**UNIT – I**

**Security Attacks on computers and Computer Security**

1. **Introduction**

**The History of Information Security**

* Began immediately after the first mainframes were developed
* Groups developing code-breaking computations during World War II created the first modern computers
* Physical controls to limit access to sensitive military locations to authorized personnel
* Rudimentary in defending against physical theft, espionage, and sabotage

**The 1960s**

* Advanced Research Procurement Agency (ARPA) began to examine feasibility of redundant networked communications
* Larry Roberts developed ARPANET from its inception

**The 1970s and 80s**

* ARPANET grew in popularity as did its potential for misuse
* Fundamental problems with ARPANET security were identified

– No safety procedures for dial-up connections to ARPANET

– Non-existent user identification and authorization to system

* Late 1970s: microprocessor expanded computing capabilities and security threats

**R-609**

* Information security began with Rand Report R-609 (paper that started the study of computer security)
* Scope of computer security grew from physical security to include:

– Safety of data

– Limiting unauthorized access to data

– Involvement of personnel from multiple levels of an organization

**The 1990s**

* Networks of computers became more common; so too did the need to interconnect networks
* Internet became first manifestation of a global network of networks
* In early Internet deployments, security was treated as a low priority

**The Present**

* + The Internet brings millions of computer networks into communication with each other—many of them unsecured
  + Ability to secure a computer’s data influenced by the security of every computer to which it is connected

1. **The need for security** 
   * **“**The quality or state of being secure—to be free from danger”
   * A successful organization should have multiple layers of security in place:

– Physical security

– Personal security

– Operations security

– Communications security

– Network security

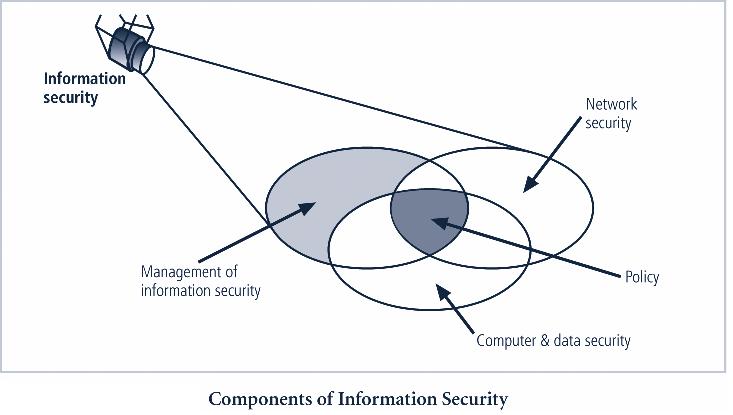
– Information security

1. **Security approaches and Security Principles**

**What is Information Security?**

The protection of information and its critical elements, including systems and hardware that use, store, and transmit that information.

* Necessary tools: policy, awareness, training, education, technology.
* C.I.A. triangle was standard based on confidentiality, integrity, and availability.
* C.I.A. triangle now expanded into list of critical characteristics of information.



**Information Security**: It can be defined as "measures adopted to prevent the unauthorized use, misuse,modification or denial of use of knowledge, facts, data or capabilities". Three aspects of IS are: **Security Attack**: Any action that comprises the security of information

**Security Mechanism**: A mechanism that is designed to detect, prevent, or recover from a security.

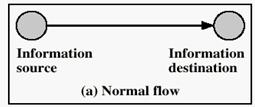
**Security Service**:

It is a processing or communication service that enhances the security of the data processing systems and information transfer. The services are intended to counter security attacks by making use of one or more security mechanisms to provide the service.

1. **Security Attacks:**

We can classify the types of attacks into four types such as

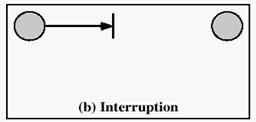
* Interruption
* Interception
* Modification
* Fabrication

The normal flow of information transmission will be like this.

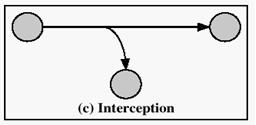
**Interruption**: This is also known as attack on availability. This is an attack where the sender data is somehowinterrupted by the intruder so that it does not reach the destination.

For example, a collection of cooperative intruders would bombard the destination with bulk of messages so that the destination will not get chance to process the sender message.

The following figure illustrate this.

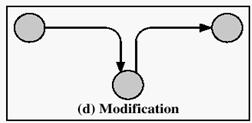


**Interception**: This is also known as attack on confidentiality. It is illustrated below.



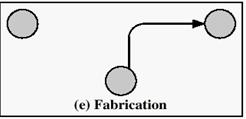
In this kind of attack , intruder would read the message of sender. Here, intruder does not do any harm for the data/message. Detecting this kind of intruders is difficult.

**Modification**: This is also known as an attack on integrity. This is illustrated below.



In this kind of attack , intruder read and modify the sender data/message and then send the modified message to the receiver.

**Fabrication**: This is also known as an attack on authenticity



This is the kind of attack where intruder will send the data to the receiver as if he is original sender. These attacks are further classified into two types such as passive attacks and active attacks.

**a. Passive attacks:**

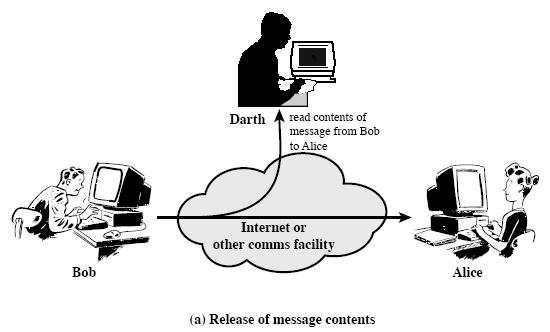
Passive attacks are those wherein the attacker monitor the data transmission. So, the attacker aims to obtain the information that is being transmitted. The term passive indicates that the attacker does not attempt to modify the data.Passive attacks are further classified into two types such as  Release of message contents

* Traffic analysis

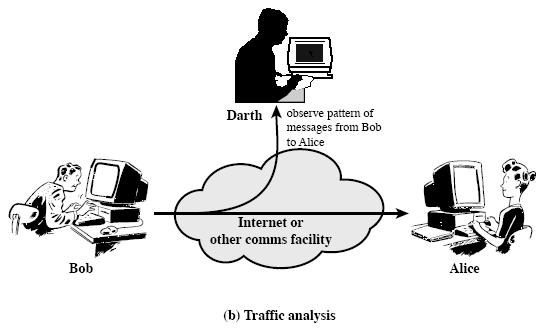
It is illustrated below.



**Release of message contents**: It is quite simple to understand. When we send a confidential email to our friend,we desire that only she be able to access it. Otherwise, the contents of the message are released to someone else. Using certain security mechanism , we can prevent this. This illustrated in the following figure.

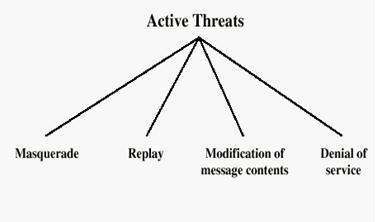


**Traffic analysis**: This is second type of passive attack. If many messages are passing through, a passive attackercould try to figure out the similarities between them to come up with some sort of pattern that provides clues regarding the communication that is taking place.This is illustrated in the following figure.

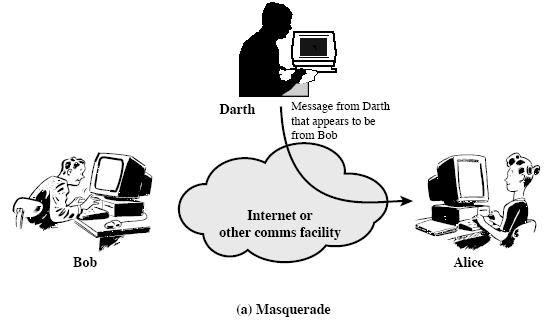


b. **Active attacks:**

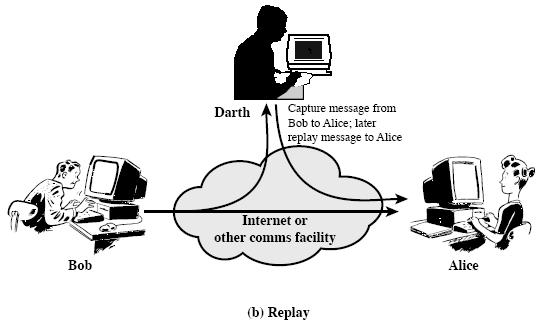
The active attacks involve some modification of the data stream. These attacks can not be prevented easily. These attacks are further classified into four types . It is illustrated in the following figure.



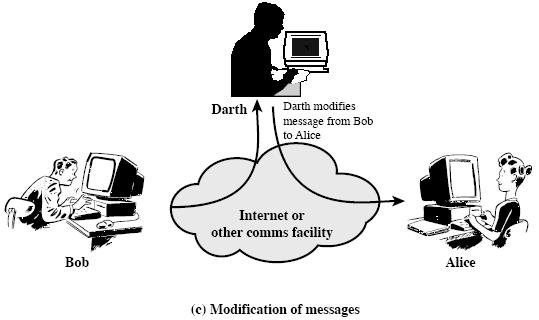
**Masquerade attack**: It takes place when one entity pretend to be a different entity. It is illustrated in thefollowing figure. It resembles attack on authentication.



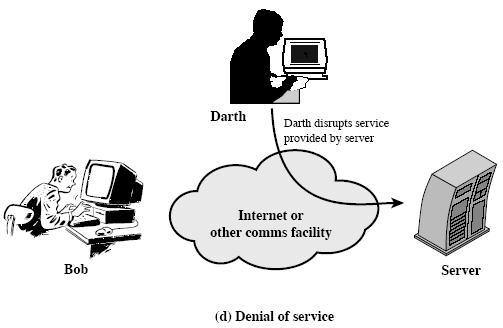
**Replay attack**: It involves the passive capture of a data unit and its subsequent retransmission to produce anunauthorized effect. It is illustrated in the following figure.



**Modification** of message simply means that some portion of the legitimate message is altered, delayed orreordered. It is illustrated below.



**The denial of service** prevents the normal use or management of communication facilities. This attack may havea specific target; for example, an entity may suppress all messages directed to a particular destination..It is illustrated below.



1. **Security Services:**

X.800 standard define the following security services. These services are categorized into five types such as

* 1. Confidentiality (privacy)
  2. Authentication (who created or sent the data)
  3. Integrity (has not been altered)
  4. Non-repudiation (the order is final)
  5. Access control (prevent misuse of resources)
  6. Availability (permanence, non-erasure)

– Denial of Service Attacks

– Virus that deletes files

1. **Confidentiality**: Confidentiality is nothing but the protection of transmitted data from passive attacks.With respect the content several levels of protection can be identified.

For example, when a TCP connection is set up between two systems, this broad protection prevents the release of any user data transmitted over the TCP connection.

The other aspect of confidentiality is the protection of traffic flow from analysis. This requires that the attacker should be able to see the traffic on the specified path.

**2.** **Authentication:** The authentication service is concerned with giving assurance that the communication isauthentic.

Authentication ensures that the receiver is receiving the message/data from intended party only. It also ensures that the message/data is not modified by any intruder.

**3. Access control** :

Access control is the ability to limit and control the access to host systems and applications.

**4. Data Integrity**:

Assurance that data received is as sent by an authorized entity

It allows the recipient of a message to verify it has not been modified in transit.

**5. Non-repudiation**:

Protection against denial by one of the parties in a communication

Makes it difficult for the originator of a message to falsely deny later that they were the party that sent the message.

**6. Availability**

It ensures that a service or information is available to an (authorized) user upon demand and without delay. *Denial of Service (DoS) attacks* seek to interrupt a service or make some information unavailable to legitimateusers.

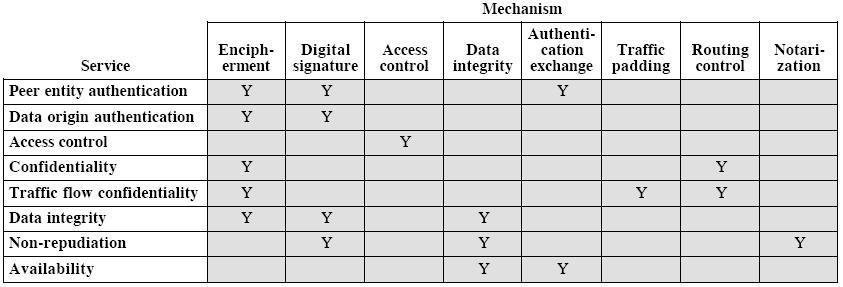
1. **Security mechanisms:**

*No single mechanism can provide all the security services wanted. But encryption or encryption-like information transformation is a key enabling technology.*

**Security Mechanisms**: We have various security mechanisms to achieve information security:

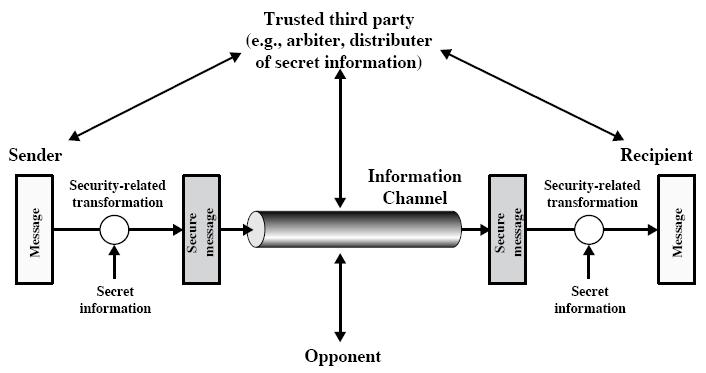
1. **Access Control** : Access control mechanism will control the data access thru access rights. It uses themechanisms such as Discretionary Access control mechanism and Mandatory Access control mechanism.
2. **Encipherment:** It uses mathematical algorithms to transform data into a unreadable format. Thetransformation and subsequent recovery of the data depend on an algorithms and zero or more encryption keys.
3. **Digital signatures:** Digital signature is an electronic signature created thru data to be transferred. Digitalsignature will allow the user to check for the forgery.
4. **Data integrity**: Data integrity ensures that the message is received as it is sent.
5. **Authentication exchange**: A mechanism used to ensure the identity of sender
6. **Traffic padding**: The insertion of bits into gaps in a data steam to frustrate traffic analysis attempts.
7. **Routing control:** Enables selection of safest route.

Relationship between Security Services and Mechanisms



1. **A Model for Network Security**:

A model for network security is shown in the following figure:



A message is to be transferred from one party to another across some sort of network. All the techniques for providing security have the following components:

A security related transformation on the information to be sent Examples include the encryption of the message. Some secret information shared by the two principals and it is hope , unknown to the opponent. For example, an encryption key is used in conjunction with the transformation to encrypt the message before transmission and decrypt the message at the recipient.

A trusted third party may be needed to achieve secure transmission.

2. **Cryptography Concepts and Techniques**

**2.1. Introduction**

**Cryptography**

A cipher is a secret method of writing as by code.

Cryptography is the study of techniques related to aspects of information security. Hence cryptography is concerned with the writing (ciphering or encoding) and deciphering (decoding) of messages in secret code.

**2.2. Plain text and Ciphertext**

**The different terms we find in Cryptography are**

**Plaintext:** This is the original intelligible message or data that is fed into the algorithm as input. **Encryption algorithm**: The encryption algorithm performs various substitutions and transformations on theplaintext.

**Secret key:** The secret key is also input to the encryption algorithm. The key is a value independent of theplaintext and of the algorithm. The algorithm will produce a different output depending on the specific key being used at the time. 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. Fora given message, two different keys will produce two different ciphertexts.

The ciphertext is an apparently random stream of data and, as it stands, is unintelligible.

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

**Cryptographic systems are classified along three independent dimensions:**

1. The type of operations used for performing plaintext to ciphertext

All the encryption algorithms make use of two general principles; substitution and transposition through which plaintext elements are rearranged. Important thing is that no information should be lost.

2. The number of keys used

If single key is used by both sender and receiver, it is called symmetric, single-key, secret-key or conventional encryption. If sender and receiver each use a different key, then it is called asymmetric, two-key or public-key encryption.

3. The way in which plaintext is processed

A block cipher process the input as blocks of elements and generated an output block for each input block. Stream cipher processes the input elements continuously, producing output one element at a time as it goes along.

**Substitution:** Method by which units of plaintext are replaced with ciphertext according to a regular system. **Transposition:** Here, units of plaintext are rearranged in a different and usually quite complex order, but the units

themselves are left unchanged

**Cryptanalysis**

The process of attempting to discover the plaintext or key is known as cryptanalysis. It is very difficult when only the ciphertext is available to the attacker as in some cases even the encryption algorithm is not known. The most common attack under these circumstances is brute-force approach of trying all the possible keys. This attack is made impractical when the key size is considerably large. The table below gives an idea on types of attacks on encrypted messages.

|  |  |  |
| --- | --- | --- |
| **Type of Attack** | **Known to Cryptanalyst** |  |
|  |  |  |
| **Ciphertext only** | Encryption algorithm |  |
|  |  |  |
|  | Ciphertext to be decoded |  |
| **Known plaintext** | Encryption algorithm |  |
|  |  |  |
|  | Ciphertext to be decoded |  |
|  | One or more plaintext-ciphertext pairs formed with the secret |  |
|  | key |  |
| **Chosen plaintext** | Encryption algorithm |  |
|  |  |  |
|  | Ciphertext to be decoded |  |
|  | Plaintext message chosen by cryptanalyst, together with its |  |
|  | ciphertext generated with the secret key |  |
| **Chosen** | Encryption algorithm |  |
| **ciphertext** |  |  |
| Ciphertext to be decoded |  |
|  | Purported (supposed) ciphertext chosen by cryptanalyst, |  |
|  | together with its corresponding decrypted plaintext generated |  |
|  | with the secret key |  |
|  |  |  |
| **Chosen text** | Encryption algorithm |  |
|  |  |  |
|  | Ciphertext to be decoded |  |
|  | Plaintext message chosen by cryptanalyst, together wit h its |  |
|  | ciphertext generated with the secret key |  |
|  |  |  |
|  | Purported (supposed) ciphertext chosen by cryptanalyst, |  |
|  | together with its corresponding decrypted plaintext generated |  |
|  | with the secret key |  |
|  |  |  |

* The most difficult problem is when all that is available is the ***ciphertext*** *only.*
* In some cases, not even the encryption algorithm is known, but in general we can assume that the opponent traces the algorithm used for encryption.
* One possible attack under these circumstances is the brute-force approach of trying all possible keys. If the key space is very large, this becomes impractical.
* Thus, the opponent must rely on an analysis of the ciphertext itself, generally applying various statistical tests to it. To use this approach, the opponent must have some general idea of the type of plaintext that is concealed (masked or hidden), such as English or French text, an EXE file, a

Java source listing, an accounting file, and so on.

