 128-bit key.

Fortunately, there are a number of alternatives to DES, the most important of which are triple DES and AES, discussed in the remainder of this section.

Key size (bits)	Cipher	Number of Alternative Keys	Time Required at 10^9 decryptions/μs	Time Required at 10^{13} decryptions/μs
56	DES	$2^{56} \approx 7.2 \times 10^{16}$	$2^{55}\,\mu s = 1.125$ years	1 hour
128	AES	$2^{128} \approx 3.4 \times 10^{38}$	$2^{127}\,\mu s = 5.3 \times 10^{21}$ years	5.3×10^{17} years
168	Triple DES	$2^{168} \approx 3.7 \times 10^{50}$	$2^{167}\,\mu s = 5.8 \times 10^{33}$ years	5.8×10^{29} years
192	AES	$2^{192} \approx 6.3 \times 10^{57}$	$2^{191}\,\mu s = 9.8 \times 10^{40}$ years	9.8×10^{36} years
256	AES	$2^{256} \approx 1.2 \times 10^{77}$	$2^{255}\,\mu s = 1.8 \times 10^{60}$ years	1.8×10^{56} years

Triple DES The life of DES was extended by the use of triple DES (3DES), which involves repeating the basic DES algorithm three times, using either two or three unique keys, for a key size of 112 or 168 bits. 3DES was first standardized for use in financial applications in ANSI standard X9.17 in 1985. 3DES was incorporated as part of the Data Encryption Standard in 1999, with the publication of FIPS PUB 46-3.

3DES has two attractions that assure its widespread use over the next few years. First, with its 168-bit key length, it overcomes the vulnerability to brute-force attack of DES. Second, the underlying encryption algorithm in 3DES is the same as in DES. This algorithm has been subjected to more scrutiny than any other encryption algorithm over a longer period of time, and no effective cryptanalytic attack based on the algorithm rather than brute force has been found. Accordingly, there is a high level of confidence that 3DES is very resistant to cryptanalysis. If security were the only consideration, then 3DES would be an appropriate choice for a standardized encryption algorithm for decades to come.

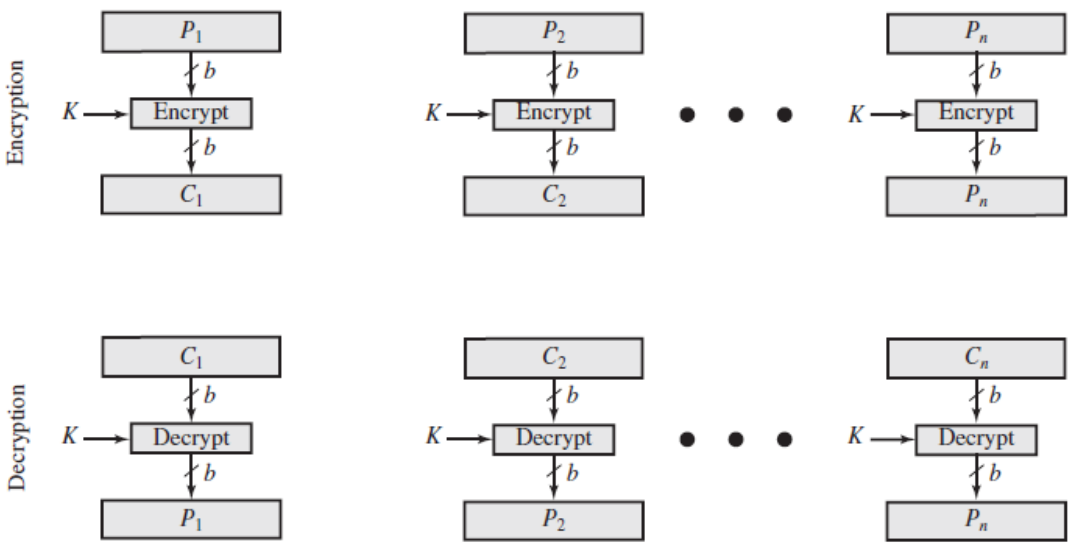
The principal drawback of 3DES is that the algorithm is relatively sluggish in software. The original DES was designed for mid-1970s hardware implementation and does not produce efficient software code. 3DES, which requires three times as many calculations as DES, is correspondingly slower. A secondary drawback is that both DES and 3DES use a 64-bit block size. For reasons of both efficiency and security, a larger block size is desirable.

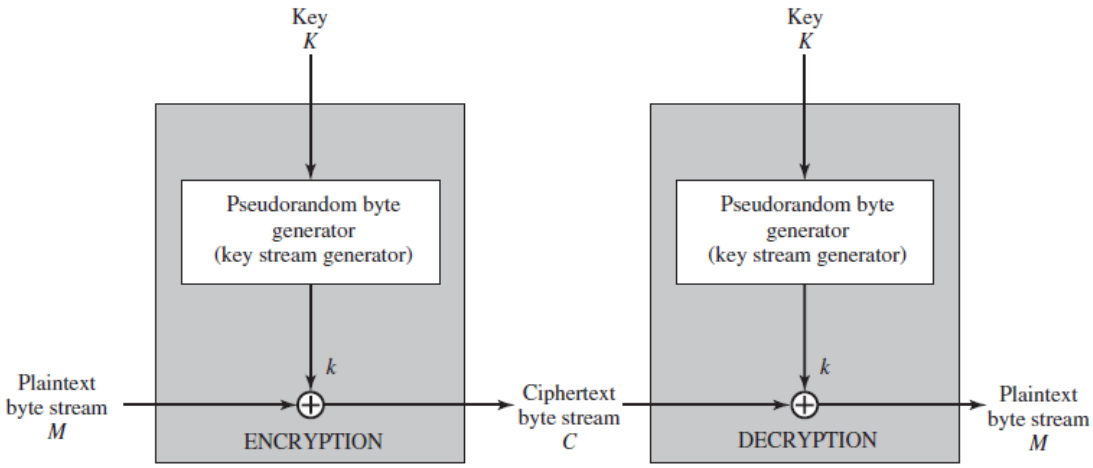
Advanced Encryption Standard Because of its drawbacks, 3DES is not a reasonable candidate for long-term use. As a replacement, NIST in 1997 issued a call for proposals for a new Advanced Encryption Standard (AES), which should have a security strength equal to or better than 3DES and significantly improved efficiency. In addition to these general requirements, NIST specified that AES must be a symmetric block cipher with a block length of 128 bits and support for key lengths of 128, 192, and 256 bits. Evaluation criteria included security, computational efficiency, memory requirements, hardware and software suitability, and flexibility.