The ciphertext only attack is the easiest to defend against because the opponent has the least am ount of information to work with.

In many cases, however, the analyst has more information. The analyst may be able to capture one or more plaintext messages as well as their encryptions. Or the analyst may know that certain plaintext patterns will appear in a message.

For example, a file that is encoded in the Postscript format always begins with the same pattern, or there may be a standardized header or banner to an electronic funds transfer message, and so on.

All these are examples of ***known plaintext****.* With this knowledge, the analyst may be able to deduce

(work out) the key on the basis of the way in which the known plaintext is transformed.

Closely related to the ***known plaintext*** attack is what might be referred to as a probable word attack. If the opponent is working with the encryption of some general prose (text) message, he or she may have little knowledge of what is in the message. However, if the opponent is after some very specific information, then parts of the message may be known.

For example, if an entire accounting file is being transmitted, the opponent may know the placement of certain key words in the header of the file. As another example, the source code for a program developed by a corporation might include a copyright statement in some standardized position.

If the analyst is somehow able to get the source system to insert into the sys tem a message chosen by the analyst, then a chosen plaintext attack is possible.

In general, if the analyst is able to choose the messages to encrypt, th e analyst may deliberately (carefully) pick patterns that can be expected to reveal the structure of the key.

The other two other types of attacks are: ***chosen ciphertext and chosen text***. These are less commonly employed as cryptanalytic techniques but are nevertheless (however) possible avenues (ways) of attack. Only relatively weak algorithms fail to withstand a ciphertext -only attack. Generally, an encryption algorithm is designed to withstand a known-plaintext attack.

An encryption scheme is **computationally secure** if the ciphertext generated by the scheme meets one or both of the following criteria:

* The cost of breaking the cipher exceeds the value of the encrypted information.
* The time required to break the cipher exceeds the useful lifetime of the info rmation. Unfortunately it is very difficult to estimate the amount of effort required to cryptanalyze ciphertext successfully.

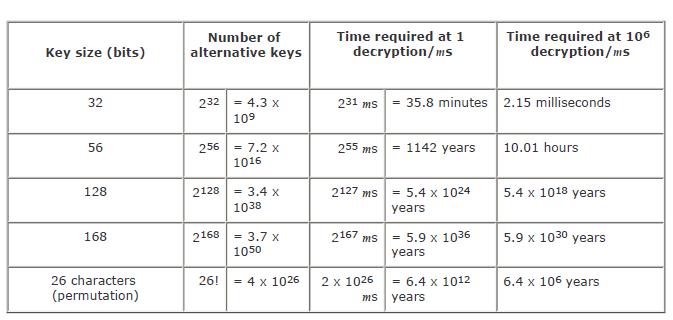
However, assuming there are **no inherent** mathematical weaknesses in the algorithm, then a brute -force approach is indicated, and here we can make some reasonable estimates about costs and time.

A brute-force approach involves trying every possible key until an intelligible (clear) translation of the

ciphertext into plaintext is obtained. On average, half of all pos sible keys must be tried to achieve success.

Table 2.2 shows how much time is involved for various key sizes.

The 56-bit key size is used with the DES



**Table: Average time required for Exhaustive Key Search**

For each key size, the results are shown assuming that it takes 1µs to perform a single decryption, which is a reasonable order of magnitude for today's machines.

The final column of Table 2.2 considers the results for a system that can process 1 million keys per microsecond. As you can see, at this performance level, DES can no longer be considered computationally secure.

**2.3. Substitution:**

Method by which units of plaintext are replaced with ciphertext according to a regular system.

These techniques involve substituting or replacing the contents of the plaintext by other letters, numbers or symbols. Different kinds of ciphers are used in substitution technique.

**Caesar Ciphers:**

It is the oldest of all the substitution ciphers. A Caesar cipher replaces each letter of the plaintext with an alphabet. Two examples can be given: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z Choose k, Shift all letters by k

For example, if k = 5

A becomes F, B becomes G, C becomes H, and so on…

Mathematically give each letter a number, a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25

then have Caesar cipher as:

***c* = E(*p*) = (*p* + *k*) mod (26) *p* = D(c) = (c– *k*) mod (26)**With a Caesar cipher, there are only 26 possible keys, ofwhich only 25 are of any use, since mapping A to A etc doesn't really obscure the message

**Mono alphabetic Ciphers:**

Here, Plaintext characters are substituted by a different alphabet stream of characters shifted to the right or left by n positions. When compared to the Caesar ciphers, these mono alphabetic ciphers are more secure as each letter of the ciphertext can be any permutation of the 26 alphabetic characters leading to 26! or greater than 4 x 1026 possible keys. But it is still vulnerable to cryptanalysis, when a cryptanalyst is aware of the nature of the plaintext, he can find the regularities of the language. To overcome these attacks, multiple substitutions for a single letter are used. For example, a letter can be substituted by different numerical cipher symbols such as 17, 54, 69….. etc. Even this method is not completely secure as each letter in the plain text affects on letter in the ciphertext. Or, using a common key which substitutes every letter of the plain text. The key *ABCDEFGHIIJ* *KLMNOPQRSTUVWXYZ QWERTYUIIOPAS DFGHJ KLZXCV BNM* Would encrypt the message *II think therefore II am* into***OZIIOFAZIITKTYGKTOQD***

But any attacker would simply break the cipher by using frequency analysis by observing the number of times each letter occurs in the cipher text and then looking upon the English letter frequency table. So, substitution cipher is completely ruined by these attacks. Monoalphabetic ciphers are easy to break as they reflect the frequency of the original alphabet. A countermeasure is to provide substitutes, known as homophones for a single letter.

**Playfair Ciphers:**

It is the best known multiple –letter encryption cipher which treats digrams in the plaintext as single units and translates these units into ciphertext digrams. The Playfair Cipher is a digram substitution cipher offering a relatively weak method of encryption. It was used for tactical purposes by British forces in the Second Boer War

and in World War I and for the same purpose by the Australians and Germans during World War II. This was because Playfair is reasonably fast to use and requires no special equipment. A typical scenario for Playfair use would be to protect important but non-critical secrets during actual combat. By the time the enemy cryptanalysts could break the message, the information was useless to them. It is based around a 5x5 matrix, a copy of which is held by both communicating parties, into which 25 of the 26 letters of the alphabet (normally either j and i are represented by the same letter or x is ignored) are placed in a random fashion. For example, the plain text is *Shi* *Sherry loves Heath Ledger* and the agreed key is *sherry*. The matrix will be built according to the following rules.in pairs, without punctuation,

All Js are replaced with Is.

*SH IS HE RR YL OV ES HE AT HL ED GE R*

Double letters which occur in a pair must be divided by an X or a Z.

*SH IS HE RX RY LO VE SH EA TH LE DG ER* The alphabet square is prepared using, a 5\*5 matrix, no repetitionletters, no Js and key is written first followed by the remaining alphabets with no i and j.

S H E R Y

A B C D F

G I K L M

N O P Q T

U V W X Z

For the generation of cipher text, there are three rules to be followed by each pair of letters.

Letters appear on the same row： replace them with the letters to their immediate right respectively

Letters appear on the same column： replace them with the letters immediately below respectively

Not on the same row or column： replace them with the letters on the same row respectively but at the other pair

of corners of the rectangle defined by the original pair.

Based on ***HE GH ER DR YS IQ WH***

***HE SC OY KR AL RY***

Another example which is simpler than the above one can be given as:

Here, key word is *playfair*. Plaintext is *Hellothere hellothere* becomes-----*he lx lo th er ex* .

Applying the rules again, for each pair, If they are in the same row, replace each with the letter to its right (mod 5) *he KG* If they are in the same column, replace each with the letter below it (mod 5) *lo RV* Otherwise, replace eachwith letter we’d get if we swapped their column indices *lx YV*

So the cipher text for the given plain text is **KG YV RV QM GI KU** To decrypt the message, just reverse the process. Shift up and left instead of down and right. Drop extra x’s and locate any missing I’s that should be j’s.

The message will be back into the original readable form. no longer used by military forces because of the advent of digital encryption devices.

Playfair is now regarded as insecure for any purpose because modern hand-held computers could easily break the cipher within seconds.

**2.4. Transposition:**

Here, units of plaintext are rearranged in a different and usually quite complex order, but the units themselves are left unchanged.

A **transposition cipher** is a method of encryption by which the positions held by units of plaintext (which are commonly characters or groups of characters) are shifted according to a regular system, so that the ciphertext constitutes a permutation of the plaintext. That is, the order of the units is changed. Transposition ciphers encrypt plaintext by moving small pieces of the message around. Anagrams are a primitive transposition cipher. This table shows "VOYAGER" being encrypted with a primitive transposition cipher where every two letters are switched with each other:

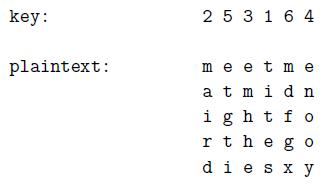
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| V | O | Y | A | G | E | R |
|  |  |  |  |  |  |  |
| O | V | A | Y | E | G | R |
|  |  |  |  |  |  |  |

Another simple example for transposition cipher is the rail fence technique, in which the plaintext is written down as a sequence of diagonals and then read off as a sequence of rows. For example, write the message “meet me after the toga party” out as:

**m e m a t r h t g p r y e t e f e t e o a a t MEMATRHTGPRYETEFETEOAAT**

The following example shows how a pure permutation cipher could work: You write your plaintext message along the rows of a matrix of some size. You generate ciphertext by reading along the columns. The order in which you read the columns is determined by the encryption key:

Ciphertext: ***TITESMAIRDEMHHEENOOYETGTI*** The cipher can be made more secure by performing multiple rounds of such permutations.



**2.5. Symmetric Key Cryptography**

Symmetric encryption, also referred to as conventional encryption, secret -key, or single-key encryption, was the only type of encryption in use prior to the development of public-key encryption in the late 1970s. It remains by far the most widely used of the two types of encryption.

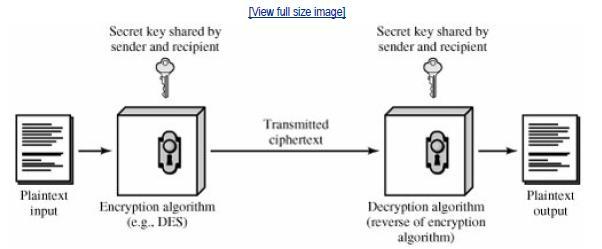
In this unit we begin with the general model for the symmetric encryp tion process: this enables us to understand the context within which the algorithms are used.

Then we look at three important encryption algorithms: DES, triple DES, and AES. We then examine the application of these algorithms to achieve confidentiality.

**CONVENTIONAL ENCRYPTION PRINCIPLES**

A symmetric encryption scheme has *five* ingredients (components)

* **Plain text:** This is the original message or data that is fed into the algorithm as input.
* **Encryption algorithm:** The encryption algorithm performs various substitutions andtransformations on the plaintext.
* **Secret key:** The secret key is also input to the 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 the encryption algorithm run in reverse.
  + It takes the ciphertext and the same secret key and produces the original plaintext.



**Fig 2.1: Simplified Model of Conventional Encryption**

There are two requirements for secure use of symmetric encryption;

1. **W e need a strong encryption algorithm**. The algorithm should be in such a way that anopponent who knows the algorithm and has access to one or more ciphertexts will not be able 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 (control) 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 usi ng this key is readable.

The security of symmetric encryption depends on the secrecy of the key, not the secrecy of the algorithm.  That is, it is not possible to decrypt a message on the basis of the ciphertext plus knowledge of the encryption/decryption algorithm. In other words, we do not need to keep the algorithm secret; we

need to keep only the key secret.

This feature of symmetric encryption is makes it feasible (possible) for widespread use.

But with the use of symmetric encryption, t h e pr i n ci p al security problem is maintaining the secrecy of the key.

**2.6. Asymmetric cryptography**

The development of public-key cryptography is the greatest and perhaps the only true revolution in the entire history of cryptography. It is *asymmetric*, involving the use of two separate keys, in contrast to symmetric encryption, which uses only one key. Public key schemes are neither more nor less secure than private key (security depends on the key size for both). Public-key cryptography *complements rather than replaces* symmetric cryptography. Both also have issues with key distribution, requiring the use of some suitable protocol. The concept of public-key cryptography evolved from an attempt to attack two of the most difficult problems associated with symmetric encryption:

1.) ***key distribution*** – how to have secure communications in general without having to trust a KDC with your key 2.) ***digital signatures*** – how to verify a message comes intact from the claimed sender

**Public-key/two-key/asymmetric** cryptography involves the use of **two** keys:

**Public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures Private-key,** known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**.

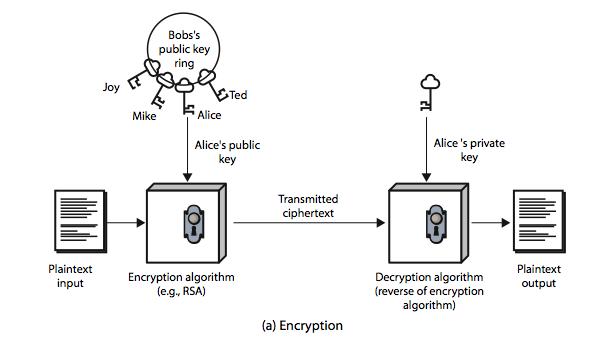
It is **asymmetric** because those who encrypt messages or verify signatures cannot decrypt messages or create signatures

Public-Key algorithms rely on one key for encryption and a different but related key for decryption. These algorithms have the following important characteristics:

It is computationally infeasible to find decryption key knowing only algorithm & encryption key It is computationally easy to en/decrypt messages when the relevant (en/decrypt) key is known

Either of the two related keys can be used for encryption, with the other used for decryption (for some algorithms like RSA)

The following figure illustrates public-key encryption process and shows that a public-key encryption scheme has six ingredients: plaintext, encryption algorithm, public & private keys, ciphertext & decryption algorithm.



The essential steps involved in a public-key encryption scheme are given below: 1.) Each user generates a pair of keys to be used for encryption and decryption.

2.) Each user places one of the two keys in a public register and the other key is kept private.

3.) If B wants to send a confidential message to A, B encrypts the message using A’s public key.

4.) When A receives the message, she decrypts it using her private key. Nobody else can decrypt the message because that can only be done using A’s private key (Deducing a private key should be infeasible).

5.) If a user wishes to change his keys –generate another pair of keys and publish the public one: no interaction with other users is needed.

Notations used in Public-key cryptography: The public key of user A will be denoted **KUA**. The private key of user A will be denoted **KRA**. Encryption method will be a function E. Decryption method will be a function D.

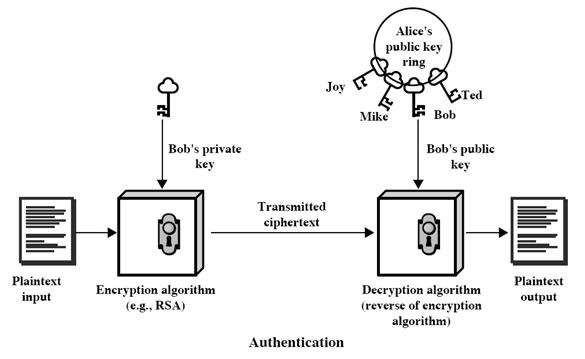
If B wishes to send a plain message X to A, then he sends the ciphertext Y=E(KUA,X) The intended receiver A will decrypt the message: D(KRA,Y)=X

The first attack on Public-key Cryptography is the attack on Authenticity.

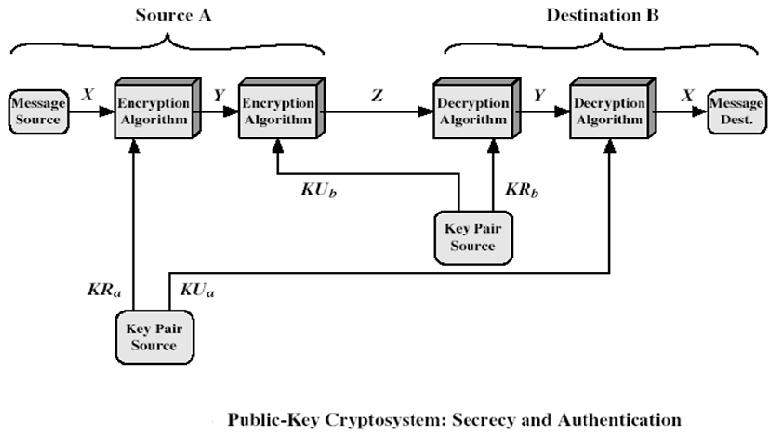
**An attacker may impersonate user B**: he sends a message E(KUA,X) and claims in the message to be B–A hasno guarantee this is so. To overcome this, B will encrypt the message using his private key: Y=E(KRB,X).

Receiver decrypts using B’s public key KRB. This shows the authenticity of the sender because (supposedly) he is the only one who knows the private key. The entire encrypted message serves as a digital signature. This scheme is depicted in the following figure:

But, a drawback still exists. Anybody can decrypt the message using B’s public key. So, secrecy or confidentiality is being compromised. One can provide both *authentication and confidentiality* using the public-key scheme twice:



But, a drawback still exists. Anybody can decrypt the message using B’s public key. So, secrecy or confidentiality is being compromised. One can provide both *authentication and confidentiality* using the public-key scheme twice:



B encrypts X with his private key: Y=E(KRB,X)

B encrypts Y with A’s public key: Z=E(KUA,Y)

A will decrypt Z (and she is the only one capable of doing it): Y=D(KRA,Z)

A can now get the plaintext and ensure that it comes from B (he is the only one who knows his private key): decrypt Y using B’s public key: X=E(KUB,Y).

*Applications for public-key cryptosystems:*

1.) **Encryption/decryption**: sender encrypts the message with the receiver’s public key.

2.) **Digital signature**: sender “signs” the message (or a representative part of the message) using his private key

3.) **Key exchange**: two sides cooperate to exchange a secret key for later use in a secret-key cryptosystem.

*The main requirements of Public-key cryptography are:*

1. Computationally easy for a party B to generate a pair (public key KUb, private key KRb).
2. Easy for sender A to generate ciphertext:
3. Easy for the receiver B to decrypt ciphertect using private key:
4. Computationally infeasible to determine private key (KRb) knowing public key (KUb)
5. Computationally infeasible to recover message M, knowing KUb and ciphertext C
6. Either of the two keys can be used for encryption, with the other used for decryption:

Easy is defined to mean a problem that can be solved in polynomial time as a function of input length. A problem is infeasible if the effort to solve it grows faster than polynomial time as a function of input size. Public-key cryptosystems usually rely on difficult math functions rather than S-P networks as classical cryptosystems. **One-way function** is one, easy to calculate in one direction, infeasible to calculate in the other direction (i.e., theinverse is infeasible to compute). **Trap-door function** is a difficult function that becomes easy if some extra information is known. Our aim to find a ***trap-door one-way function*** is easy to calculate in one direction and infeasible to calculate in the other direction unless certain additional information is known. *Security of Public-key* *schemes:*

Like private key schemes brute force **exhaustive search** attack is always theoretically possible. But keys used are too large (>512bits).

Security relies on a **large enough** difference in difficulty between **easy** (en/decrypt) and **hard** (cryptanalyse) problems. More generally the **hard** problem is known, its just made too hard to do in practise.

Requires the use of **very large numbers,** hence is **slow** compared to private key schemes

**2.7. Steganography**

Steganography is the art and science of hiding communication; a steganographic system thus embeds hidden content in unremarkable cover media so as not to arouse an eavesdropper’s suspicion. In the past, people used hidden tattoos or invisible ink to convey steganographic content. Today, computer and network technologies provide easy-to-use communication channels for steganography. Essentially, the information-hiding process in a steganographic system starts by identifying a cover medium’s redundant bits (those that can be modified without destroying that medium’s integrity). The embedding process creates a stego medium by replacing these redundant bits with data from the hidden message. Modern steganography’s goal is to keep its mere presence undetectable, but steganographic systems— because of their invasive nature—leave behind detectable traces in the cover medium. Even if secret content is not revealed, the existence of it is: modifying the cover medium changes its statistical properties, so eavesdroppers can detect the distortions in the resulting stego medium’s statistical properties. The process of finding these distortions is called statistical steganalysis.

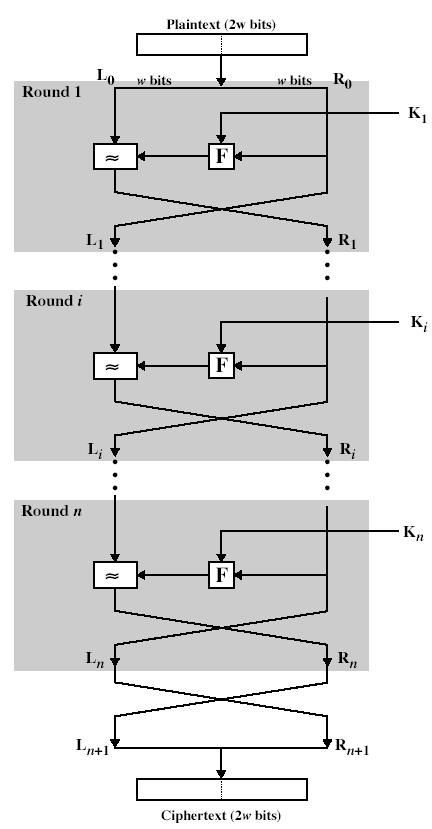
**\* \* \* \* \* \* \* \*END OF UNIT I\* \* \* \* \* \* \* \***

**3. Symmetric key ciphers:**

**3.1. Block cipher principles**

**Feistel Cipher Structure**

Most symmetric block ciphers are based on a *Feistel Cipher Structure.* It was first described by Horst Feistel of IBM in 1973 and is still forms the basis for almost all conventional encryption schemes. It makes use of two properties namely *diffusion* and *confusion*; identified by Claude Shannon for frustrating statistical cryptanalysis. Confusion is basically defined as the concealment of the relation between the secret key and the cipher text. On the other hand, diffusion is regarded as the complexity of the relationship between the plain text and the cipher text.



The function of Feistel Cipher is shown in the above figure and can be explained by following steps: The input to the encryption algorithm is a plaintext block of length 2w bits and a key K.

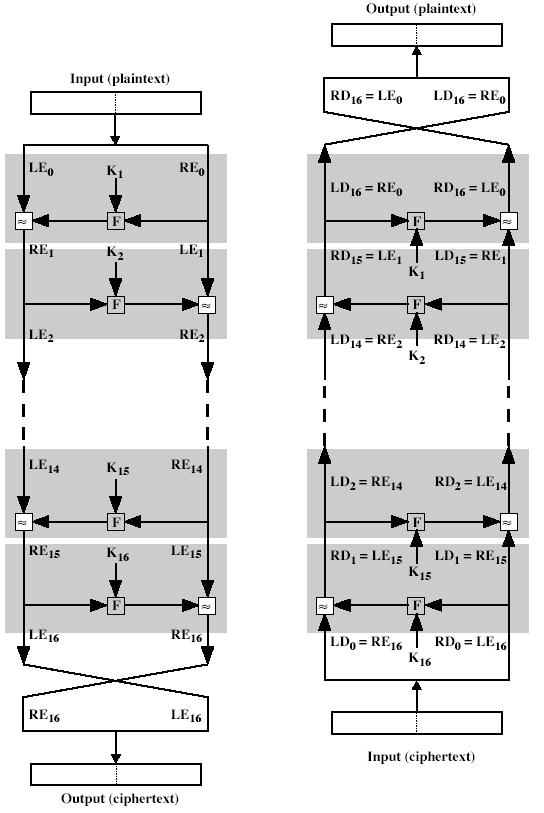
The two halves pass through n rounds of processing and then combine to produce the cipher text block

Each Round i has inputs Li-1 and Ri-1, derived from the previous round, as well as a unique subkey Ki generated by a sub-key generation algorithm.

applying a round function F to right half of data and then taking XOR of the output of that function and left half of data. The round function F is common to every round but parameterized by round subkey Ki.

The structure is a particular form of substitution-permutation network (SPN) proposed by Shannon. The realization or development of a Feistel encryption scheme depends on the choice of the following parameters and design features:

* **Block size:** larger block sizes mean greater security but slower processing. Block size of 64 bits has been nearlyuniversal in block cipher design.
* **Key Size:** larger key size means greater security but slower processing. Most common key length in modernalgorithms is 128 bits.
* **Number of rounds:** multiple rounds offer increasing security but slows cipher. Typical size is 16 rounds.
* **Subkey generation algorithm:** greater complexity will lead to greater difficulty of cryptanalysis.
* **Round Function:** greater complexity will make cryptanalysis harder.
* **Fast software en/decryption & ease of analysis:** are more recent concerns for practical use and testing.

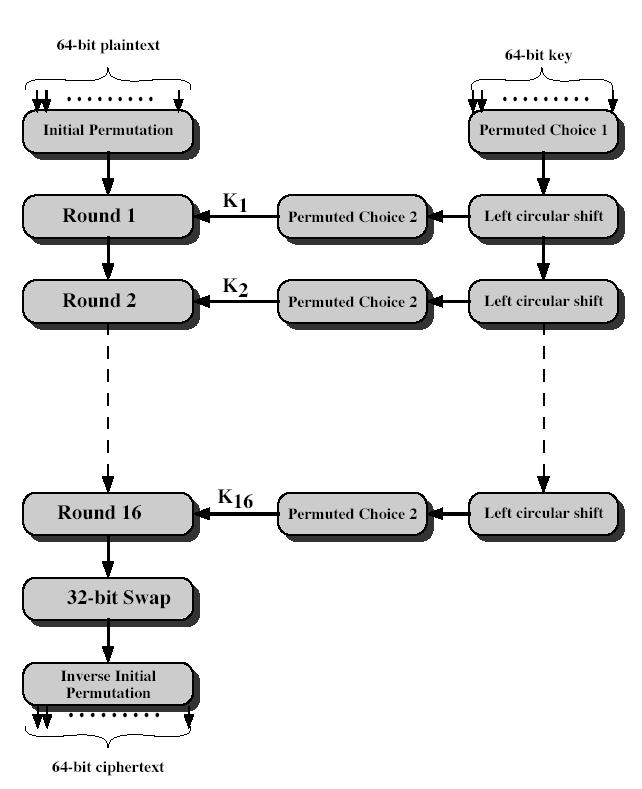


**Feistel Cipher Decryption**

The process of decryption with a Fiestel cipher is same as the encyption process. Use the ciphertext as input to the algorithm, but use the subkeys Ki in the reverse order. Use Kn in the first round and Kn-1 in the second round and so on until k1 is used in the last round. Main advantage is we need not implement two different algorithms for encryption and decryption. The Fiestel cipher has the advantage that encryption and decryption operations are very similar, even identical in some cases requiring only a reversal in the key schedule. Therefore, the size of the code or circuitry required to implement such a cipher is nearly halved.

**3.2. Data Encryption Standard**

In 1974, IBM proposed "Lucifer", an encryption algorithm using 64-bit keys. Two years later (1977), NBS (now NIST) in consultation with NSA made a modified version of that algorithm into a standard. DES uses the two basic techniques of cryptography - confusion and diffusion. At the simplest level, diffusion is achieved through numerous permutations and confusion is achieved through the XOR operation and the S-Boxes. This is also called an S-P network The DES encryption scheme can be explained by the following figure



The plain text is 64 bits in length and the key in 56 bits in length. Longer plain text amounts are processed in 64-bit blocks. The main phases in the left hand side of the above figure i.e. processing of the plain text are,

**Initial Permutation (IP):** The plaintext block undergoes an initial permutation. 64 bits of the block arepermuted.

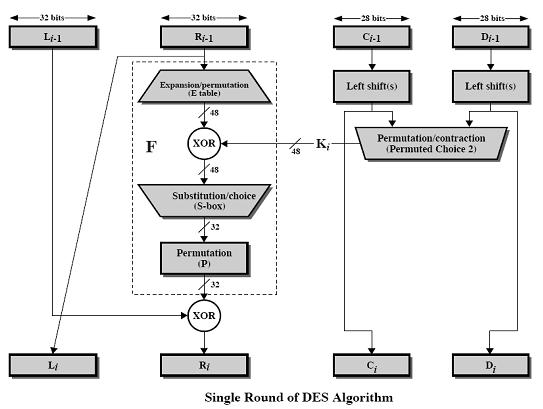
**A Complex Transformation:** 64 bit permuted block undergoes 16 rounds of complex transformation. Subkeysare used in each of the 16 iterations.

**32-bit swap:** The output of 16th round consists of 64bits that are a function of input plain text and key.32 bit leftand right halves of this output is swapped.

**Inverse Initial Permutation (IP-1):** The 64 bit output undergoes a permutation that is inverse of the initialpermutation.

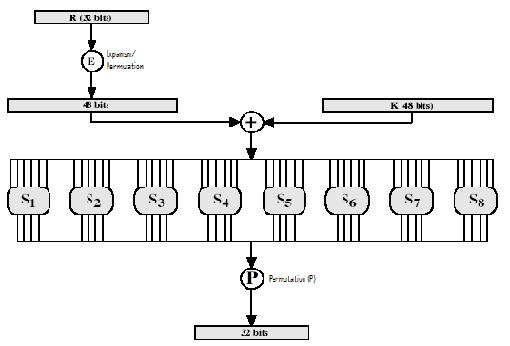
On the right hand side part of the figure, the usage of the 56 bit key is shown. Initially the key is passed through a permutation function. Now for each of the 16 iterations, a new subkey (Ki) is produced by combination of a left circular shift and a permutation function which is same for each iteration. A different subkey is produced because of repeated shifting of the key bits.

The following figure shows a closer view of algorithms for a single iteration. The 64bit permuted input passes through 16 iterations, producing an intermediate 64-bit value at the conclusion of each iteration.



The left and right halves of each 64 bit intermediate value are treated as separated 32-bit quantities labeled L (left) and R (Right). The overall processing at each iteration is given by following steps, which form one round in an S-P network.

The left hand output of an iteration (Li) is equal to the right hand input to that iteration Ri-1. The right hand output Ri is exclusive OR of Li-1 and a complex function F of Ri-1 and Ki. The function F can be depicted by the following figure. S1, S2-----S8 represent the ”S-boxes” , which maps each combination of 48 input bits into a particular 32 bit pattern. For the generation of sub key of length 48 bits, a 56bit key is used which is first passed through a permutation function and then halved to get two 28 bit quantities labeled C0 and D0. At each iteration, these two C and D are subjected to a circular left shift or rotation of 1 or 2 bits. These shifted values serve as input to the next iteration and also to another permutation function which produces a 48-bit output. This output is fed as input to function **F(R i-1, Ki).**



The first and last bits of the input to the box Si form a 2-bit binary number to select one of four substitutions defined by the four rows in the table for Si. The middle 4-bits select a particular column. The decimal value in the cell selected by the row and column is converted to its 4-bit representation to produce the output.

The substitution consists of a set of eight S-boxes, each of which accepts 6 bits as input and produces 4 bits as output. The process of decryption with DES is essentially the same as the encryption process: no different algorithm is used. The ciphertext is used as input to the DES algorithm and the keys are used in the reverse order i.e. K16 in the first iteration, K15 on the second iteration and so on until k1 is used on the sixteenth and last iteration.

**Strength of DES:** *Avalanche Effect*: An effect in DES and other secret key ciphers where each small change inplaintext implies that somewhere around half the ciphertext changes. The avalanche effect makes it harder to successfully cryptanalyze the ciphertext. DES exhibits a strong Avalanche effect. Concern about the strength of DES falls into two categories i.e. strength of algorithm itself and use of 56- bit key. Though many attempts were made over the years to find and exploit weaknesses in the algorithm, none of them were successful in discovering any fatal weakness in DES. A serious concern is with the key size as the time passed the security in DES became getting compromised by the advent of supercomputers which succeeded in breaking the DES quickly using a brute-force attack. If the only form of attack that could be made on an encryption algorithm is brute force, the way of countering it is obviously using long keys. If a key of size 128 bits is used, it takes approximately 1018 years to break the code making the algorithm unbreakable by brute-force approach. The two analytical attacks on DES are Differential cryptanalysis and Linear cryptanalysis. Both make use of Known plaintext-ciphertext pairs and try to attack the round structure and the S-Boxes. Recent advancements showed that using Differential cryptanalysis, DES can be broken using 247 plaintext-ciphertext pairs and for linear cryptanalysis, the number is even reduced to 241.

**Triple DES**

The first answer to problems of DES is an algorithm called Double DES which includes double encryption with two keys. It increases the key size to 112 bits, which seems to be secure. But, there are some problems associated with this approach. issue of reduction to single stage: In other words, could there be a key K3 such that EK2

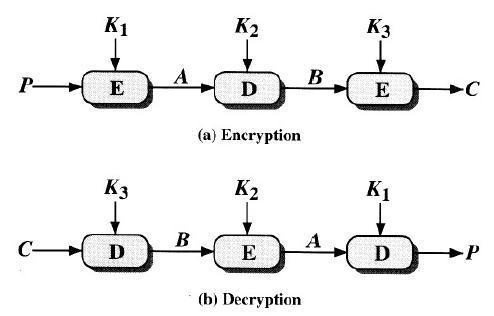
(EK21(P))= EK3(P)? “meet-in-the-middle” attack:

**X = EK1(P) = DK2(C)**

Test the two keys for the second pair of plaintext-ciphertext and if they match, correct keys are found

Triple DES was the answer to many of the shortcomings of DES. Since it is based on the DES algorithm, it is very easy to modify existing software to use Triple DES. 3DES was developed in 1999 by IBM – by a team led by Walter Tuchman. 3DES prevents a meet-in-the-middle attack. 3DES has a 168-bit key and enciphers blocks of

64 bits. It also has the advantage of proven reliability and a longer key length that eliminates many of the shortcut attacks that can be used to reduce the amount of time it takes to break DES. 3DES uses three keys and three executions of the DES algorithm. The function follows an encrypt-decrypt-encrypt (EDE) sequence.



plaintext and EK[X] = encryption of X using key K DK[Y] = decryption of Y using key K Decryption is simply the same operation with the keys reversed

Triple DES runs three times slower than standard DES, but is much more secure if used properly. With three distinct keys, TDEA has an effective key length of 168 bits making it a formidable algorithm. As the underlying algorithm is DEA, it offers the same resistance to cryptanalysis as is DEA. Triple DES can be done using 2 keys or 3 keys.

**3.3. AES**

The AES Cipher The Rijndael proposal for AES defined a cipher in which the block length and the key length can be independently specified to be 128, 192, or 256 bits. The AES specification uses the same three key size alternatives but limits the block length to 128 bits. The number of rounds is dependent on the key size i.e. for key sizes of **128/192/256 bits,** the number of rounds are **10/12/14**. AES is an iterated cipher (rather than Feistel cipher) as it processes data as block of 4 columns of 4 bytes and operates on entire data block in every round. Rijndael was designed to have the following characteristics:

The input to the encryption and decryption algorithms is a single 128-bit block. In FIPS PUB 197, this block is depicted as a square matrix of bytes. This block is copied into the State array, which is modified at each stage of encryption or decryption. After the final stage, State is copied to an output matrix. In the same way, the 128-bit key is depicted as a square matrix of bytes. This key is then expanded into an array of key schedule words; each word is four bytes and the total key schedule is 44 words for the 128-bit key.