In a first round of evaluation, 15 proposed algorithms were accepted. A second round narrowed the field to 5 algorithms. NIST completed its evaluation process and published a final standard (FIPS PUB 197) in November of 2001. NIST

selected Rijndael as the proposed AES algorithm. AES is now widely available in commercial products. AES is described in detail in Chapter 20.

Practical Security Issues Typically, symmetric encryption is applied to a unit of data larger than a single 64-bit or 128-bit block. E-mail messages, network packets, database records, and other plaintext sources must be broken up into a series of fixed-length block for encryption by a symmetric block cipher. The simplest approach to multiple-block encryption is known as electronic codebook (ECB) mode, in which plaintext is handled *b* bits at a time and each block of plaintext is encrypted using the same key. Typically *b* = 64 or *b* = 128. Figure shows the ECB mode. A plain text of length *nb* is divided into *n b*-bit blocks (P1, P2,c,P*n*). Each block is encrypted using the same algorithm and the same encryption key, to produce a sequence of *n b*-bit blocks of ciphertext (*C*1, C2,c,C*n*).





able to exploit regularities in the plaintext to ease the task of decryption. For example, if it is known that the message always starts out with certain predefined fields, then the cryptanalyst may have a number of known plaintext-ciphertext pairs to work with.

To increase the security of symmetric block encryption for large sequences of data, a number of alternative techniques have been developed, called **modes of operation**. These modes overcome the weaknesses of ECB; each mode has its own particular advantages.

A *block cipher* processes the input one block of elements at a time, producing an output block for each input block. A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along. Although block ciphers are far more common, there are certain applications in which a stream cipher is more appropriate. Examples are given subsequently in this book.

A typical stream cipher encrypts plaintext one byte at a time, although a stream cipher may be designed to operate on one bit at a time or on units larger than a byte at a time. Figure is a representative diagram of stream cipher structure. In this structure a key is input to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently random. A pseudorandom stream is one that is unpredictable without knowledge of the input key and which has an apparently random character (see Section 2.5). The output of the generator, called a **keystream**, is combined one byte at a time with the plaintext stream using the bitwise exclusive-OR (XOR) operation.

With a properly designed pseudorandom number generator, a stream cipher can be as secure as a block cipher of comparable key length. The primary advantage of a stream cipher is that stream ciphers are almost always faster and use far less code than do block ciphers. The advantage of a block cipher is that you can reuse keys. For applications that require encryption/decryption of a stream of data, such as over a data communications channel or a browser/Web link, a

stream cipher might be the better alternative. For applications that deal with blocks of data, such as file transfer, e-mail, and database, block ciphers may be more appropriate. However, either type of cipher can be used in virtually any application.

7.1.2 Authentication Using Symmetric Encryption

It would seem possible to perform authentication simply by the use of symmetric encryption. If we assume that only the sender and receiver share a key (which is as it should be), then only the genuine sender would be able to encrypt a message successfully for the other participant, provided the receiver can recognize a valid message.

Furthermore, if the message includes an error-detection code and a sequence number, the receiver is assured that no alterations have been made and that sequencing is proper. If the message also includes a timestamp, the receiver is assured that the message has not been delayed beyond that normally expected for network transit.

In fact, symmetric encryption alone is not a suitable tool for data authentication. To give one simple example, in the ECB mode of encryption, if an attacker reorders the blocks of ciphertext, then each block will still decrypt successfully.

However, the reordering may alter the meaning of the overall data sequence. Although sequence numbers may be used at some level (e.g., each IP packet), it is typically not the case that a separate sequence number will be associated with each *b*-bit block of plaintext. Thus, block reordering is a threat.

7.1.3 Message Authentication without Message Encryption

In this section, we examine several approaches to message authentication that do not rely on message encryption. In all of these approaches, an authentication tag is generated and appended to each message for transmission. The message itself is not encrypted and can be read at the destination independent of the authentication function at the destination.

Because the approaches discussed in this section do not encrypt the message, message confidentiality is not provided. As was mentioned, message encryption by itself does not provide a secure form of authentication. However, it is possible to combine authentication and confidentiality in a single algorithm by encrypting a message plus its authentication tag. Typically, however, message authentication is provided as a separate function from message encryption. Experts suggest three situations in which message authentication without confidentiality is preferable:

1. There are a number of applications in which the same message is broadcast to a number of destinations. Two examples are notification to users that the network is now unavailable, and an alarm signal in a control center. It is cheaper and more reliable to have only one destination responsible for monitoring authenticity.

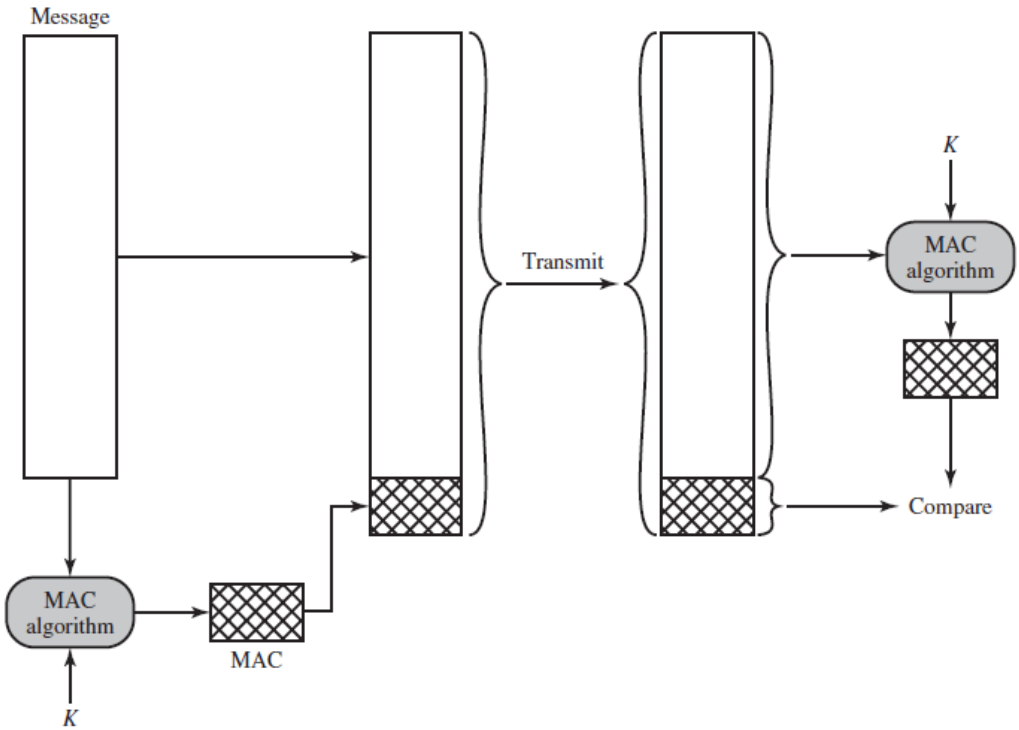
Thus, the message must be broadcast in plaintext with an associated message authentication tag. The responsible system performs authentication. If a violation occurs, the other destination systems are alerted by a general alarm.