1. The key that is provided as input is expanded into an array of forty-four 32-bit words, w[i]. Four distinct words (128 bits) serve as a round key for each round; these are indicated in above figure.
2. Four different stages are used, one of permutation and three of substitution:

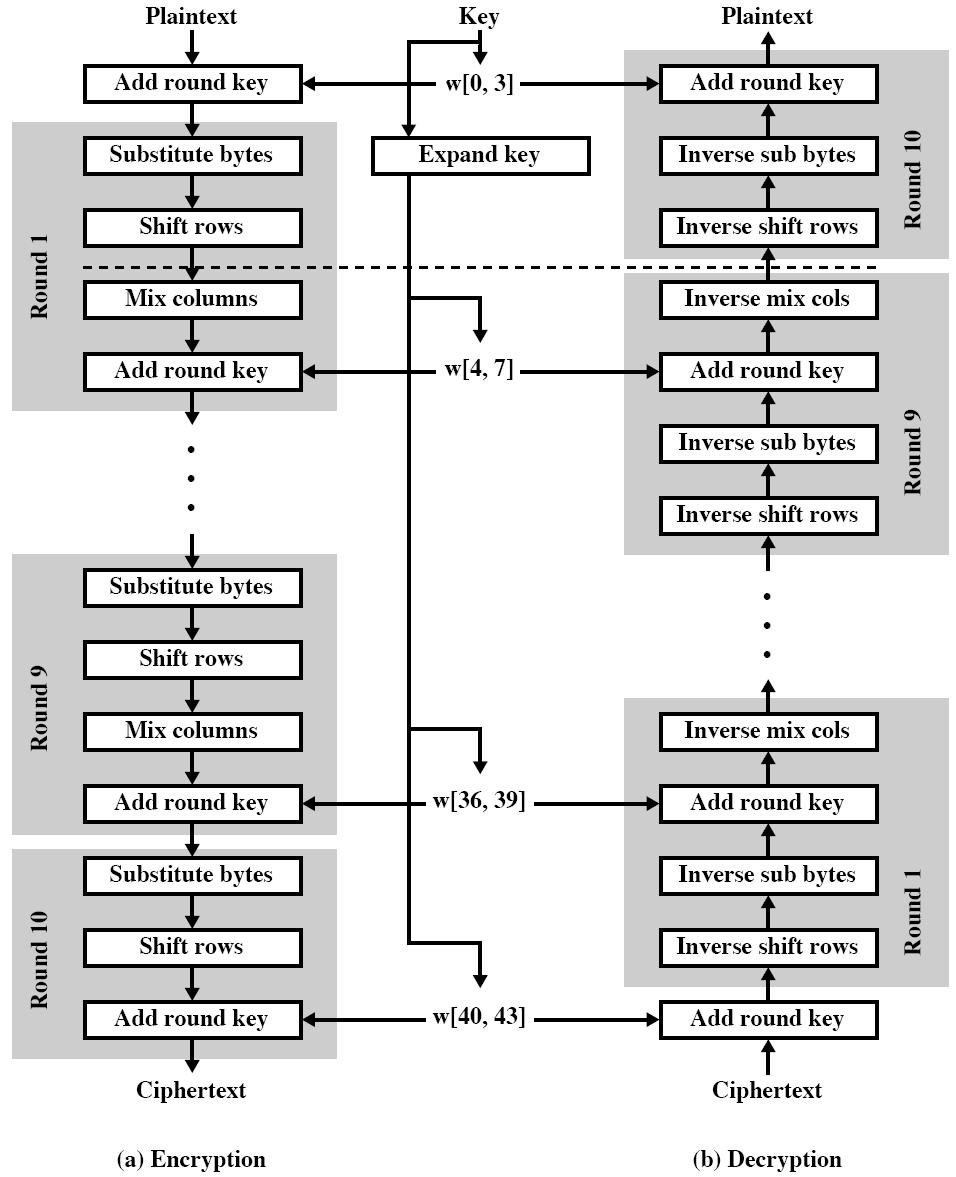
I. *Substitute bytes:* Uses an S-box to perform a byte-by-byte substitution of the block II. *ShiftRows:* A simple permutation

III. *MixColumns*: A substitution that makes use of arithmetic over GF(28)

IV. *AddRoundKey:* A simple bitwise XOR of the current block with a portion of the expanded key

3. The structure is quite simple. For both encryption and decryption, the cipher begins with an AddRoundKey stage, followed by nine rounds that each includes all four stages, followed by a tenth round of three stages. The following figure depicts the structure of a full encryption round.

1. Only the AddRoundKey stage makes use of the key. For this reason, the cipher begins and ends with an AddRoundKey stage. Any other stage, applied at the beginning or end, is reversible without knowledge of the key and so would add no security.
2. The AddRoundKey stage is, in effect, a form of Vernam cipher and by itself would not be formidable. The other three stages together provide confusion, diffusion, and nonlinearity, but by themselves would provide no security because they do not use the key. We can view the cipher as alternating operations of XOR encryption (AddRoundKey) of a block, followed by scrambling of the block (the other three stages), followed by XOR encryption, and so on. This scheme is both efficient and highly secure.



Each stage is easily reversible. For the Substitute Byte, ShiftRows, and MixColumns stages, an inverse function is used in the decryption algorithm. For the AddRoundKey stage, the inverse is achieved by XORing the same round key to the block, using the result that

1. As with most block ciphers, the decryption algorithm makes use of the expanded key in reverse order. However, the decryption algorithm is not identical to the encryption algorithm. This is a consequence of the particular structure of AES.
2. Once it is established that all four stages are reversible, it is easy to verify that decryption does recover the plaintext. AES structure figure lays out encryption and decryption going in opposite vertical directions. At each horizontal point (e.g., the dashed line in the figure), State is the same for both encryption and decryption.

9. The final round of both encryption and decryption consists of only three stages. Again, this is a consequence of the particular structure of AES and is required to make the cipher reversible.

**3.4.Blowfish**

Blowfish was developed in 1993 by Bruce Schneier, an independent consultant and cryptographer, and quickly became one of the most popular alternatives to DES.

* Blowfish is easy to implement and it has a high execution speed. It is also a very compact algorithm that can run in less than 5K of memory.
* The key length of Blowfish is variable and can be as long as 448 bits, In practice, 128-bit keys are

used. Blow-fish uses 16 rounds.

Blowfish uses S-boxes and the XOR function, similar to DES, but also uses binary addition. Unlike DES, which uses fixed S-boxes, Blowfish uses dynamic S-boxes that are generated as a function of the key.

 In Blowfish, the subkeys and the S-boxes are generated by repeated application of the Blowfish algorithm itself to the key. Blowfish encryption algorithm requires a total of 521 executions to produce the subkeys and S-boxes. As a result, Blowfish is not suitable for applications in which the secret key changes frequently.

Blowfish is one of the most difficult symmetric encryption algorithms so far implemented, because both the subkeys and the S-boxes are produced by a process of repeated applications of Blowfish itself, which thoroughly mangles (crush) the bits and makes cryptanalysis very difficult. So far, no practica l weaknesses have been found in Blowfish cryptanalysis. Blowfish is used in a number of commercial applications.

**3.5. Differential and Linear Cryptanalysis**

Differential cryptanalysis and Linear cryptanalysis. Both make use of Known plaintext -ciphertext pairs and try to attack the round structure and the S-Boxes. Recent advancements showed that using Differential cryptanalysis, DES can be broken using 2 47 plaintext-ciphertext pairs and for linear cryptanalysis, the number is even reduced to 2 41.

One of the most significant advances in cryptanalysis in recent years is differential cryptanalysis.

**Differential Cryptanalysis**

Differential cryptanalysis is the first published attack that is capable of breaking DES in less than 255 complexity The scheme, as reported in [BIHA93], can successfully cryptanalyze DES with an effort on the order of 2 47 encryptions, requiring 2 47 chosen plaintexts. Although 2 47 is certainly significantly less than 255 the need for the adversary to find 247 chosen plaintexts makes this attack of only theoretical interest.

**Linear Cryptanalysis**

A more recent development is linear cryptanalysis, described in [MATS93]. This attack is based on finding linear approximations to describe the transformations performed in DES. This method can f ind a DES key given 243 known plaintexts, as compared to 2 47 chosen plaintexts for differential cryptanalysis. Although this is a minor improvement, because it may be easier to acquire known plaintext rather than chosen plaintext, it still leaves linear cryptanalysis infeasible as an attack on DES. So far, little work has been done by other groups to validate the linear cryptanalytic approach.

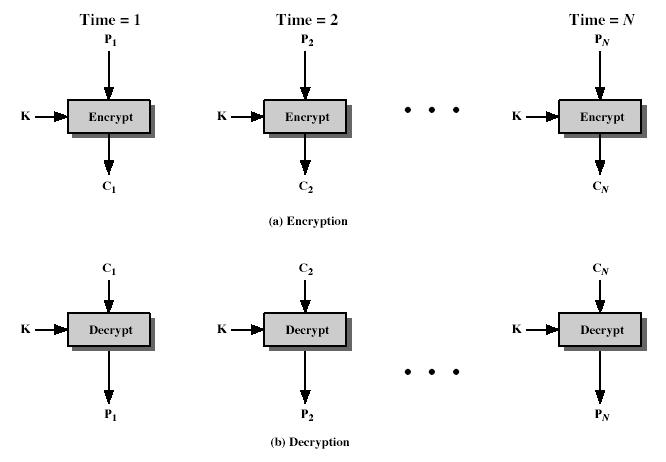
**Avalanche Effect:** An effect in DES and other secret key ciphers where each small change in plaintextimplies that somewhere around half the cipher text changes. The avalanche effect makes it harder to

successfully cryptanalyze the cipher text. DES exhibits a strong Avalanche effect. Concern about the strength of DES falls into two categories i.e. strength of algorith m itself and use of 56- bit key. Though many attempts were made over the years to find and exploit weaknesses in the algorithm, none of them were successful in discovering any fatal weakness in DES. A serious concern is with the key size as the time passed the security in DES became getting compromised by the advent of supercomputers which succeeded in breaking the DES quickly using a brute-force attack. If the only form of attack that could be made on an encryption algorithm is brute force, the way of countering it is obviously using long keys. If a key of size 128 bits is used, it takes approximately 1018 years to break the code making the algorithm unbreakable by brute-force approach. The two analytical attacks on DES are

**3.6. Block Cipher Modes Of Operation**

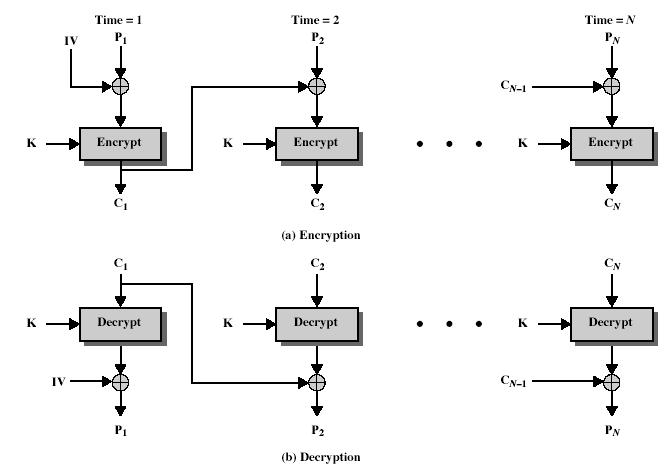
To apply a block cipher in a variety of applications, four “modes of operation” have been defined by NIST (FIPS

81). The four modes are intended to cover virtually all the possible applications of encryption for which a block cipher could be used. As new applications and requirements have appeared, NIST has expanded the list of recommended modes to five in Special Publication 800-38A. These modes are intended for use with any symmetric block cipher, including triple DES and AES. **Electronic Codebook Book (ECB)** The simplest mode is the electronic codebook (ECB) mode, in which plaintext is handled one block at a time and each block of plaintext is encrypted using the same key. *ECB is the simplest of the modes, and is used when only a single block* *of info needs to be sent.*



Break the plaintext into 64-bit blocks and encrypt each of them with the same key. The last block should be padded to 64-bit if it is shorter. Same block and same key always yields same cipher block. Each block is a value which is substituted, like a codebook, hence the name Electronic Code Book. Each block is encoded independently of the other blocks. **Ci = DESK1(Pi)** ECB is not appropriate for any quantity of data, since repetitions can be seen, esp. with graphics, and because the blocks can be shuffled/inserted without affecting the en/decryption of each block. Its main use is to send one or a very few blocks, eg a session encryption key.

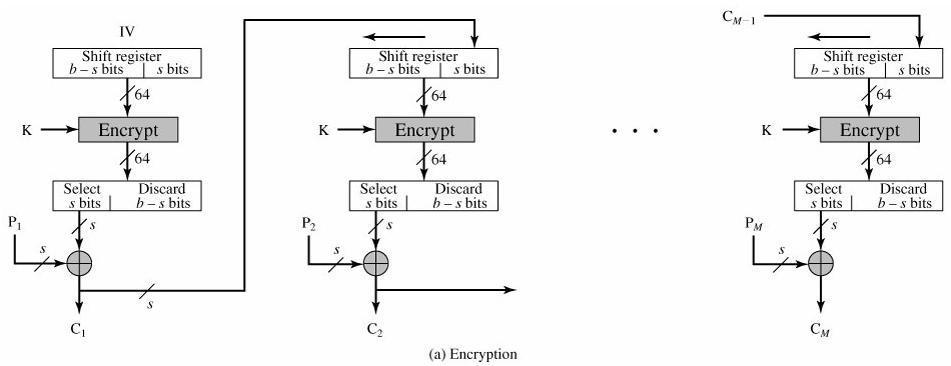
**Cipher Block Chaining Mode (CBC**) To overcome the problems of repetitions and order independence in ECB,want some way of making the ciphertext dependent on **all** blocks before it. This is what CBC gives us, by combining the previous ciphertext block with the current message block before encrypting. To start the process, use an Initial Value (IV), which is usually well known (often all 0's), or otherwise is sent, ECB encrypted, just before starting CBC use.

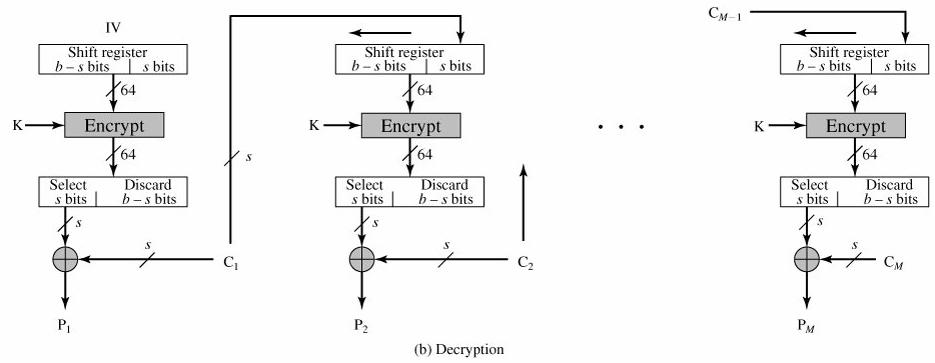


All cipher blocks will be chained so that if one is modified, the ciphertext cannot be decrypted correctly. Each plaintext block is XORed with the previous cipher block before encryption, hence the name CBC. The first plaintext block is XORed with an initialization vector IV, which is to be protected securely, (e.g., send it encrypted in ECB mode). **Ci = DESK1(Pi XOR Ci-1)** CBC is the block mode generally used. The chaining provides an avalanche effect, which means the encrypted message cannot be changed or rearranged without totally destroying the subsequent data. However there is the issue of ensuring that the IV is either fixed or sent encrypted in ECB mode to stop attacks on 1st block. **Cipher Feed Back Mode (CFB)**

If the data is only available a bit/byte at a time (eg. terminal session, sensor value etc), then must use some other approach to encrypting it, so as not to delay the info. it is possible to convert DES into a stream cipher, using either the cipher feedback (CFB) or the output feedback mode. A stream cipher eliminates the need to pad a message to be an integral number of blocks. It also can operate in real time. Thus, if a character stream is being transmitted, each character can be encrypted and transmitted immediately using a character-oriented stream cipher.

One desirable property of a stream cipher is that the ciphertext be of the same length as the plaintext. Thus, if 8-bit characters are being transmitted, each character should be encrypted to produce a cipher text output of 8 bits. If more than 8 bits are produced, transmission capacity is wasted.



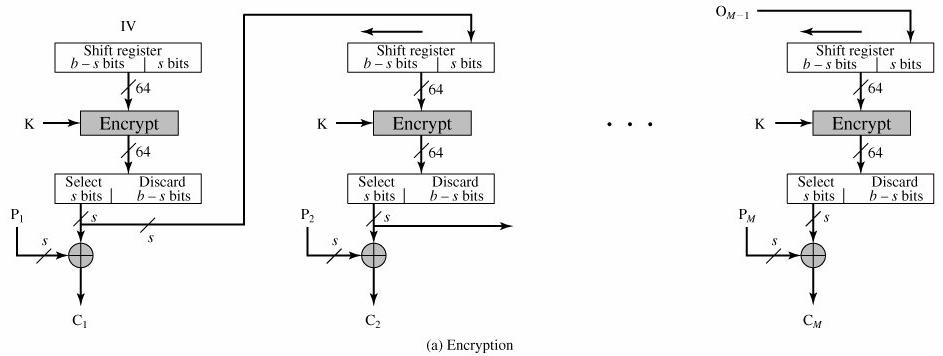


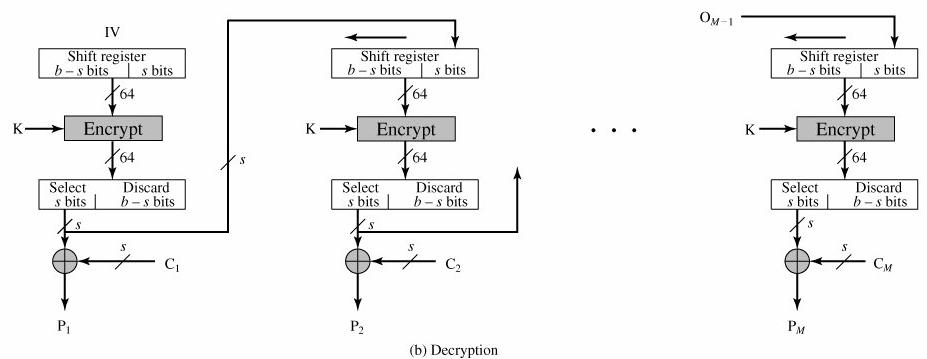
The input to the encryption function is a b-bit shift register that is initially set to some initialization vector (IV). The leftmost (most significant) s bits of the output of the encryption function are XORed with the first segment of plaintext P1 to produce the first unit of ciphertext C1, which is then transmitted. In addition, the contents of the shift register are shifted left by s bits and C1 is placed in the rightmost (least significant) s bits of the shift register. This process continues until all plaintext units have been encrypted. For decryption, the same scheme is used, except that the received ciphertext unit is XORed with the output of the encryption function to produce the plaintext unit. Note that it is the *encryption* function that is used, not the decryption function. **Ci = Pi XOR DES**

**K1(Ci-1)**

CFB is the usual stream mode. As long as can keep up with the input, doing encryptions every 8 bytes. A possible problem is that if its used over a "noisy" link, then any corrupted bit will destroy. values in the current and next blocks (since the current block feeds as input to create the random bits for the next). So either must use over a reliable network transport layer (pretty usual) or use OFB.

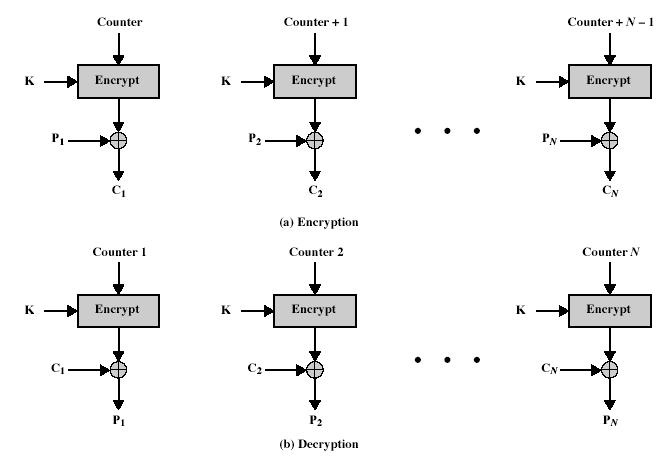
**Output Feedback Mode (OFB)** The output feedback (OFB) mode is similar in structure to that of CFB. It is theoutput of the encryption function that is fed back to the shift register in OFB, whereas in CFB the ciphertext unit is fed back to the shift register.