2. Another possible scenario is an exchange in which one side has a heavy load and cannot afford the time to decrypt all incoming messages. Authentication is carried out on a selective basis, with messages being chosen at random for checking.

3. Authentication of a computer program in plaintext is an attractive service. The computer program can be executed without having to decrypt it every time, which would be wasteful of processor resources. However, if a message authentication tag were attached to the program, it could be checked whenever assurance is required of the integrity of the program.Thus, there is a place for both authentication and encryption in meeting security requirements.

Message Authentication Code One authentication technique involves the use of a secret key to generate a small block of data, known as a message authentication code, that is appended to the message. This technique assumes that two communicating parties, say A and B, share a common secret key *KAB*. When A has a message to send to B, it calculates the message authentication code as a complex function of the message and the key: MAC*M* = F(*KAB*, *M*).3

The message plus code 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 message authentication code. The received code is compared to the calculated code (Figure). If we assume that only the receiver and the sender know the identity of the secret key, and if the received code matches the calculated code, then:



1) The receiver is assured that the message has not been altered. If an attacker alters the message but does not alter the code, then the receiver's calculation of the code will differ from the received code. Because the attacker is assumed not to know the secret key, the attacker cannot alter the code to correspond fo 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 code.

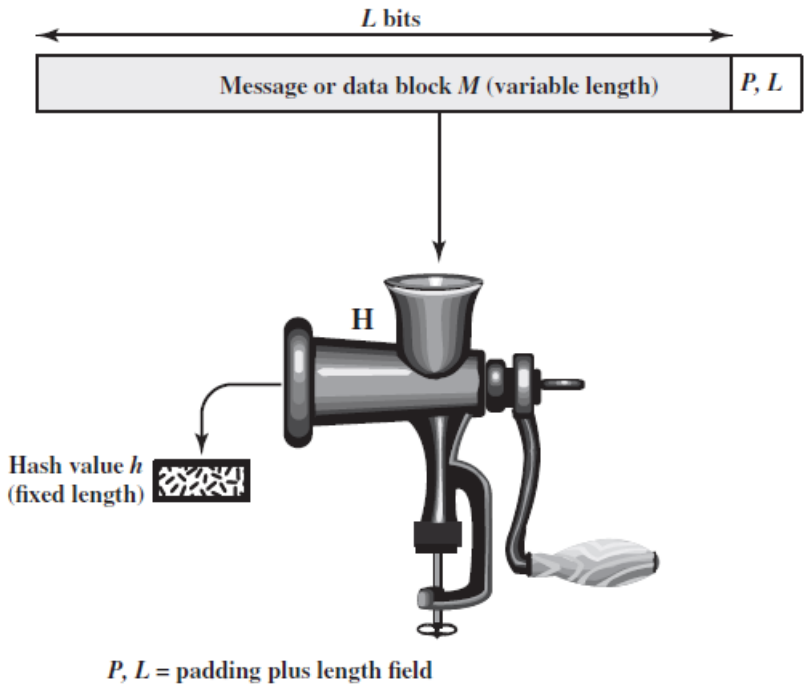
3) If the message includes a sequence number (such as is used with X.25, HDLC, and TCP), then the receiver can be assured of the proper sequence, because an attacker cannot successfully alter the sequence number.

A number of algorithms could be used to generate the code. The NIST specification, FIPS PUB 113, recommends the use of DES. DES is used to generate an encrypted version of the message, and the last number of bits of ciphertext are used as the code. A 16- or 32-bit code is typical.

4) The process just described is similar to encryption. One difference is that the authentication algorithm need not be reversible, as it must for decryption. It turns out that because of the mathematical properties of the authentication function, it is less vulnerable to being broken than encryption.

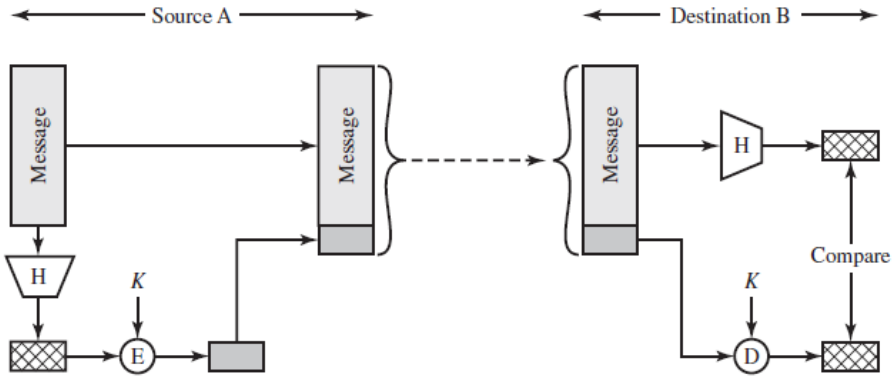
One-Way Hash Function An alternative to the message authentication code is

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 message digest H(*M*) as output (Figure). Typically, the message is padded out to an integer multiple of some fixed length (e.g., 1024 bits) and the padding includes the value of the length of the original message in bits. The length field is a security measure to increase the difficulty for an attacker to produce an alternative message with the same hash value.

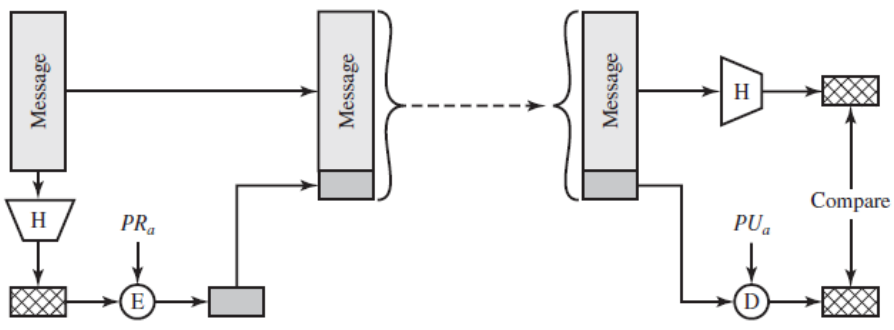


P, L = padding plus length field

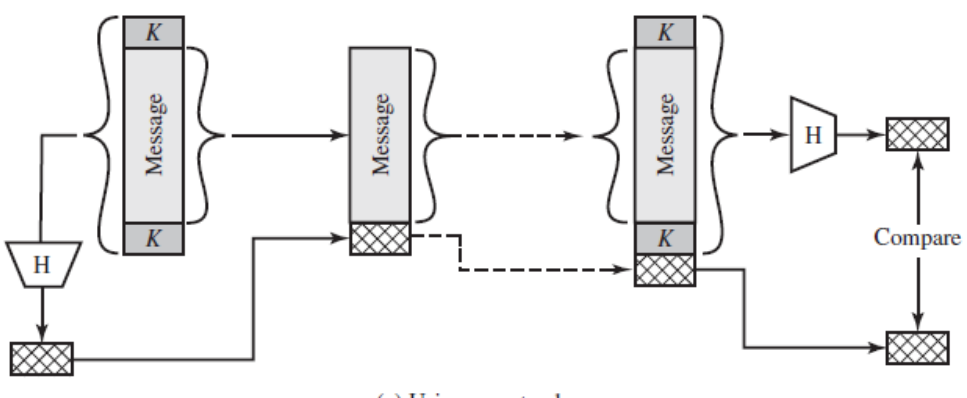
Unlike the MAC, a hash function does not take a secret key as input. Figure illustrates three ways in which the message can be authenticated using a hash function. The message digest can be encrypted using symmetric encryption (Figure); if it is assumed that only the sender and receiver share the encryption key, then authenticity is assured. The message digest can also be encrypted using public-key encryption (Figure); this is explained in a next section. The public-key approach has two advantages: It provides a digital signature as well as message authentication; and it does not require the distribution of keys to communicating parties.



(a) Using symmetric encryption



(b) Using public-key encryption



(c) Using secret value

These two approaches have an advantage over approaches that encrypt the entire message in that less computation is required. But an even more common approach is the use of a technique that avoids encryption altogether. Several reasons for this interest are pointed out:

- Encryption software is quite slow. Even though the amount of data to be encrypted per message is small, there may be a steady stream of messages into and out of a system.
- Encryption hardware costs are nonnegligible. Low-cost chip

implementations of DES are available, but the cost adds up if all nodes in a network must have this capability.
- Encryption hardware is optimized toward large data sizes. For small blocks of data, a high proportion of the time is spent in initialization/invocation overhead.
- An encryption algorithm may be protected by a patent.

Previous figure shows a technique that uses a hash function but no encryption for message authentication. This technique, known as a keyed hash MAC, assumes that two communicating parties, say A and B, share a common secret key *K*.

This secret key is incorporated into the process of generating a hash code. In the approach illustrated in previous Figure, when A has a message to send to B, it calculates the hash function over the concatenation of the secret key and the message:

MDM = H(*K*‥*M*‥*K*).5 It then sends [*M*‥*MDM*] to B. Because B possesses *K*, it can recompute H(*K*‥*M*‥*K*) and verify *MDM*. Because the secret key itself is not sent, it should not be possible for an attacker to modify an intercepted message. As long as the secret key remains secret, it should not be possible for an attacker to generate a false message.

Note that the secret key is used as both a prefix and a suffix to the message. If the secret key is used as either only a prefix or only a suffix, the scheme is less secure.

7.1.4 Public-Key Encryption

Of equal importance to symmetric encryption is public-key encryption, which finds use in message authentication and key distribution.

Public-Key Encryption Structure

Public-key encryption, first publicly proposed by Diffie and Hellman in 1976 is the first truly revolutionary advance in encryption in literally thousands of years.

Public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns, such as are used in symmetric encryption algorithms. More important, public-key cryptography is **asymmetric**, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. The use of two keys has profound consequences in the areas of confidentiality, key distribution, and authentication.

Before proceeding, we should first mention several common misconceptions concerning public-key encryption. One is that public-key encryption is more

secure from cryptanalysis than symmetric encryption. In fact, the security of any encryption scheme depends on (1) the length of the key and (2) the computational work involved in breaking a cipher. There is nothing in principle about either symmetric or public-key encryption that makes one superior to another from the point of view of resisting cryptanalysis.

A second misconception is that public-key encryption is a general-purpose technique that has made symmetric encryption obsolete. On the contrary, because of the computational overhead of current public-key encryption schemes, there seems no foreseeable likelihood that symmetric encryption will be abandoned. Finally, there is a feeling that key distribution is trivial when using public-key encryption, compared to the rather cumbersome handshaking involved with key distribution centers for symmetric encryption. For public-key key distribution, some form of protocol is needed, often involving a central agent, and the procedures involved are no simpler or any more efficient than those required for symmetric encryption.

A public-key encryption scheme has six ingredients (Figure):

- **Plaintext: This is the readable message or data that is fed into the algorithm as input.**
- **Encryption algorithm: The encryption algorithm performs various transformations on the plaintext.**
- **Public and private key: This is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the encryption algorithm depend on the public or private key that is provided as input.**
- **Ciphertext: This is the scrambled message produced as output. It depends on the plaintext and the key. For a given message, two different keys will produce two different ciphertexts.**
- **Decryption algorithm: This algorithm accepts the ciphertext and the matching key and produces the original plaintext.**