Keystream is independent of the data and can be computed in advance. **Ci = Pi XOR Oi Oi = DESK1(Oi-1)** Here the generation of the "random" bits is independent of the message being encrypted. The advantage is that firstly, they can be computed in advance, good for bursty traffic, and secondly, any bit error only affects a single bit. Thus this is good for noisy links (eg satellite TV transmissions etc). The disadvantage of OFB is that it is more vulnerable to a message stream modification attack than is CFB.

**Counter Mode (CTR)** The Counter (CTR) mode is a variant of OFB, but which encrypts a counter value (hencename). Although it was proposed many years before, it has only recently been standardized for use with AES along with the other existing 4 modes. It is being used with applications in ATM (asynchronous transfer mode) network security and IPSec (IP security). All modes of operations except ECB make random access to the file impossible: to access data at the end of the file one has to decrypt everything. Plaintext is not encrypted directly. IV plus a constant is encrypted and the resulting ciphertext is XORed with the plaintext – add 1 to IV in each step.



If the same IV is used twice with the same key, then cryptanalyst may XOR the ciphers to get the XOR of the plaintexts –this could be used in an attack. A counter, equal to the plaintext block size is used. The only requirement stated in SP 800-38A is that the counter value must be different for each plaintext block that is

encrypted. Typically the counter is initialized to some value and then incremented by 1 for each subsequent block. CTR mode has a number of advantages in parallel h/w & s/w efficiency, can preprocess the output values in advance of needing to encrypt, can get random access to encrypted data blocks, and is simple. But like OFB have issue of not reusing the same key + counter value.

**3.7. Stream ciphers**

**S t r e a m C i p h e r S t r u c t u r e**

A *stream cipher* processes the input elements continuously, producing output one element at a time, as it goes along.

A stream cipher encrypts one byte of plaintext at a time, although a stream cipher is designed to operate on one bit at a time or on units larger than a byte at a time. The following Fig 2.7 shows the stream cipher structure.

The input is a key to a pseudorandom bit generator that produces a stream of 8-bit numbers that are apparently (actually) random. We cannot predict a pseudorandom stream without the knowledge of the input key and it has an apparently random character.

* The output of the generator is called a key stream. It is combined one byte at a time with the plaintext stream using the bitwise XOR operation.
* For ex: if the next byte generated by the generator is 01101100 and the next plaintext byte is 11001100, then the resulting ciphertext byte is:

|  |  |
| --- | --- |
| 11001100 | plaintext |
| 01101100 | key stream |
| 10100000 | ciphertext |

Decryption requires the same pseudorandom sequence:

|  |  |
| --- | --- |
| 10100000 | ciphertext |
| 01101100 | key stream |
| 11001100 | plaintext |

The following are the important considerations for stream cipher:

1. The encryption sequence should have a larger period. The longer the period of repeat the more difficult it will be to do cryptanalysis.
2. The key stream should be as random as possible. The more random the key stream, it will be more difficult for cryptanalysis.
3. The output of the pseudorandom number generator depends on the value of the input key. To protect against brute-force attacks, the key needs to be sufficiently long. A key length of at least 128 bits is

required.

If the pseudorandom number generator is properly designed, a stream cipher can be as secure as block cipher. The main advantage of stream cipher is that stream ciphers are almost always faster and use far less code than block ciphers.

The advantage of block cipher is that they keys can be reused.

A stream cipher is a better alternative for applications that require encryption/decryption of a stream of data, such as over a data comm’tn channel or a browser/Web link.

**3.8. RC4**

RC4 was developed in 1994 by Ron Rivest, one of the inventors of the public -key algorithm RSA. RC4

is defined in RFC 2040 and was designed to have the following characteristics:

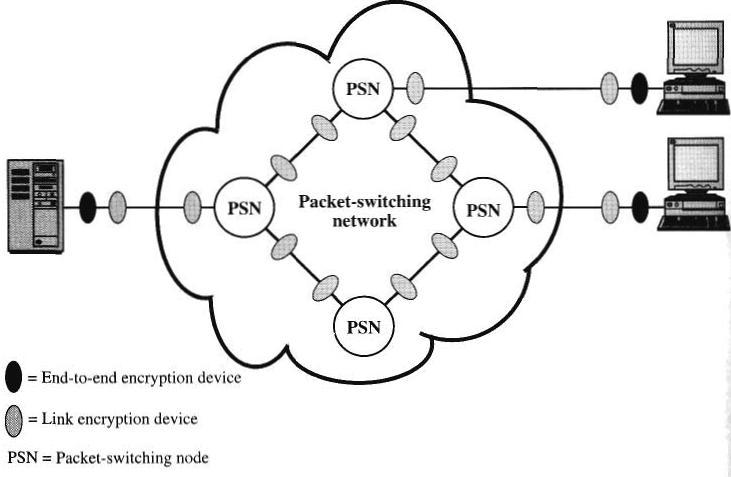
* **Suitable for hardware or software:** RC4 uses only primitive (old) computational operationscommonly found on microprocessors.
* **Fast:** To achieve this, RC4 is a simple algorithm and is word oriented. The basic operations workon full words of data at a time.
* **Adaptable to processors of different word lengths:** The number of bits in a word is a parameter(constraint) of RC4: different word lengths yield different algorithms.
* **Variable number of rounds:** The number of rounds is a second parameter of RC4*.*This parameterallows a tradeoff between higher speed and higher security.
* **Variable-length key:** The key length is a third parameter of RC4. Again, this allows a tradeoffbetween speed and security.
* **Simple:** RC4’ssimple structure is easy to implement and simplifies the task of determining thestrength of the algorithm.
* **Low memory requirement:** A low memory requirement makes RC4 suitable for smart cards andother devices with restricted memory.
* **High security:** RC4 is intended (proposed) to provide high security with suitable parameters.
* **Data-dependent rotations:** RC4 incorporates rotations (circular bit shifts) whose amount is data

dependent. This appears to strengthen the algorithm against cryptanalysis. RC4 is used in a number of products from RSA Data Security. Inc.

**3.9.Placement of encryption function**

Encryption is the most powerful, and most common, approach to counter the threats to n/w security. When using encryption, we need to decide what to encrypt and where the encryption gear (equipment or mechanism) should be located.

 There are two fundamental alternatives: link encryption and end-to-end encryption: these are shown in Fig 2.11 using a packet-switching n/w.



**Fig 2.11: Encryption across a Packet-Switching N/w**



With **link encryption**, each vulnerable (susceptible) comm’tns link is equipped on both ends with an encryption device. Thus, all traffic over all comm’tns links is secured. Although this requires a lot of encryption devices in a large network, it provides a high level of security.

 One disadvantage of this approach is that the message must be decrypted each time it enters a packet switch: this is b’coz the switch must read the address (virtual circuit num) in the packet header to route the packet. Thus, the message is vulnerable (at risk) at each switch. If this is a public packet-switching n/w, the user has no control over the security of the nodes.



With **end-to-end encryption**, the encryption process is carried out at the two end systems.

* The source host or terminal encrypts the data. The data, in encrypted form, is then transmitted without any alteration across the n/w to the destination terminal or host. The destination shares a key with the source and so is able to decrypt the data.
* This approach seems to provide secure transmission against attacks on the n/w links or switches. But, still there is a weak spot.



Consider the following situation. A host connects to an X.25 packet -switching n/w, sets up a virtual circuit to another host, and is prepared to transfer data to that other host using end-to-end encryption.

* Data is transmitted over such a n/w in the form of packets, consisting of a header and some user data. What part of each packet will the host encrypt?
* Suppose that the host encrypts the entire packet, including the header. But, this will not work b’coz, only the other host can perform the decryption. The packet-switching node will receive an encrypted packet and it will be unable to read the header. Therefore, it will not be able to route the packet.
* So the host may only encrypt the user data portion of the packet and must leave the header in the clear, so that it can be read by the n/w.



Thus, with end-to-end encryption, the user data is secure. But, the traffic pattern is not, b’coz, packet headers are transmitted in the clear.

 So to achieve greater security, both link and end -to-end encryptions are needed, shown in Fig 2.11.



To summarize, when both techniques are used, the host encrypts the user data portion of a packet using an end-to-end encryption key. The entire packet is then encrypted using a link encryption key.  As the packet traverses (pass thru) the n/w, each switch decrypts the packet using a link

encryption key to read the header and then encrypts the entire packet again for s ending it out on the next link.

 Now the entire packet is secure except when the packet is actually in the memory of a packet switch, at which time the packet header is in the clear.

**3.10. KEY DISTRIBUTION**



For symmetric encryption to work, the two parties who are doing a secure exchng must have the same key, and that key must be protected from access by others.

* Therefore, the strength of any cryptographic system rests with the key distribution technique .
* **Key Distribution** refers to the means of delivering a key to two parties that wish to exchange data,without allowing others to see the key.
* Key distribution can be achieved in a number of ways. For two parties A and B,

1. A key can be selected by A and physically delivered to B.
2. A third party can select the key and physically deliver it to A and B.
3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
4. If A and B each have an encrypted connection to a third party C, C c an deliver a key on the

encrypted links to A and B.



Options 1 and 2 call for manual delivery of a key. This is a reasonable requirement for link encryption, b’coz each link encryption device is only going to exchange data with its partner on the other end of the link.



Option 3 can be used for either link encryption or end-to-end encryption, but if an attacker succeeds in gaining access to one key, then all subsequent keys are revealed.

* Even if frequent changes are made to the link encryption keys , these should be done manually.
* To provide keys for end-to-end encryption, option 4 is preferable.



**Fig 2.12 Automatic Key Distribution for Connection-Oriented Protocol**



Fig 2.12 shows an implementation that satisfies option 4 for end -to-end encryption. Link encryption is ignored in the fig. This can be added, or not, as required. For this scheme, two kinds of keys are identified:

* + **Session key:** When two end systems (hosts, terminals, etc.) wish to commu nicate, theyestablish a logical connection (e.g., virtual circuit).

o For the duration of that logical connection, all user data is encrypted with a one-time session key.

o At the conclusion of the session, or connection, the session key is destroyed.

* + **Permanent key:** A permanent key is a key used between entities for the purpose ofdistributing session keys.
* The configuration consists of the following elements:
  + **Key distribution center:** The key distribution center (KDC) determines which systems areallowed to communicate with each other.

o When permission is granted for two systems to establish a connection, the key distribution center provides a one-time session key for that connection.

* + **Security service Module (SSM)*:*** The SSM performs end-to-end encryption and obtainssession keys on behalf of its host or terminal or users.



The steps involved in establishing a connection are shown in Figure 2.12.

* **Step 1:** When one host wishes to set up a connection to another host, it transmits a connection-request packet.
* **Step 2:** The SSM saves that packet and applies to the KDC for permission, to establish theconnection.
* **Step 3:** The communication between the SSM and the KDC is encrypted using a master keyshared only by the SSM and the KDC. If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM.
* **Step 4:** The requesting SSM can now release the connection request packet, and a connection is setup, b/w the two end systems. All user data exchanged between the two end systems are encrypted

by their respective SSMs using the one-time session key.



The automated key distribution approach provides the flexibility and dynamic characteristics. It allows a num of terminal users to access a num of hosts and for the hosts to exchange data with each other.

**4.Asymmetric Key Ciphers**

**4.1. Public key cryptography principles**

Public key encryption is as important as conventional encryption. Public key encryption is used in message authentication and key distribution.

**Public-Key Encryption Structure**

* Public-key encryption was first proposed by Diffie and Hellman in 1976.
* Public-key algorithms are based on mathematical functions rather than on simple operations on bit patterns.
* Public-key cryptography is asymmetric, i.e., it involves the use of two separate keys.
* The use of two keys has a strong effect in the areas of confidentiality, key distribution, and authentication.
* There are a few common misconceptions about public-key encryption.
* One is it is believed that public-key encryption is more secure from cryptanalysis than conventional encryption. But the fact is, the security of any encryption scheme depends on (1) the length of the key and
  1. the computational work involved in breaking a cipher. It does not depend on whether we are using either conventional or public-key encryption that makes one superior (better) to another from resisting cryptanalysis.
* A second misconception is that public-key encryption is a general-purpose technique that made conventional encryption obsolete (out of date). On the contrary, because of the computational overhead of current public-key encryption schemes, there are no expected chances that conventional encryption will be discarded.
* Finally, there is a feeling that key distribution is trivial (simple) when using public-key encryption, compared to the process involved with key distribution centers for conventional encryption. In fact, some form of protocol is needed, which frequently involves a central agent, and the procedures involved are not simpler or more efficient than those required for conventional encryption.

A public-key encryption scheme has six ingredients (Fig 3.1a):

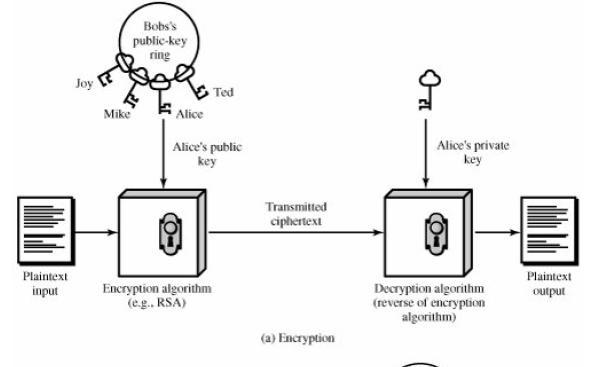


Fig 3.1a: Public-Key Cryptography (Encryption)

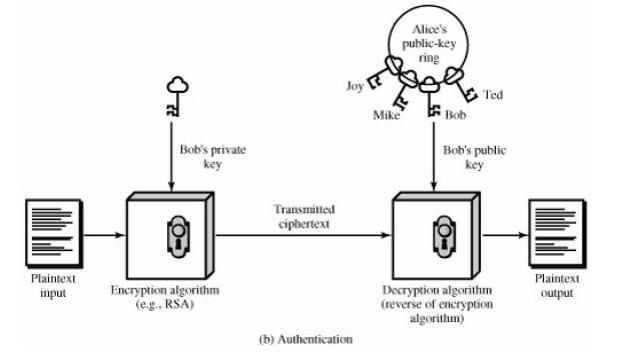


Fig Public-Key Cryptography (Authentication)

**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 are selected so that if one is used for encryption, the other isused 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 agiven message, two different keys will produce two different ciphertexts.

**Decryption algorithm**: This algorithm accepts the ciphertext and the matching key and produces the originalplaintext.

As the names suggest, the public key 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 following are the essential steps**:

Each user generates a pair of keys to be used for the encryption and decryption of messages.

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 shown in fig 3.7a, each user maintains a collection of public keys obtained from others.

If Bob wishes to send a private message to Alice, Bob encrypts the message using Alice's public key.

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 will have access to public keys, and private keys are generated locally by each participant and they are not distributed.

The incoming comm’tion is secure as long as a user protects his or her private key. At any time, a user can change the private key and publish the companion public key to replace the old public key.

The key used in conventional encryption is referred as a secret key.

The two keys used for public-key encryption are referred to as the public key and the private key. The private key is always kept secret, but it is referred as a private key instead of a secret key to avoid confusion with conventional encryption.

**Applications for Public-Key Cryptosystems**

Public-key systems are 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.

The use of public-key cryptosystems are classified into three categories: Encryption/decryption: The sender encrypts a message with the recipient's public key.

Digital signature: The sender "signs" a message with its private key. Signing is achieved by a cryptographic algorithm applied to the message or to a small block of data that is a function of the message.

Key exchange: Two sides cooperate to exchange a session key by using different approaches involving the private key(s) of one or both parties.

Some algorithms are suitable for all three applications, whereas others can be used only for one or two of these applications.

Table 3.1 shows the applications supported by the algorithms, RSA and Diffie Hellman, the Digital Signature Standard (DSS) and elliptic-curve cryptography.

|  |  |  |  |
| --- | --- | --- | --- |
| Algorithm | Encryption/Decryption | Digital Signature | Key Exchange |
| RSA | Yes | Yes | Yes |
|  |  |  |  |
| Diffie-Hellman | No | No | Yes |
|  |  |  |  |
| DSS | No | Yes | No |
|  |  |  |  |
| Elliptic Curve | Yes | Yes | Yes |

Table 3.1: Applications for Public-Key Cryptosystems Requirements for Public-Key Cryptography

The cryptosystem shown in Figure 3.1 depends on a cryptographic algorithm based on two related keys. This system was suggested by Diffie and Hellman, but they did not give show any demonstration to the existence of such algorithms.

But, they have give some conditions which need to be fulfilled by such algorithms:

It is computationally easy for a party B to generate a pair (public key PUb, private key PRb).

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)

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)].

It is computationally infeasible for an opponent, knowing the public key, PUb, to determine the private key, PRb. It is computationally infeasible for an opponent, knowing the public key, PUb, and a ciphertext, C, to recover the original message, M.

Either of the two related keys can be used for encryption, with the other used for decryption.

But the sixth requirement is not necessary for all public-key applications.

M = D[PUb, E(PRb, M)] = D[PRb, E(PUb, M)]

PUBLIC KEY CRYPTOGRAPHY ALGORITHMS

The two most widely used public-key algorithms are RSA and Diffie-Hellman.

**4.2. RSA**

RSA is the best known, and by far the most widely used general public key encryption algorithm, and was first published by Rivest, Shamir & Adleman of MIT in 1978 [RIVE78]. Since that time RSA has reigned supreme as the most widely accepted and implemented general-purpose approach to public-key encryption. The RSA scheme is a block cipher in which the plaintext and the ciphertext are integers between 0 and n-1 for some fixed n and typical size for n is 1024 bits (or 309 decimal digits). It is based on exponentiation in a finite (Galois) field over integers modulo a prime, using large integers (eg. 1024 bits). Its security is due to the cost of factoring large numbers.

RSA involves a public-key and a private-key where the public key is known to all and is used to encrypt data or message. The data or message which has been encrypted using a public key can only be decryted by using its corresponding private-key. Each user generates a key pair i.e. public and private key using the following steps: *each user selects two large primes at random - p, q*

*compute their system modulus n=p.q calculate ø(n), where* ***ø(n)=(p-1)(q-1)***

*selecting at random the encryption key e, where 1<e<ø(n),and* ***gcd(e,ø(n))=1*** *solve following equation to find decryption key d:* ***e.d=1 mod ø(n)*** *and 0≤d≤n publish their public encryption key:* ***KU={e,n}***

*keep secret private decryption key****: KR={d,n}***

Both the sender and receiver must know the values of n and e, and only the receiver knows the value of d. Encryption and Decryption are done using the following equations.

To encrypt a message M the sender:

– obtains **public key** of recipient *KU={e,n}*

– computes: **C=Me mod n**, where 0≤M<n

To decrypt the ciphertext C the owner:

– uses their private key *KR={d,n}*

– computes: **M=Cd mod n = (Me) d mod n = Med mod n**

For this algorithm to be satisfactory, the following requirements are to be met.