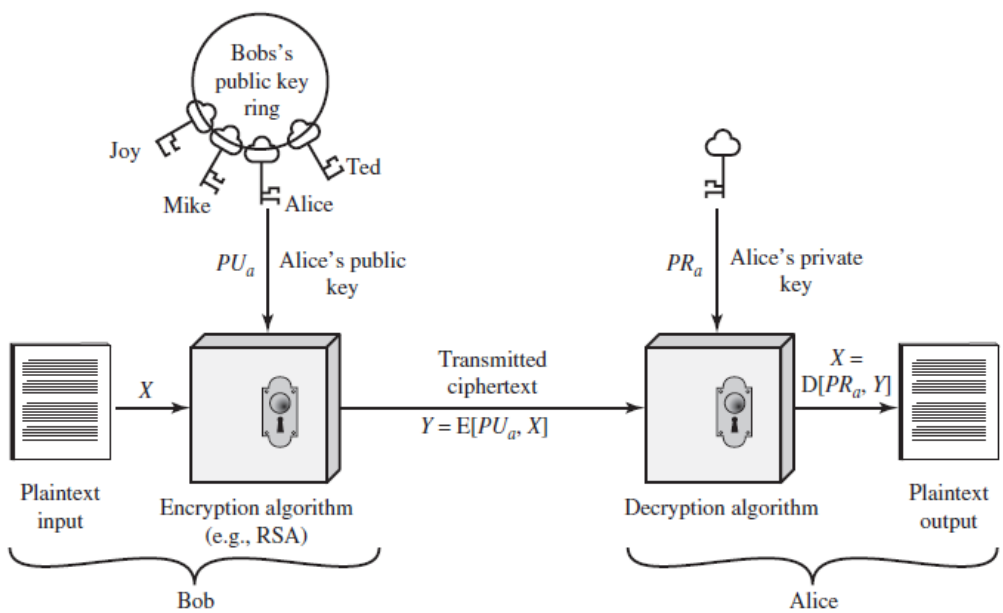
As the names suggest, the public key of the pair is made public for others to use, while the private key is known only to its owner. A general-purpose public-key cryptographic algorithm relies on one key for encryption and a different but related key for decryption.
The essential steps are the following:
1. Each user generates a pair of keys to be used for the encryption and decryption of messages.
2. Each user places one of the two keys in a public register or other accessible file. This is the public key. The companion key is kept private. As Figure 2.6a suggests, each user maintains a collection of public keys obtained from others.
3. If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.
4. When Alice receives the message, she decrypts it using her private key. No other recipient can decrypt the message because only Alice knows Alice's

private key. With this approach, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed. As long as a user protects his or her private key, incoming communication is secure. At any time, a user can change the private key and publish the companion public key to replace the old public key. Figure illustrates another mode of operation of public-key cryptography.

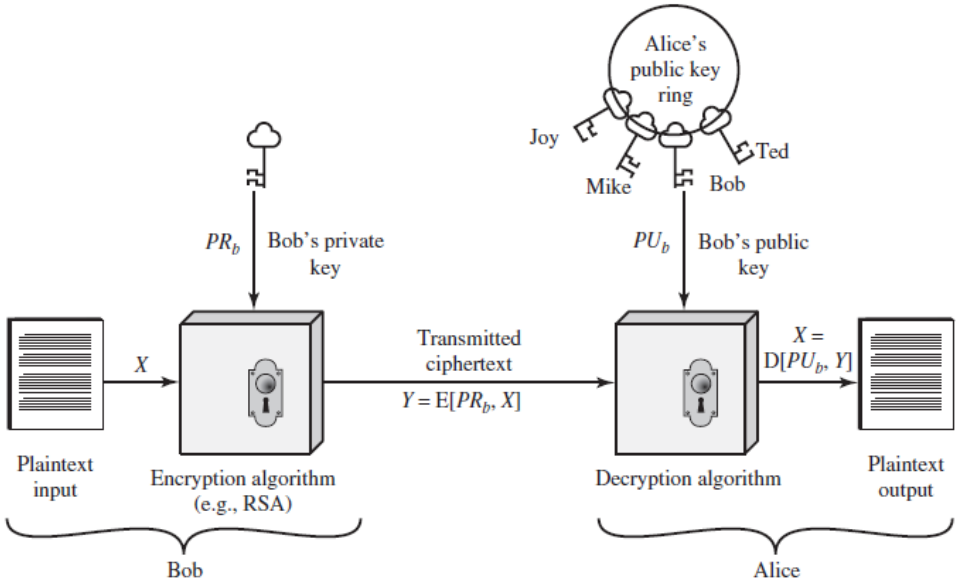
In this scheme, a user encrypts data using his or her own private key. Anyone who knows the corresponding public key will then be able to decrypt the message.

Note that the scheme of Figure is directed toward providing confidentiality:

Only the intended recipient should be able to decrypt the ciphertext because only the intended recipient is in possession of the required private key. Whether in fact confidentiality is provided depends on a number of factors, including the security of the algorithm, whether the private key is kept secure, and the security of any protocol of which the encryption function is a part.



(a) Encryption with public key



(b) Encryption with private key

The scheme of Figure is directed toward providing authentication and/ or data integrity. If a user is able to successfully recover the plaintext from Bob's ciphertext using Bob's public key, this indicates that only Bob could have encrypted the plaintext, thus providing authentication. Further, no one but Bob would be able to modify the plaintext because only Bob could encrypt the plaintext with Bob's private key. Once again, the actual provision of authentication or data integrity depends on a variety of factors.

7.1.5 Applications for Public-Key Cryptosystems

Before proceeding, we need to clarify one aspect of public-key rypto systems that is otherwise likely to lead to confusion. Public-key ystems are characterized by the use of a cryptographic type of algorithm with two keys, one held private and one available publicly. Depending on the application, the sender uses either the sender's private key or the receiver's public key, or both, to perform some type of cryptographic function.

In broad terms, we can classify the use of public-key cryptosystems into three categories: digital signature, symmetric key distribution, and encryption of secret keys.

These applications are discussed in next section. Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications. Table indicates the applications supported by the algorithms discussed in this section.

7.1.6 Requirements for Public-Key Cryptography

The cryptosystem illustrated in the previous Figure depends on a cryptographic algorithm based on two related keys. Diffie and Hellman postulated this system without demonstrating that such algorithms exist. However, they did lay out the conditions that such algorithms must fulfill:

1) It is computationally easy for a party B to generate a pair (public key *PUb*,Private key *PRb*).
2) It is computationally easy for a sender A, knowing the public key and the Message to be encrypted, *M*, to generate the corresponding ciphertext:
   $C = E(PUb, M)$
3) It is computationally easy for the receiver B to decrypt the resulting ciphertext using the private key to recover the original message:
   $M = D(PRb, C) = D[PRb, E(PUb, M)]$
4) It is computationally infeasible for an opponent, knowing the public key, *PUb*, to determine the private key, *PRb*.
5) It is computationally infeasible for an opponent, knowing the public key, *PUb*, and a ciphertext, *C*, to recover the original message, *M*. We can add a sixth requirement that, although useful, is not necessary for all public-key applications:
6) Either of the two related keys can be used for encryption, with the other used for decryption.
   $M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]$

7.1.7 Asymmetric Encryption Algorithms

In this subsection, we briefly mention the most widely used asymmetric encryption algorithms.

RSA One of the first public-key schemes was developed in 1977 by Ron Rivest, AdiShamir, and Len Adleman at MIT and first published in 1978 [RIVE78]. The RSA scheme has since reigned supreme as the most widely accepted and implemented approach to public-key encryption. RSA is a block cipher in which the plaintext and ciphertext are integers between 0 and $n - 1$ for some n.

In 1977, the three inventors of RSA dared *Scientific American* readers to decode a cipher they printed in Martin Gardner's "Mathematical Games" column. They offered a $100 reward for the return of a plaintext sentence, an event they predicted might not occur for some 40 quadrillion years. In April of 1994, a group working over the Internet and using over 1600 computers claimed the prize after only eight months of work.This challenge used a public-key size (length of n) of 129 decimal digits, or around 428 bits.

This result does not invalidate the use of RSA; it simply means that larger key sizes must be used. Currently, a 1024-bit key size (about 300 decimal digits) is considered strong enough for virtually all applications.

7.1.8 Digital Signatures and Key Management

As mentioned in former sections, public-key algorithms are used in a variety of applications. In broad terms, these applications fall into two categories: digital signatures, and various techniques to do with key management and distribution.

With respect to key management and distribution, there are at least three distinct aspects to the use of public-key encryption in this regard:
• The secure distribution of public keys
• The use of public-key encryption to distribute secret keys
• The use of public-key encryption to create temporary keys for message encryption

This section provides a brief overview of digital signatures and the various types of key management and distribution.

Digital Signature

Public-key encryption can be used for authentication, as suggested by Figure. Suppose that Bob wants to send a message to Alice. Although it is not important that the message be kept secret, he wants Alice to be certain that the message is indeed from him. For this purpose, Bob uses a secure hash function, such as SHA-512, to generate a hash value for the message and then encrypts the hash code with his private key, creating a digital signature. Bob sends the message with the signature attached.

When Alice receives the message plus signature, she (1)calculates a hash value for the message; (2) decrypts the signature using Bob's public key; and (3) compares the calculated hash value to the decrypted hash value. If the two hash values match, Alice is assured that the message must have been signed by Bob. No one else has Bob's private key and therefore no one else could have created a ciphertext that could be decrypted with Bob's public key. In addition, it is impossible to alter the message without access to Bob's private key, so the message is authenticated both in terms of source and in terms of data integrity.

It is important to emphasize that the digital signature does not provide confidentiality. That is, the message being sent is safe from alteration but not safe from eavesdropping. This is obvious in the case of a signature based on a portion of the message because the rest of the message is transmitted

in the clear. Even in the case of complete encryption, there is no protection of confidentiality because any observer can decrypt the message by using the sender's public key.

Public-Key Certificates

On the face of it, the point of public-key encryption is that the public key is public. Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large. Although this approach is convenient, it has a major weakness. Anyone can forge such a public announcement. That is, some user could pretend to be Bob and send a public key to another participant or broadcast such a public key. Until such time as Bob discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for Bob and can use the forged keys for authentication.

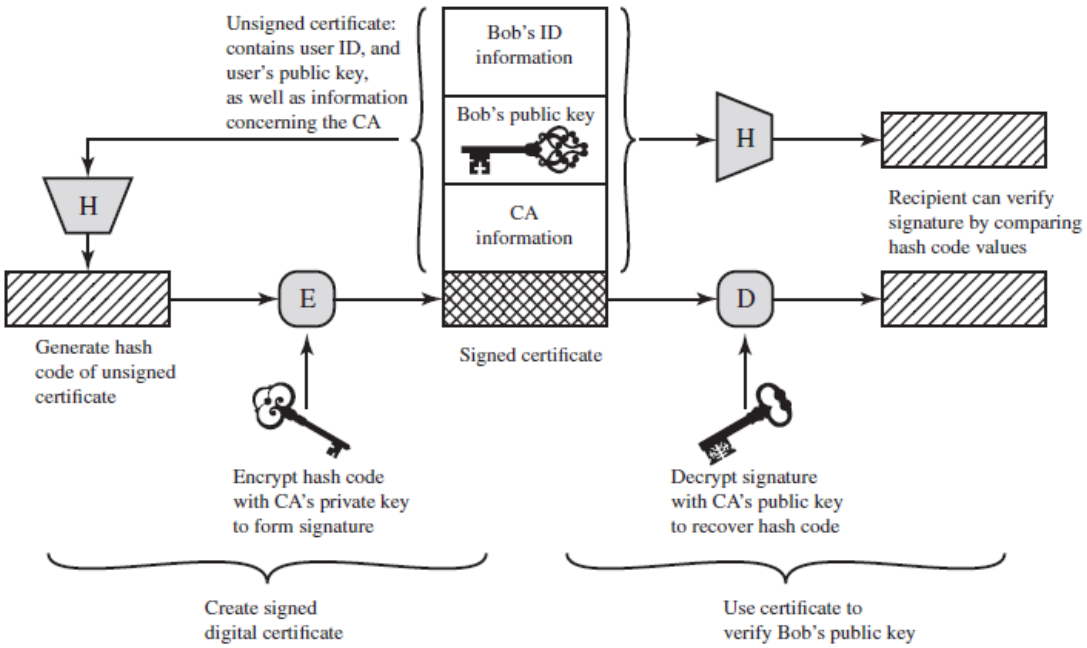
The solution to this problem is the public-key certificate. In essence, a certificate consists of a public key plus a user ID of the key owner, with the whole block signed by a trusted third party. The certificate also includes some information about the third party plus an indication of the period of validity of the certificate. Typically, the third party is a certificate authority (CA) that is trusted by the user community, such as a government agency or a financial institution. A user can present his or her public key to the authority in a secure manner and obtain a signed certificate.

The user can then publish the certificate. Anyone needing this user's public key can obtain the certificate and verify that it is valid by means of the attached trusted signature. Next Figure illustrates the process.

The key steps can be summarized as follows:

1) User software (client) creates a pair of keys: one public and one private.
2) Client prepares an unsigned certificate that includes the user ID and user's public key.
3) User provides the unsigned certificate to a CA in some secure manner. This might require a face-to-face meeting, the use of registered e-mail, or happen via a web form with e-mail verification.
4) CA creates a signature as follows:
    A. CA uses a hash function to calculate the hash code of the unsigned certificate. A hash function is one that maps a variable-length data block or message into a fixed-length value called a hash code, such as SHA family.
    B. CA encrypts the hash code with the CA's private key.
5) CA attaches the signature to the unsigned certificate to create a signed certificate.
6) CA returns the signed certificate to client.

7) Client may provide the signed certificate to any other user.



8) Any user may verify that the certificate is valid as follows:
   a) User calculates the hash code of certificate (not including signature).
   b) User decrypts the signature using CA's known public key.
   c) User compares the results of (a) and (b). If there is a match, the certificate is valid.