1. Its possible to find values of e, d, n such that **Med = M mod n** for all M<n
2. It is relatively easy to calculate Me and C for all values of M < n.

**c)** It is impossible to determine d given e and n

The way RSA works is based on Number theory: **Fermat’s little theorem**: if **p** is prime and **a** is positive integer not divisible by **p**, then **ap-1** ≡ **1 mod p. Corollary**: For any positive integer **a** and prime **p**, **ap** ≡ **a mod p.**

Fermat’s theorem, as useful as will turn out to be does not provide us with integers d,e we are looking for – Euler’s theorem (a refinement of Fermat’s) does. Euler’s function associates to any positive integer **n,** a number

φ**(n)**: the number of positive integers smaller than **n** and relatively prime to **n.** For example, φ**(37) = 36** i.e. φ**(p) =** **p-1** for any prime **p.** For any two primes **p,q**, φ**(pq)=(p-1)(q-1). Euler’s theorem**: for any relatively primeintegers a,n we have **a**φ**(n)**≡**1 mod n. Corollary: F**or any integers a,n we have **a**φ**(n)+1**≡**a mod n Corollary:** Let p,q be two odd primes and n=pq. Then: φ(n)=(p-1)(q-1) For any integer m with 0<m<n, m(p-1)(q-1)+1 ≡ m mod n For any integers k,m with 0<m<n, mk(p-1)(q-1)+1 ≡ m mod n Euler’s theorem provides us the numbers d, e such that Med=M mod n. We have to choose d,e such that ed=kφ(n)+1, or equivalently, d≡e-1mod φ(n) An example of RSA can be given as,

*p*=17 & *q*=11 *n* = *pq* =17×11=187

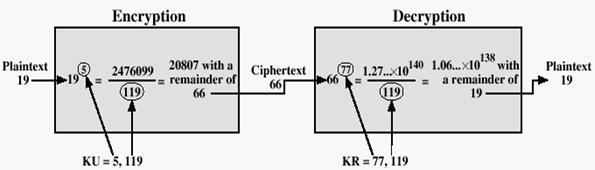
Compute ø(*n*)=(*p–*1)(*q-*1)=16×10=160 Select e *:* gcd(e,160)=1; choose *e*=7

Determine d*: de=*1 mod 160 and *d* < 160 Value is d=23 since 23×7=161= 10×160+1 Publish public key KU={7,187}

Keep secret private key KR={23,187} Now, given message M = 88 (nb. 88<187) encryption: C = 887 mod 187 = 11 decryption: M = 1123 mod 187 = 88

Another example of RSA is given as, Let **p = 11, q = 13, e = 11, m = 7 n = pq** i.e. n= 11\*13 = 143 **ø(n)= (p-1)(q-1)** i.e. (11-1)(13-1) = 120 ***e.d=1 mod ø(n) i.e.*** 11d mod 120 = 1 i.e. (11\*11) mod 120=1; so d = 11 public key:{11,143} and private key: {11,143} **C=Me mod n**, so ciphertext = 711mod143 = 727833 mod 143; i.e. C = 106 **M=Cd mod n,** plaintext = 10611 mod 143 = 1008 mod 143; i.e. M = 7

Another example is:



For RSA key generation,

– determine two primes at random - p, q

– select either e or d and compute the other

– means must be sufficiently large

– typically guess and use probabilistic test

onents e, d are inverses, so use Inverse algorithm to compute the other

**Security of RSA** There are three main approaches of attacking RSA algorithm. **Brute force key search** (infeasible given size of numbers) As explained before, involves trying all possible private keys. Best defence is

using large keys. **Mathematical attacks** (based on difficulty of computing ø(N), by factoring modulus N) There are several approaches, all equivalent in effect to factoring the product of two primes. Some of them are given as:

– factor N=p.q, hence find ø(N) and then d

– determine ø(N) directly and find d

– find d directly

The possible defense would be using large keys and also choosing large numbers for p and q, which should differ only by a few bits and are also on the order of magnitude 1075 to 10100. And gcd (p-1, q-1) should be small. **Timing attacks** (on running of decryption) These attacks involve determination of a private key by keeping trackof how long a computer takes to decipher a message (ciphertext-only attack) –this is essentially an attack on the fast exponentiation algorithm but can be adapted for any other algorithm. Though these attacks are a quite serious threat, there are some simple countermeasures that can be used. They are explained below: *Constant* *exponentiation time*:- Ensure that all exponentiations take the same time before returning a result: degradeperformance of the algorithm. *Random Delay*:- A random delay can be added to exponentiation algorithm to confuse the timing attack. If there is not enough noise added by defenders, the attackers can succeed. *Blinding:*-Multiply the ciphertext by a **random** number before performing exponentiation –in this way the attacker does not know the input to the exponentiation algorithm. RSA Data Security incorporates a blinding feature into some of its products. The private-key operation M = Cd mod n is implemented as follows:

Generate a secret random number r between 0 and n-1

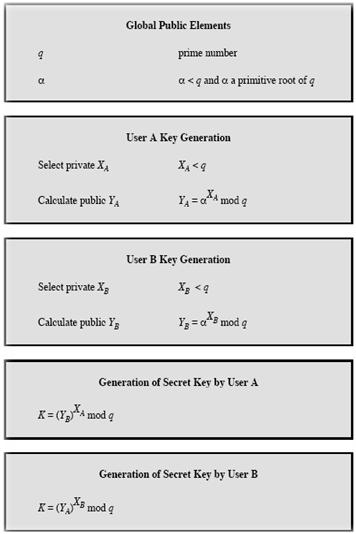
-1mod n

1. **Diffie – Hellman**

**Diffie-Hellman key exchange** (**D-H**) is a cryptographic protocol that allows two parties that have no priorknowledge of each other to jointly establish a shared secret key over an insecure communications channel. This key can then be used to encrypt subsequent communications using a symmetric key cipher. The D-H algorithm depends for its effectiveness on the difficulty of computing discrete logarithms.

First, a primitive root of a prime number p, can be defined as one whose powers generate all the integers from 1 to p-1. If a is a primitive root of the prime number p, then the numbers, ***a mod p, a2 mod p,..., ap-1 mod p***, are distinct and consist of the integers from 1 through p 1 in some permutation.

For any integer ***b*** and a primitive root ***a*** of prime number ***p***, we can find a unique exponent **i** such that .The exponent i is referred to as the discrete logarithm of b for the base a, mod p. We express this value as ***dloga,p (b).*** The algorithm is summarized below:



For this scheme, there are two publicly known numbers: a prime number q and an integer α that is a primitive root of q. Suppose the users A and B wish to exchange a key. User A selects a random integer XA < q and computes

YA = αXA mod q. Similarly, user B independently selects a random integer XA < q and computes YB = αXB mod q. Each side keeps the X value private and makes the Y value available publicly to the other side. User A computes the key as K = (YB)XA mod q and user B computes the key as K = (YA)XB mod q. These two calculations produce identical results.

*Discrete Log Problem* The (discrete) exponentiation problem is as follows: Given a base a, an exponent b and amodulus p, calculate c such that ab ≡ c (mod p) and 0 ≤ c < p. It turns out that this problem is fairly easy and can be calculated "quickly" using fast-exponentiation. The discrete log problem is the inverse problem: Given a base a, a result c (0 ≤ c < p) and a modulus p, calculate the exponent b such that ab ≡ c (mod p). It turns out that no one has found a quick way to solve this problem With DLP, if P had 300 digits, Xa and Xb have more than 100 digits, it would take longer than the life of the universe to crack the method. **Examples for D-H key distribution** **scheme:**

**1)** Let p = 37 and g = 13.

Let Alice pick a = 10. Alice calculates 1310 (mod 37) which is 4 and sends that to Bob. Let Bob pick b = 7. Bob calculates 137 (mod 37) which is 32 and sends that to Alice. (Note: 6 and 7 are secret to Alice and Bob, respectively, but both 4 and 32 are known by all.)

**2)** Let p = 47 and g = 5. Let Alice pick a = 18. Alice calculates 518 (mod 47) which is 2 and sends that to Bob.Let Bob pick b = 22. Bob calculates 522 (mod 47) which is 28 and sends that to Alice.

ey

**Man-in-the-Middle Attack on D-H protocol** Suppose Alice and Bob wish to exchange keys, and Darth is theadversary. The attack proceeds as follows:

1. Darth prepares for the attack by generating two random private keys XD1 and XD2 and then computing the corresponding public keys YD1 and YD2.
2. Alice transmits YA to Bob.
3. Darth intercepts YA and transmits YD1 to Bob. Darth also calculates K2 = (YA)XD2mod q.
4. Bob receives YD1 and calculates K1 = (YD1)XE mod q.
5. Bob transmits XA to Alice.
6. Darth intercepts XA and transmits YD2 to Alice. Darth calculates K1 = (YB)XD1 mod q.
7. Alice receives YD2 and calculates K2 = (YD2)XA mod q.

At this point, Bob and Alice think that they share a secret key, but instead Bob and Darth share secret key K1 and Alice and Darth share secret key K2. All future communication between Bob and Alice is compromised in the following way:

1. Alice sends an encrypted message M: E(K2, M).
2. Darth intercepts the encrypted message and decrypts it, to recover M.
3. Darth sends Bob E(K1, M) or E(K1, M'), where M' is any message. In the first case, Darth simply wants to eavesdrop on the communication without altering it. In the second case, Darth wants to modify the message going to Bob.

The key exchange protocol is vulnerable to such an attack because it does not authenticate the participants. This vulnerability can be overcome with the use of digital signatures and public-key certificates.

1. **ECC**

**Elliptic curve cryptography (ECC)** is an approach to public-key cryptography based on the algebraic structureof elliptic curves over finite fields. The use of elliptic curves in cryptography was suggested independently by Neal Koblitz and Victor S. Miller in 1985. The principal attraction of ECC compared to RSA is that it appears to offer equal security for a far smaller bit size, thereby reducing the processing overhead. **Elliptic Curve over** **GF(p) *4a3 + 27b2 0 (mod p).*** An

***y2***  ***x3 + ax + b (mod p),*** together with a special point, O*, called the point at infinity*. Let P and Q be two points onE(a,b)(GF(p)) and O is the point at infinity.

* P+O = O+P = P
* If P = (x1,y1) then -P = (x1 ,-y1) and P + (-P) = O.
* If P = (x1,y1) and Q = (x2,y2), and P and Q are not O.

then P +Q = (x3 ,y3) where ***x3 =*** ***2 - x1 - x2 y3 = (x1 - x3) - y1 and*** ***= (y2-y1)/(x2-x1) if P ≠ Q*** ***=***

***(3x12+a)/ 2y1 if P = Q***

An elliptic curve may be defined over any finite field GF(*q*). For GF(2*m*), the curve has a different form:- ***y*2 +** ***xy***

**= *x*3 + *ax*2 + *b,*** *where b 0.*

**Cryptography with Elliptic Curves**

The addition operation in ECC is the counterpart of modular multiplication in RSA, and multiple addition is the counterpart of modular exponentiation. To form a cryptographic system using elliptic curves, some kind of hard problem such as discrete logarithm or factorization of prime numbers is needed. Considering the equation, Q=kP,

where Q,P are points in an elliptic curve, it is “easy” to compute Q given k,P , but “hard” to find k given Q,P.

This is known as the elliptic curve logarithm problem. k could be so large as to make brute-force fail. **ECC Key** **Exchange** Pick a prime number p= 2180 and elliptic curve parameters a and b for the equation ***y2 x3 + ax + b* *(mod p)*** which defines the elliptic group of points Ep(a,b). Select generator point G=(x1,y1) in Ep(a,b) such thatthe smallest value for which nG=O be a very large prime number. Ep(a,b) and G are parameters of the cryptosystem known to all participants. The following steps take place:

* **A & B select private keys nA<n, nB<n**
* **compute public keys: PA=nA×G, PB=nB×G**
* **compute shared key: K=nA×PB, K=nB×PA {same since K=nA×nB×G }**

**ECC Encryption/Decryption** As with key exchange system, an encryption/decryption system requires a point Gand and elliptic group Ep(a,b) as parameters. First thing to be done is to encode the plaintext message m to be sent as an x-y point **Pm**. Each user chooses private key nA<n and computes public key PA=nA×G. To encrypt and send a message to Pm to B, A chooses a random positive integer k and produces the ciphertext **Cm** consisting of the pair of points **Cm={kG, Pm+kPb}.** here, A uses B’s public key. To decrypt the ciphertext, B multiplies the first point in the pair by B’s secret key and subtracts the result from the second point **Pm+*k*Pb** **–** **nB(*kG*) =** **Pm+*k*(nB*G*) – nB(*kG*) = Pm** A has masked the message **Pm** by adding ***k*Pb** to it. Nobody but A knows the valueof k, so even though **Pb** is a public key, nobody can remove the mask k**Pb.** For an attacker to recover the message, he has to compute k given G and kG, which is assumed hard. **Security of ECC** To **protect** a 128 bit AES key it would take a RSA Key Size of 3072 bits whereas an ECC Key Size of 256 bits.

Hence for similar security ECC offers significant computational advantages.

**Applications of ECC:**

necessary for our current cryptosystems

1. **Key Management**

One of the major roles of public-key encryption has been to address the problem of key distribution. Two distinct aspects to use of public key encryption are present.

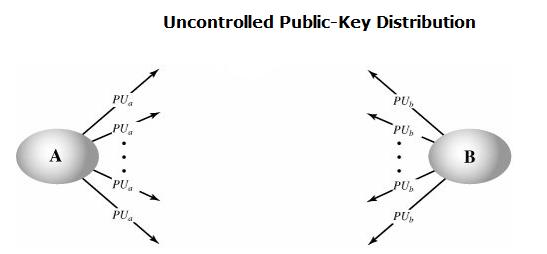
The distribution of public keys.

Use of public-key encryption to distribute secret keys.

Distribution of Public Keys The most general schemes for distribution of public keys are given below

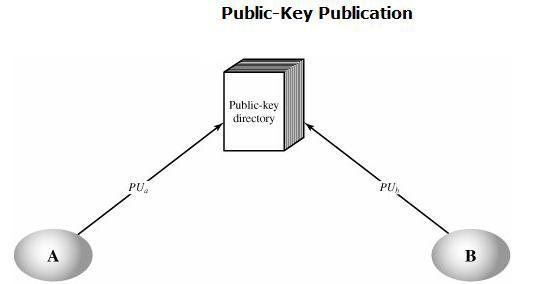
PUBLIC ANNOUNCEMENT OF PUBLIC KEYS

Here any participant can send his or her public key to any other participant or broadcast the key to the community at large. For example, many PGP users have adopted the practice of appending their public key to messages that they send to public forums.



Though this approach seems convenient, it has a major drawback. Anyone can forge such a public announcement. Some user could pretend to be user A and send a public key to another participant or broadcast such a public key. Until the time when A discovers about the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication.

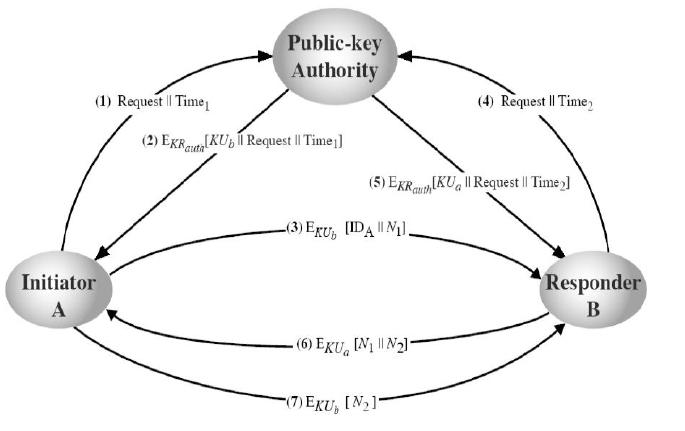
PUBLICLY AVAILABLE DIRECTORY



A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys. Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization. It includes the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way. 4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory. This scheme has still got some vulnerabilities. If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant. Or else, the adversary may tamper with the records kept by the authority.

PUBLIC-KEY AUTHORITY



Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory. This scenario assumes the existence of a public authority (whoever that may be) that maintains a dynamic directory of public keys of all users. The public authority has its own (private key, public key) that it is using to communicate to users. Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key. For example, consider that Alice and Bob wish to communicate with each other and the following steps take place and are also shown in the figure below:

1.) Alice sends a time stamped message to the central authority with a request for Bob’s public key (the time stamp is to mark the moment of The authority sends back a message encrypted with its private key (for authentication) –message contains Bob’s public key and the original message of Alice–this way Alice knows this is not a reply to an old request;

3.) Alice starts the communication to Bob by sending him an encrypted message containing her identity IDA and a nonce N1 (to identify uniquely this transaction)

4.) Bob requests Alice’s public key in the same way (step 1)

5.) Bob acquires Alice’s public key in the same way as Alice did. (Step-2)

6.) Bob replies to Alice by sending an encrypted message with N1 plus a new generated nonce N2 (to identify uniquely the transaction)

7.) Alice replies once more encrypting Bob’s nonce N2 to assure bob that its correspondent is Alice Thus, a total of seven messages are required. However, the initial four messages need be used only infrequently because both A and B can save the other's public key for future use, a technique known as caching. Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

PUBLIC-KEY CERTIFICATES

The above technique looks attractive, but still has some drawbacks. For any communication between any two users, the central authority must be consulted by both users to get the newest public keys i.e. the central authority must be online 24 hours/day. If the central authority goes offline, all secure communications get to a halt. This clearly leads to an undesirable bottleneck. A further improvement is to use certificates, which can be used to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority. A certificate binds an identity to public key, with all contents signed by a trusted Public-Key or Certificate Authority (CA). A user can present his or her public key to the authority in a secure manner, and obtain a certificate. The user can then publish the certificate. Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature. A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority. This certificate issuing scheme does have the following requirements:

1. Any participant can read a certificate to determine the name and public key of the certificate's owner.
2. Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.
   * **\* \* \* \* \* \* \* \* \* \* \* END OF UNIT II \* \* \* \* \* \* \* \* \* \* \* \* \***

**5. Message authentication algorithms and Hash Functions:**

Authentication requirements, Functions, Message authentication codes, Hash Functions, Secure Hash Functions, Whirlpool, HMAC, CMAC, digital signatures, Knapsack algorithm.