One scheme has become universally accepted for formatting public-key certificates: the X.509 standard. X.509 certificates are used in most network security applications, including IP Security (IPsec), Transport Layer Security (TLS), Secure Shell (SSH), and Secure/Multipurpose Internet Mail Extension (S/MIME). We examine most of these applications in Part Five.

7.1.9 Symmetric Key Exchange Using Public-Key Encryption

With symmetric encryption, a fundamental requirement for two parties to communicate securely is that they share a secret key. Suppose Bob wants to create a messaging application that will enable him to exchange e-mail securely with anyone who has access to the Internet or to some other network that the two of them share. Suppose Bob wants to do this using symmetric encryption.

With symmetric encryption, Bob and his correspondent, say, Alice, must come up with a way to share a unique secret key that no one else knows. How are they going to do that? If Alice is in the next room from Bob, Bob could generate a key and write it down on a piece of paper or store it on a disc or thumb drive and hand it to Alice. But if Alice is on the other side of the continent or the world, what can Bob do? He could encrypt this key using symmetric encryption and e-mail it to Alice, but this means that Bob and Alice must share a secret key to encrypt this new secret key.

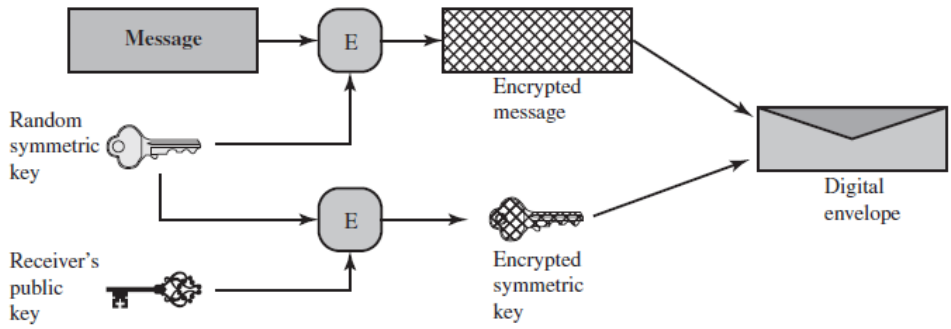
Furthermore, Bob and everyone else who uses this new e-mail package faces the same problem with every potential correspondent: Each pair of correspondents must share a unique secret key. One approach is the use of Diffie-Hellman key exchange. This approach is indeed widely used. However, it suffers the drawback that, in its simplest form, Diffie-Hellman provides no authentication of the two communicating partners. There are variations to Diffie-Hellman that overcome this problem. Also, there are protocols using other public-key algorithms that achieve the same objective.

7.1.10 Digital Envelopes

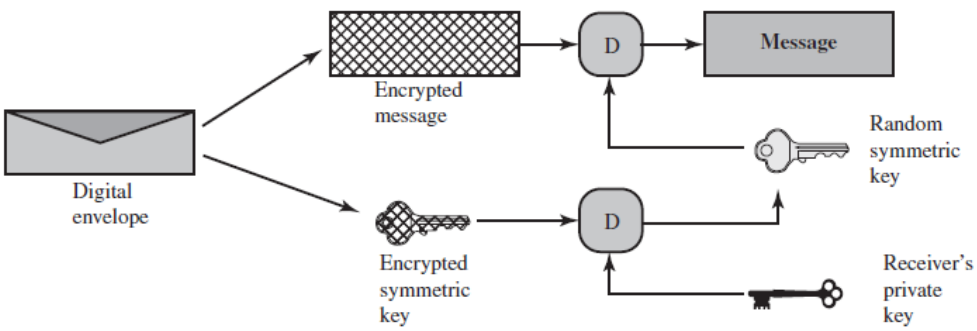
Another application in which public-key encryption is used to protect a symmetric key is the digital envelope, which can be used to protect a message without needing to first arrange for sender and receiver to have the same secret key. The technique is referred to as a digital envelope, which is the equivalent of a sealed envelope containing an unsigned letter. The general approach is shown in Figure. Suppose Bob wishes to send a confidential message to Alice, but they do not share a symmetric secret key. Bob does the following:

1) **Prepare a message.**
2) **Generate a random symmetric key that will be used this one time only.**
3) **Encrypt that message using symmetric encryption the one-time key.**
4) **Encrypt the one-time key using public-key encryption with Alice's public key.**
5) **Attach the encrypted one-time key to the encrypted message and send it to Alice.**

Only Alice is capable of decrypting the one-time key and therefore of recovering the original message. If Bob obtained Alice's public key by means of Alice's public-key certificate, then Bob is assured that it is a valid key.



(a) Creation of a digital envelope



(b) Opening a digital envelope

7.1.11 The Use of Random Numbers

A number of network security algorithms based on cryptography make use of random numbers. For example,

- ✓ **Generation of keys for the RSA public-key encryption algorithm (described in Chapter 21) and other public-key algorithms.**
- ✓ **Generation of a stream key for symmetric stream cipher.**
- ✓ **Generation of a symmetric key for use as a temporary session key or in creating a digital envelope In a number of key distribution scenarios, such as Kerberos (described in**
- ✓ **Chapter 23), random numbers are used for handshaking to prevent replay attacks.**
- ✓ **Session key generation, whether done by a key distribution center or by one of the principals.**

These applications give rise to two distinct and not necessarily compatible Requirements for a sequence of random numbers: randomness and unpredictability.

Randomness **Traditionally, the concern in the generation of a sequence of allegedly random numbers have been that the sequence of numbers be**

random in some well-defined statistical sense. The following criteria are used to validate that a sequence of numbers is random:

- ✓ **Uniform distribution: The distribution of numbers in the sequence should be uniform; that is, the frequency of occurrence of each of the numbers should be approximately the same.**
- ✓ **Independence: No one value in the sequence can be inferred from the others. Although there are well-defined tests for determining that a sequence of numbers matches a particular distribution, such as the uniform distribution, there is no such test to "prove" independence. Rather, several tests can be applied to demonstrate if a sequence does not exhibit independence. The general strategy is to apply a number of such tests until the confidence that independence exists is sufficiently strong.**

In the context of our discussion, the use of a sequence of numbers that appear statistically random often occurs in the design of algorithms related to cryptography. For example, a fundamental requirement of the RSA public-key encryption scheme is the ability to generate prime numbers. In general, it is difficult to determine if a given large number *N* is prime. A brute-force approach would be to divide *N* by every odd integer less than 1N. If *N* is on the order, say, of 10150, a not uncommon occurrence in public-key cryptography, such a brute-force approach, is beyond the reach of human analysts and their computers. However, a number of effective algorithms exist that test the primality of a number by using a sequence of randomly chosen integers as input to relatively simple computations.