**Authentication Applications:** Kerberos, X.509 Directory Authentication Service, Public key infrastructure,Biometric Authentication

**5.1. Message Authentication Requirements**

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

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

Message authentication is a procedure to verify that received messages come from the alleged source and have not been altered. Message authentication may also verify sequencing and timeliness. A digital signature is an authentication technique that also includes measures to counter repudiation by either source or destination. Any message authentication or digital signature mechanism can be viewed as having fundamentally two levels. At the lower level, there must be some sort of function that produces an authenticator: a value to be used to authenticate a message. This lowerlevel function is then used as primitive in a higher-level authentication protocol that enables a receiver to verify the authenticity of a message. This section is concerned with the types of functions that may be used to produce an authenticator. These functions may be grouped into three classes, as follows:

1. **Message Encryption:** The ciphertext of the entire message serves as its authenticator.
2. **Message Authentication Code1 (MAC):** A public function of the message and a secret key that produces afixed length value that serves as the authenticator.
3. **Hash Functions:** A public function that maps a message of any length into a fixed length hash value, whichserves as the authenticator.

We will mainly be concerned with the last class of function however it must be noted that hash functions and

MACs are very similar except that a hash code doesn’t require a secret key. With regard to the first class, this can be seen to provide authentication by virtue of the fact that only the sender and receiver know the key. Therefore the message could only have come from the sender. However there is also the problem that the plaintext message should be recognisable as plaintext message (for example if it was some sort of digitised X-rays it mightn’t be).

1. **Message authentication codes**

A MAC, also known as a cryptographic checksum, is generated by a function C of the form MAC = C(K, M) where

M is a variable-length message,

K is a secret key shared only by sender and receiver, and C(K, M) is the fixed-length authenticator.

The MAC is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by re-computing the MAC.

**Requirements for MACs**

When an entire message is encrypted for confidentiality, using either symmetric or asymmetric encryption, the security of the scheme generally depends on the bit length of the key. Barring some weakness in the algorithm, the opponent must resort to a brute-force attack using all possible keys. On average, such an attack will require 2(k-1) attempts for a k-bit key. In particular, for a ciphertext-only attack, the opponent, given ciphertext C, would perform Pi = D(Ki, C) for all possible key values Ki until a Pi was produced that matched the form of acceptable plaintext.

In the case of a MAC, the considerations are entirely different. In general, the MAC function is a many to one function, due to the many-to-one nature of the function. Using brute-force methods, how would an opponent attempt to discover a key? If confidentiality is not employed, the opponent has access to plaintext messages and their associated MACs. Suppose k > n; that is, suppose that the key size is greater than the MAC size. Then, given a known M1 and MAC1, with MAC1 = C(K, M1), the cryptanalyst can perform MACi = C(Ki, M1) for all possible key values Ki. At least one key is guaranteed to produce a match of MACi = MAC1. Note that a total of 2k MACs will be produced, but there are only 2n < 2k different MAC values. Thus, a number of keys will produce the correct MAC and the opponent has no way of knowing which is the correct key. On average, a total of 2k/2n = 2(k-n) keys will produce a match. Thus, the opponent must iterate the attack:

● Round 1

Given: M1, MAC1 = C(K, M1)

Compute MACi = C(Ki, M1) for all 2k keys Number of matches 2(k-n)

● Round 2

Given: M2, MAC2 = C(K, M2)

Compute MACi = C(Ki, M2) for the 2(k-n) keys resulting from Round 1 Number of matches 2(k-2xn) and so on

On average, a rounds will be needed if k = a x n. For example, if an 80-bit key is used and the MAC is 32 bits long, then the first round will produce about 248 possible keys. The second round will narrow the possible keys to about 216 possibilities. The third round should produce only a single key, which must be the one used by the sender.

If the key length is less than or equal to the MAC length, then it is likely that a first round will produce a single match. It is possible that more than one key will produce such a match, in which case the opponent would need to perform the same test on a new (message, MAC) pair. Thus, a brute-force attempt to discover the authentication key is no less effort and may be more effort than that required to discover a decryption key of the same length. However, other attacks that do not require the discovery of the key are possible.

Consider the following MAC algorithm. Let M = (X1||X2||...||Xm) be a message that is treated as a concatenation of 64-bit blocks Xi. Then define

D(M) =X1 X2 ... Xm

C(K, M = E(K, D(M)) where is the exclusive-OR (XOR) operation and the encryption algorithm is DES in electronic codebook mode. Thus, the key length is 56 bits and the MAC length is 64 bits. If an opponent observes {M||C(K, M)}, a brute-force attempt to determine K will require at least 256 encryptions. But the opponent can attack the system by replacing X1 through Xm-1 with any desired values Y1 through Ym-1 and replacing Xm with Ym where Ym is calculated as follows:

Ym = Y1 Y2 ... Ym1 D(M)

The opponent can now concatenate the new message, which consists of Y1 through Ym, with the original MAC to form a message that will be accepted as authentic by the receiver. With this tactic, any message of length 64 x (m 1) bits can be fraudulently inserted.

Thus, in assessing the security of a MAC function, we need to consider the types of attacks that may be mounted against it. With that in mind, let us state the requirements for the function. Assume that an opponent knows the MAC function C but does not know K. Then the MAC function should satisfy the following requirements:

1. If an opponent observes M and C(K, M), it should be computationally infeasible for the opponent to construct a message M' such that C(K, M') = C(K, M).
2. C(K, M) should be uniformly distributed in the sense that for randomly chosen messages, M and M', the probability that C(K, M) = C(K, M') is 2n, where n is the number of bits in the MAC.
3. Let M' be equal to some known transformation on M. That is, M' = f(M). For example, f may involve inverting one or more specific bits. In that case, Pr[C(K, M) = C(K, M')] = 2n.

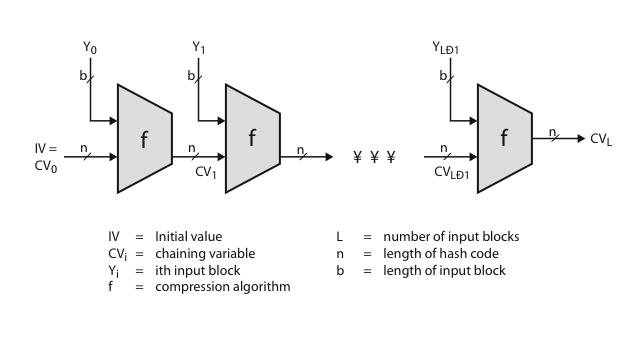
The first requirement speaks to the earlier example, in which an opponent is able to construct a new message to match a given MAC, even though the opponent does not know and does not learn the key. The second requirement deals with the need to thwart a brute-force attack based on chosen plaintext. That is, if we assume that the opponent does not know K but does have access to the MAC function and can present messages for MAC generation, then the opponent could try various messages until finding one that matches a given MAC. If the MAC function exhibits uniform distribution, then a brute-force method would require, on average, 2(n1) attempts before finding a message that fits a given MAC. The final requirement dictates that the authentication algorithm should not be weaker with respect to certain parts or bits of the message than others. If this were not the case, then an opponent who had M and C(K, M) could attempt variations on M at the known "weak spots" with a likelihood of early success at producing a new message that matched the old MAC.

1. **Hash Functions**

A hash value is generated by a function H of the form: h = H(M) where M is a variable-length message, and H(M) is the fixed length hash value (also referred to as a message digest or hash code). Figures 10.1 and 10.2 shows the basic uses of the hash function whereas figure 10.3 shows the general structure of a hash code. The hash value is appended to the message at the source at the time when the message is assumed or known to be correct. The receiver authenticates that message by recomputing the hash value. Because the hash function itself is not considered to be secret, some means is required to protect the hash value (see figures 10.1 and 10.2). We begin by examining the requirements for a hash function to be used for message authentication. Because hash functions are, typically, quite complex, it is useful to examine next some very simple hash functions to get a feel for the issues involved. We then look at several approaches to hash function design.

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

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



The first three properties are requirements for the practical application of a hash function to message authentication. The fourth property is the “one-way” property; it is easy to generate a code given a message but virtually impossible to generate a message given a code. This property is important if the authentication technique involves the use of a secret value (see figure 10.2e). The secret value itself is not sent; however, if the hash function is not one-way, an attacker can easily discover the secret value. If the attacker can observe or intercept a transmission, the attacker obtains the message M and the hash code C = H(SAB||M). The attacker then inverts the hash function to obtain SAB||M = H−1 (C). Because the attacker now has both M and SAB||M, it is a trivial matter to recover SAB. The fifth property guarantees that an alternative message hashing to the same value as a given message cannot be found. This prevents forgery when an encrypted hash code is used (see figures 10.1b and c). For these cases, the opponent can read the message and therefore generate its hash code. But, because the opponent does not have the secret key, the opponent should not be able to alter the message without detection. If this property were not true, an attacker would be capable of the following sequence:

1. Observe or intercept a message plus its encrypted hash code.
2. Generate an unencrypted hash code from the message.
3. Generate an alternate message with the same hash code.

A hash function that satisfies the first five properties in the preceding list is referred to as a weak hash function. If the sixth property is also satisfied, then it is referred to as a strong hash function. The sixth property protects against a sophisticated class of attack known as the birthday attack which we will be looking at later in the notes. Figure 10.3 shows the general structure of a secure hash code. In the next section we are going to study a specific algorithm (SHA-1) which will be seen to have this format. Notice this has a similar structure to the CBC mode used for symmetric algorithms.

1. **The Secure Hash Algorithm**

The Secure Hash Algorithm (SHA) was developed by the National Institute of Standards and Technology (NIST) and published as a federal information processing standard (FIPS 180) in 1993; a revised version was issued as FIPS 180-1 in 1995 and is generally referred to as SHA-1. The actual standards document is entitled Secure Hash Standard. SHA is based on the MD4 algorithm which is a message digest algorithm that was developed by Ron Rivest at MIT (the “R” in the RSA (Rivest-ShamirAdelman) public key encryption algorithm). MD4 was later replaced with the popular MD5 algorithm also by Ron Rivest however advances in cryptanalysis and computing power have led to their decline in popularity. Both MD4 and MD5 produce a 128 bit message digest whereas SHA-1 produces a 160 bit as will be seen. In 2002, NIST produced a revised version of the standard, FIPS180-2, that defined three new versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512 respectively. These new versions have the same underlying structure and use the same types of modular arithmetic and logical binary operations as SHA-1. In 2005, NIST announced the intention to phase out approval of SHA-1 and move to a reliance on the other SHA versions by 2010. Shortly thereafter, a research team described an attack in which two separate messages could be found that deliver the same SHA-1 hash using 2 69 operations, far fewer that the 2 80 operations previously thought needed

to find a collision with an SHA-1 hash. SHA-1 takes as input a message with a maximum length of less than 2 64 bits and produces as output a 160 bit message digest. The input is processed in 512-bit blocks. Figure 10.4 depicts the overall processing of a message to produce a digest. Although this diagram has MD5 as the hash function the structure is exactly the same for SHA-1 with the exception that the message length is limited in size (its isn’t for

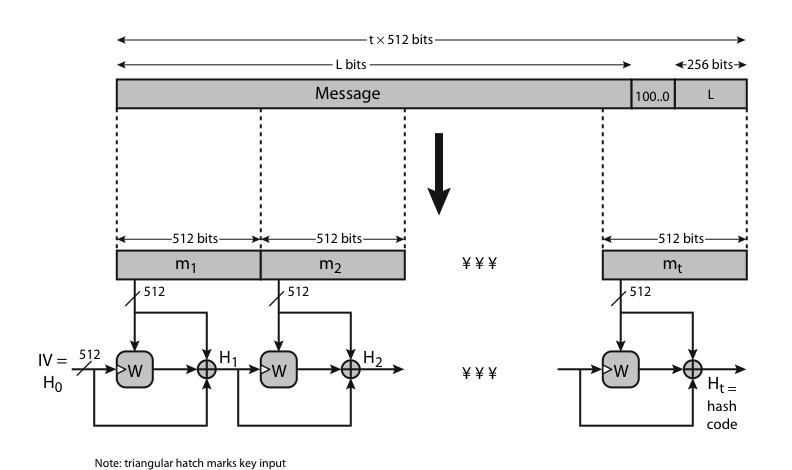
MD5) and the hash value (and intermediate values CVi) are 160 bits and not 128 as shown (which is the case for MD5). The processing consists of the following 5 steps:

1. Append padding bits: The message is padded so that its length is congruent to 448 modulo 512 (length ≡ 448

(mod 512)). That is, the length of the padded message is 64 bits less than an integer multiple of 512 bits. Padding is always added, even if the message is already of the desired length. Thus, the number of padding bits is in the range of 1 to 512 bits. The padding consists of a single 1-bit followed by the necessary number of 0-bits.

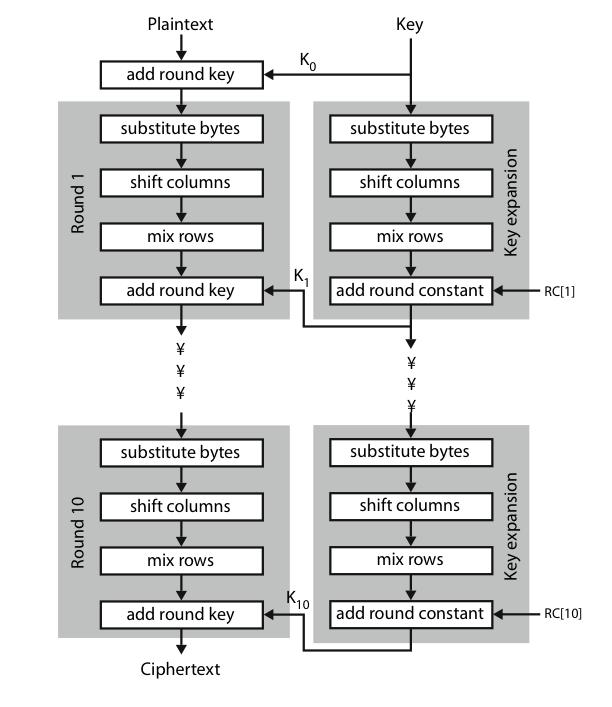
1. Append length: A block of 64 bits is appended to the message. This block is treated as an unsigned 64-bit integer (most significant byte first) and contains the length of the original message (before padding).
2. Initialize MD buffer: A 160 bit buffer is used to hold intermediate values and final results of the Hash function represented as 5, 32 bit registers (A, B, C, D, E) initialized as follows: A = 67452301 B = EF CDAB89 C = 98BADCF E D = 10325476 E = C3D2E1F0
3. Process message in 512 bit (16 word) blocks: The heart of the algorithm is a module which consists of four “rounds” of processing of 20 steps each (see fig- ure 10.5). Each round has similar structure but uses a different primitive logical function (f1, f2, f3 and f4). Each round takes as input the current 512-bit block being processed (Yq) and the 160-bit buffer value ABCDE and updates the contents of the buffer. Each round also makes use of an additive constant Kt where 0 ≤ t ≤ 79 indicates one of the 80 steps across four rounds. In fact, only four distinct constants are used (one for 0 ≤ t ≤ 19, 20 ≤ t ≤ 39, 40 ≤ t ≤ 59 and 60 ≤ t ≤ 79). The output of the fourth round is added (modulo 2 32) to the input to the first round (CVq) to produce CVq+1.
4. Output after all L 512 bit blocks have been processed the output from the Lth stage is the 160 bit digest.
   1. **Whirlpool**

* now examine the Whirlpool hash function
* endorsed by European NESSIE project
* uses modified AES internals as compression function
* addressing concerns on use of block ciphers seen previously
* with performance comparable to dedicated algorithms like SHA



**Whirlpool Block Cipher W**

* designed specifically for hash function use
* with security and efficiency of AES
* but with 512-bit block size and hence hash
* similar structure & functions as AES but
  + input is mapped row wise
  + has 10 rounds
  + a different primitive polynomial for GF(2^8)
  + uses different S-box design & values



**Whirlpool Performance & Security**

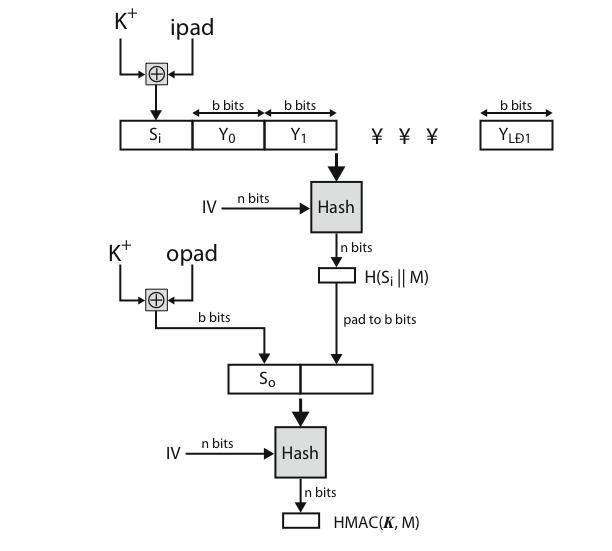
* + Whirlpool is a very new proposal
  + hence little experience with use
  + but many AES findings should apply
  + does seem to need more h/w than SHA, but with better resulting performance

1. **HMAC** 
   * specified as Internet standard RFC2104
   * uses hash function on the message:

HMACK = Hash[(K+ XOR opad) ||

Hash[(K+ XOR ipad)||M)]]

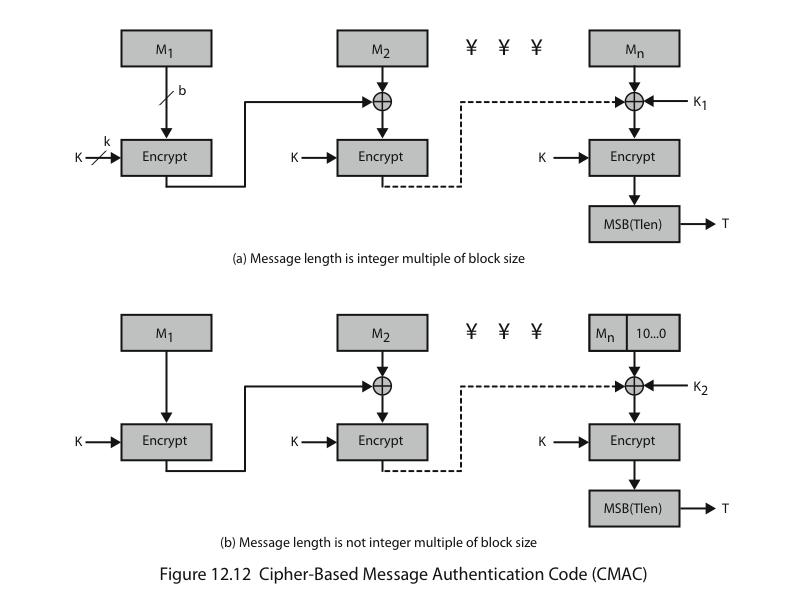
* where K+ is the key padded out to size
* and opad, ipad are specified padding constants
* overhead is just 3 more hash calculations than the message needs alone
* any hash function can be used
* eg. MD5, SHA-1, RIPEMD-160, Whirlpool



**HMAC Security**

* + - proved security of HMAC relates to that of the underlying hash algorithm
    - attacking HMAC requires either:
      * brute force attack on key used
      * birthday attack (but since keyed would need to observe a very large number of messages)
    - choose hash function used based on speed verses security constraints

1. **CMAC** 
   * previously saw the DAA (CBC-MAC)
   * widely used in govt & industry
   * but has message size limitation
   * can overcome using 2 keys & padding
   * thus forming the Cipher-based Message Authentication Code (CMAC)
   * adopted by NIST SP800-38B



1. **Digital Signatures**

Message authentication protects two parties who exchange messages from any third party. However, it does not protect the two parties against each other. Several forms of dispute between the two are possible. For example, suppose that John sends an authenticated message to Mary using one of the schemes described earlier. Consider the following disputes that could arise:

* Mary may forge a different message and claim that it came from John. Mary would simply have to create a message and append an authentication code using the key that John and Mary share.
* John can deny sending the message. Because it is possible for Mary to forge a message, there is no way to prove that John did in fact send the message. Both scenarios are of legitimate concern. In situations where there is not complete trust between sender and receiver, something more than authentication is needed. The most attractive solution to this problem is the digital signature. The digital signature is analogous to the handwritten signature. It must have the following properties:
* It must verify the author and the date and time of the signature.
* It must authenticate the contents at the time of the signature.
* It must be verifiable by third parties, to resolve disputes. Thus, the digital signature function includes the authentication function. On the basis of these properties, we can formulate the following requirements for a digital signature:
* The signature must be a bit pattern that depends on the message being signed.
* The signature must use some information unique to the sender, to prevent both forgery and denial.
* It must be relatively easy to produce the digital signature. • It must be relatively easy to recognise and verify the digital signature.
* It must be computationally infeasible to forge a digital signature, either by constructing a new message for an existing digital signature or by constructing a fraudulent digital signature for a given message.
* It must be practical to retain a copy of the digital signature in storage. One of the most popular algorithms for implementing digital signatures is discussed next.