If the sequence is sufficiently long (but far, far less than 110150), the primality of a number can be determined with near certainty. This type of approach, known as randomization, crops up frequently in the design of algorithms. In essence, if a problem is too hard or time-consuming to solve exactly, a simpler, shorter approach based on randomization is used to provide an answer with any desired level of confidence.

Random versus Pseudorandom

Cryptographic applications typically make use of algorithmic techniques for random number generation. These algorithms are deterministic and therefore produce sequences of numbers that are not statistically random. However, if the algorithm is good, the resulting sequences will pass many reasonable tests of randomness. Such numbers are referred to as pseudorandom numbers.

You may be somewhat uneasy about the concept of using numbers generated by a deterministic algorithm as if they were random numbers. Despite what might be called philosophical objections to such a practice, it generally works. That is, under most circumstances, pseudorandom

numbers will perform as well as if they were random for a given use. The phrase "as well as" is unfortunately subjective, but the use of pseudorandom numbers is widely accepted. The same principle applies in statistical applications, in which a statistician takes a sample of a population and assumes that the results will be approximately the same as if the whole population were measured.

A true random number generator (TRNG) uses a nondeterministic source to produce randomness. Most operate by measuring unpredictable natural processes, such as pulse detectors of ionizing radiation events, gas discharge tubes, and leaky capacitors.

Intel has developed a commercially available chip that samples thermal noise by amplifying the voltage measured across undriven resistors. LavaRnd is an open-source project for creating truly random numbers using inexpensive cameras, open source code, and inexpensive hardware. The system uses a saturated charge-coupled device (CCD) in a light-tight can as a chaotic source to produce the seed. Software processes the result into truly random numbers in a variety of formats. The first commercially available TRNG that achieves bit production rates comparable with that of PRNGs, is the Intel digital random number generator, offered on new multicore chips since May 2012.

One of the principal security requirements of a computer system is the protection of stored data. Security mechanisms to provide such protection include access control, intrusion detection, and intrusion prevention schemes, all of which are discussed in this book. The book also describes a number of technical means by which these various security mechanisms can be made vulnerable. But beyond technical approaches, these approaches can become vulnerable because of human factors.

We list a few examples here.

• In December of 2004, Bank of America employees backed up and sent to its backup data center tapes containing the names, addresses, bank account numbers, and Social Security numbers of 1.2 million government workers enrolled in a charge-card account. None of the data were encrypted. The tapes never arrived and indeed have never been found. Sadly, this method of backing up and shipping data is all too common. As an another example, in April of 2005, Ameritrade blamed its shipping vendor for losing a backup tape containing unencrypted information on 200,000 clients.

• In April of 2005, San Jose Medical group announced that someone had physically stolen one of its computers and potentially gained access to 185,000 unencrypted patient records.
• There have been countless examples of laptops lost at airports, stolen from a parked car, or taken while the user is away from his or her desk. If the data on the laptop's hard drive are unencrypted, all of the data are

available to the thief.

Although it is now routine for businesses to provide a variety of protections, including encryption, for information that is transmitted across networks, via the Internet, or via wireless devices, once data are stored locally (referred to as *data at rest*), there is often little protection beyond domain authentication and operating system access controls. Data at rest are often routinely backed up to secondary storage such as CD-ROM or tape, archived for indefinite periods. Further, even when data are erased from a hard disk, until the relevant disk sectors are reused, the data are recoverable. Thus it becomes attractive, and indeed should be mandatory, to encrypt data at rest and combine this with an effective encryption key management scheme.

There are a variety of ways to provide encryption services. A simple approach available for use on a laptop is to use a commercially available encryption package such as Pretty Good Privacy (PGP). PGP enables a user to generate a key from a password and then use that key to encrypt selected files on the hard disk. The PGP package does not store the password. To recover a file, the user enters the password, PGP generates the password, and PGP decrypts the file. So long as the user protects his or her password and does not use an easily guessable password, the files are fully protected while at rest. Some more recent approaches are :

- **Back-end appliance: This is a hardware device that sits between servers and storage systems and encrypts all data going from the server to the storage system and decrypts data going in the opposite direction. These devices encrypt data at close to wire speed, with very little latency. In contrast, encryption software on servers and storage systems slows backups. A system manager configures the appliance to accept requests from specified clients, for which unencrypted data are supplied.**

- **Library-based tape encryption: This is provided by means of a co-processor board embedded in the tape drive and tape library hardware. The co-processor encrypts data using a nonreadable key configured into the board. The tapes can then be sent off-site to a facility that has the same tape drive hardware. The key can be exported via secure e-mail or a small flash drive that is transported securely. If the matching tape drive hardware co-processor is not available at the other site, the target facility can use the key in a software decryption package to recover the data.**

- **Background laptop and PC data encryption: A number of vendors offer software products that provide encryption that is transparent to the application and the user. Some products encrypt all or designated files and folders. Other products create a virtual disk, which can be maintained locally on the user's hard drive or maintained on a**

network storage device, with all data on the virtual disk encrypted. Various key management solutions are offered to restrict access to the owner of the data.

8 User authentication

In most computer security contexts, user authentication is the fundamental building block and the primary line of defense. User authentication is the basis for most types of access control and for user accountability. RFC 4949 defines user authentication as follows:

The process of verifying an identity claimed by or for a system entity.
An authentication process consists of two steps:

- Identification step: Presenting an identifier to the security system. (Identifiers should be assigned carefully, because authenticated identities are the basis for other security services, such as access control service.)
- Verification step: Presenting or generating authentication information that corroborates the binding between the entity and the identifier.

For example, user Alice Toklas could have the user identifier ABTOKLAS. This information needs to be stored on any server or computer system that Alice wishes to use and could be known to system administrators and other users. A typical item of authentication information associated with this user ID is a password, which is kept secret (known only to Alice and to the system)

If no one is able to obtain or guess Alice's password, then the combination of Alice's user ID and password enables administrators to set up Alice's access permissions and audit her activity. Because Alice's ID is not secret, system users can send her e-mail, but because her password is secret, no one can pretend to be Alice.

In essence, identification is the means by which a user provides a claimed identity to the system; user authentication is the means of establishing the validity of the claim. Note that user authentication is distinct from message authentication. Message authentication is a procedure that allows communicating parties to verify that the contents of a received message have not been altered and that the source is authentic. This chapter is concerned solely with user authentication.