**Digital Signature Standard (DSS)**

The National Institute of Standards and Technology (NIST) has published FIPS 186 known as the Digital Signature Standard (DSS). The DSS makes use of the Secure Hash Algorithm (SHA) that we just discussed and presents a new digital signature technique, the Digital Signature Algorithm (DSA). The DSS was originally proposed in 1991 and revised in 1993 in response to public feedback concerning the security of the scheme. There was a further minor revision in 1996. In 2000, an expanded version of the standard was issued as FIPS 186-2. This latest version also incorporates digital signature algorithms based on RSA and on elliptic curve cryptography. In this section, we discuss the original DSS algorithm. The DSS uses an algorithm that is designed to provide only the digital signature function. Unlike RSA, it cannot be used for encryption or key exchange. Nevertheless, it is a public-key technique. It is based on the difficulty of computing discrete logarithms (as is the Diffie Hellman key exchange).

The overall scheme is seen in figure 10.6. There are three parameters that are public and can be common to a group of users. A 160-bit prime number q is chosen. Next, a prime number p is selected with a length between

512 and 1024 bits such that q divides (p − 1). Finally, g is chosen to be of the form h (p−1)/q mod p, where h is an integer between 14 and (p − 1) with the restriction that g must be greater than 1. With these numbers in hand, each user selects a private key and generates a public key. The private key x must be a number from 1 to (p−1) and should be chosen randomly or pseudorandomly. The public key is calculated from the private key as y = g x mod p. The calculation of y is relatively straightforward. However finding x given the other parameters appears not to be. To create a signature, a user calculates two quantities, r and s, that are functions of the public key components (p, q, g), the user’s private key (x), the hash code of the message H(M), and an additional integer k that should be generated randomly or pseudorandomly and be unique for each signing. At the receiving end, verification is performed using the formulas shown in Figure 10.6. The receiver generates a quantity v that is a

function of the public key component, the sender’s public key, and the hash code of the incoming message. If this quantity matches the r component of the signature, then the signature is validated. Figure 10.7 depicts the functions of signing and verifying. The structure of the algorithm, as revealed in the figure, is quite interesting. Note that the test at the end is on the value r, which does not depend on the message at all. Instead, r is a function of k and the three global public-key components. The multiplicative inverse of k mod q is passed to a function that also has as inputs the message hash code and the user’s private key. The structure of this function is such that the receiver can recover r using the incoming message and signature, the public key of the user, and the global public key. It is certainly not obvious from figure 10.7 that such a scheme would work. However it has been proven that is does. Given the difficulty of taking discrete logarithms, it is infeasible for an opponent to recover K from r or to recover x from s.

1. **The Knapsack Problem**

The First General Public-Key Algorithm used what we call the Knapsack Algorithm. Although we now know that this algorithm is not secure we can use it to look at how these types of encryption mechanisms work.

The knapsack algorithm works like this:

Imagine you have a set of different weights which you can use to make any total weight that you need by adding combinations of any of these weights together

Let us look at an example:

Imagine you had a set of weights 1, 6, 8, 15 and 24. To pack a knapsack weighing 30, you could use weights 1, 6, 8 and 15. It would not be possible to pack a knapsack that weighs 17 but this might not matter. You might represent the weight 30 by the binary code 11110 (one 1, one 6, one 8, one 15 and no 24). Example:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Plain text |  | 10011 | |  | 11010 | |  | 01011 | |  | 00000 | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  | |  |  | |  |  | |  |
| Knapsack |  | 1 | 6 8 15 24 |  | 1 | 6 8 15 24 |  | 1 | 6 8 15 24 |  | 1 | 6 8 15 24 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  | |  |  | |  |  | |  |  | |  |
| Cipher text |  | 1 | + 15 + 24 = 40 |  | 1 | + 6 + 15 = 22 |  | 6 | + 15 + 24 = 45 |  | 0 | = 0 |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |

What total weights is it possible to make?

So, if someone sends you the code 38 this can only have come from the plain text 01101. When the Knapsack Algorithm is used in public key cryptography, the idea is to create two different knapsack problems. One is easy to solve, the other not. Using the easy knapsack, the hard knapsack is derived from it. The hard knapsack becomes the public key. The easy knapsack is the private key. The public key can be used to encrypt messages, but cannot be used to decrypt messages. The private key decrypts the messages.

6. **Authentication Applications**

* 1. **Kerberos**
* Kerberos is an authentication protocol and a software suite implementing this protocol.
* Kerberos uses symmetric cryptography to authenticate clients to services and vice versa.
* For example, Windows servers use Kerberos as the primary authentication mechanism, working in conjunction with Active Directory to maintain centralized user information.
* Other possible uses of Kerberos include allowing users to log into other machines in a local-area network, authentication for web services, authenticating email client and servers, and authenticating the use of devices such as printers.
* Services using Kerberos authentication are commonly referred to as “Kerberized”.

**Kerberos Tickets**

* Kerberos uses the concept of a ticket as a token that proves the identity of a user.
* Tickets are digital documents that store session keys. They are typically issued during a login session and then can be used instead of passwords for any Kerberized services. During the course of authentication, a client receives two tickets: – A ticket-granting ticket (TGT), which acts as a global identifier for a user and a session key – A service ticket, which authenticates a user to a particular service
* These tickets include time stamps that indicate an expiration time after which they become invalid. This expiration time can be set by Kerberos administrators depending on the service.

**Kerberos Servers**

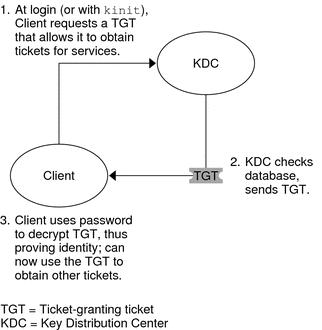
* To accomplish secure authentication, Kerberos uses a trusted third party known as a key distribution center

(KDC), which is composed of two components, typically integrated into a single server: – An authentication server (AS), which performs user authentication – A ticket-granting server (TGS), which grants tickets to users

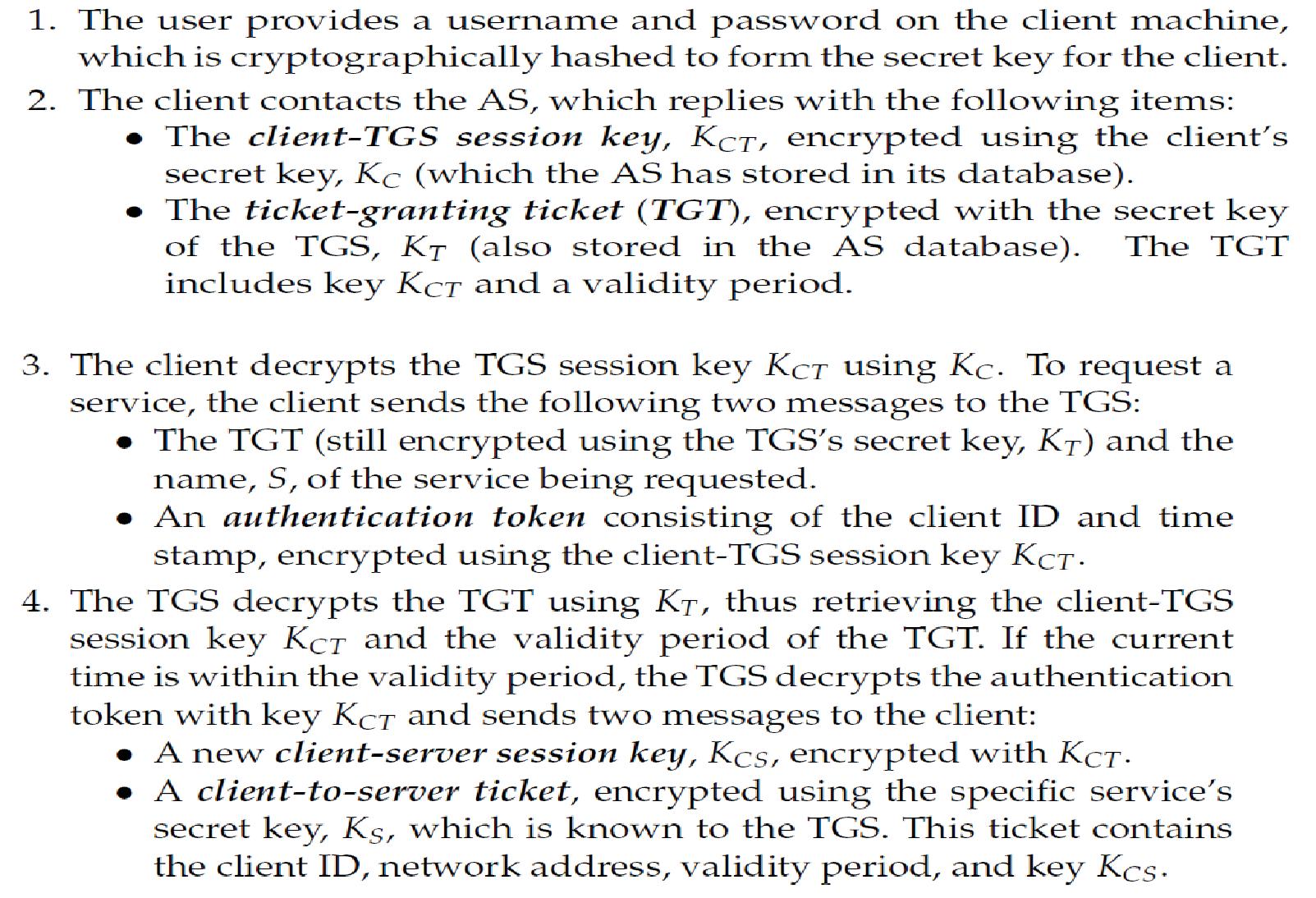
* The authentication server keeps a database storing the secret keys of the users and services. The secret key of a user is typically generated by performing a one-way hash of the user-provided password. Kerberos is designed to be modular, so that it can be used with a number of encryption protocols, with AES being the default cryptosystem.
* Kerberos aims to centralize authentication for an entire network—rather than storing sensitive authentication information at each user’s machine, this data is only maintained in one presumably secure location.

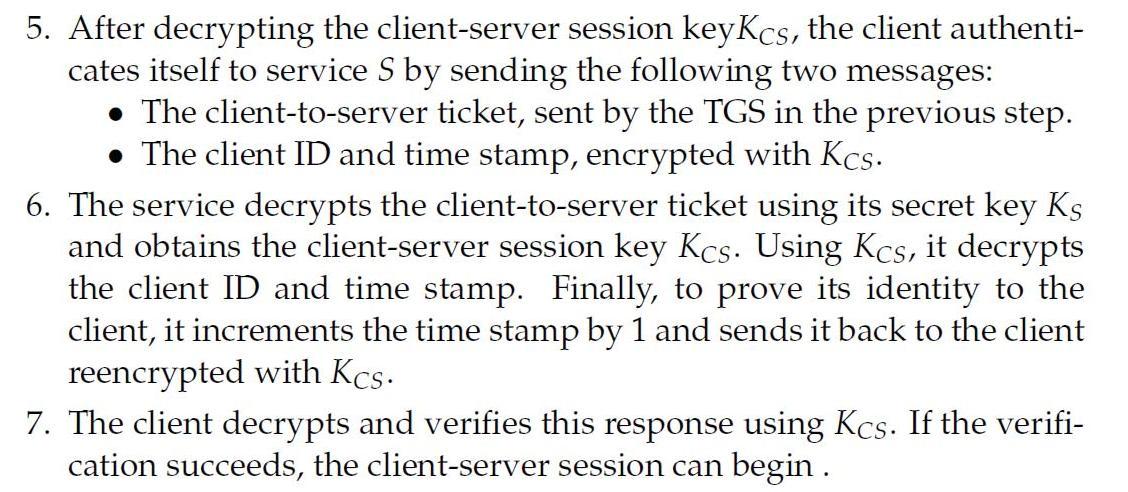
**Kerberos Authentication**

* The client and authentication server authenticate themselves to each other.
* The client and ticket-granting server authenticate themselves to each other.
* The client and requested service authenticate themselves to each other, at which point the service will be provided to the client.



**Authentication Details**





**Kerberos Advantages**

* The Kerberos protocol is designed to be secure even when performed over an insecure network.
* Since each transmission is encrypted using an appropriate secret key, an attacker cannot forge a valid ticket to gain unauthorized access to a service without compromising an encryption key or breaking the underlying encryption algorithm, which is assumed to be secure.
* Kerberos is also designed to protect against replay attacks, where an attacker eavesdrops legitimate Kerberos communications and retransmits messages from an authenticated party to perform unauthorized actions.

– The inclusion of time stamps in Kerberos messages restricts the window in which an attacker can retransmit messages.

– Tickets may contain the IP addresses associated with the authenticated party to prevent replaying messages from a different IP address.

– Kerberized services make use of a “replay cache,” which stores previous authentication tokens and detects their reuse.

* Kerberos makes use of symmetric encryption instead of public-key encryption, which makes Kerberos computationally efficient
* The availability of an open-source implementation has facilitated the adoption of Kerberos.

**Kerberos Disadvantages**

* Kerberos has a single point of failure: if the Key Distribution Center becomes unavailable, the authentication scheme for an entire network may cease to function.

– Larger networks sometimes prevent such a scenario by having multiple KDCs, or having backup KDCs available in case of emergency.

* If an attacker compromises the KDC, the authentication information of every client and server on the network would be revealed.
* Kerberos requires that all participating parties have synchronized clocks, since time stamps are used.

1. **X.509 Directory Authentication Service**

ITU-T (International Telecommunication Union – Telecommunication Standardization Sector) recommendation X.509 is a part of the X.500 series of recommendations that define a directory service. The directory is a server or distributed set of servers that maintains a database of information about users. The information includes a mapping from user name to network address, as well as other attributes and information about the users.

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. X.509 defines alternative authentication

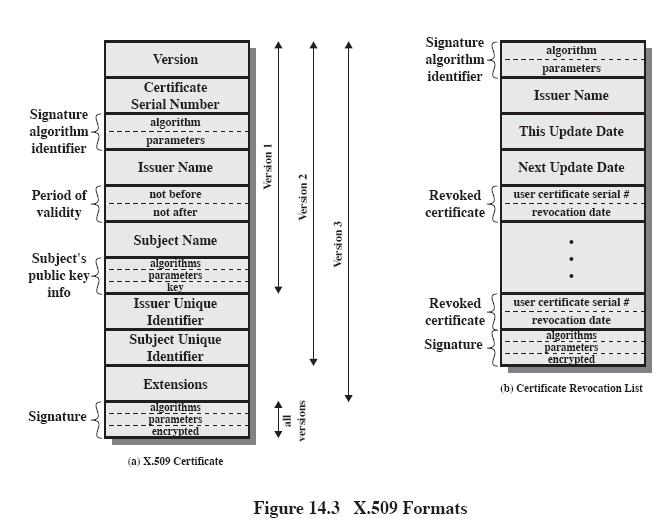
protocols based on the use of public-key certificates.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in variety of contexts (SSL, SET, etc.).

A third version of X.509 was issued in 1995 and revised in 2000.

The heart of X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certificate authority (CA) and placed in the directory by the CA or by the user. The directory itself is not responsible for the creation of public keys or for the certification function; it merely provides an easily accessible location for users to obtain certificates.

Figure 14.3a shows the general format of a certificate, which includes the following elements (www.dante.net/np/ds/osi/9594-8-X.509.A4.ps ):



1. Version: Differentiates among successive versions of the certificate format: the default version is 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.
2. Serial number: An integer value, unique within the issuing CA, that is unambiguously associated with this certificate
3. Signature algorithm identifier: The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility
4. Issuer name: X.500 name of the CA that created and signed this certificate (about X.500 names see, for example, http://java.sun.com/products/jndi/tutorial/ldap/models/x500.html )
5. Period of validity: Consists of two dates: the first and the last on which certificate is valid
6. Subject name: The name of the user to whom this certificate refers. That is, this certificate certifies the

public key of the subject who holds the corresponding private key

1. Subject’s public key information: The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters
2. Issuer unique identifier: An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities
3. Subject unique identifier: An optional bit string used to identify uniquely the subject in the event the X.500 name has been reused for different entities
4. Extensions: A set of one or more extension fields. Extensions were added in version 3 and are discussed later
5. Signature: Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA’s private key. This field includes the signature algorithm identifier

The unique identifier fields were added in version 2 to handle the possible reuse of subject and/or issuer names over time. These fields are rarely used.

The standard uses the following notation to define a certificate: CA<<A>> = CA{V,SN,AI,CA,TA,A,Ap}

Where

Y<<X>> = certificate of user X issued by certification authority Y

Y{I} = the signing of I by Y. It consists of I with an encrypted hash code appended

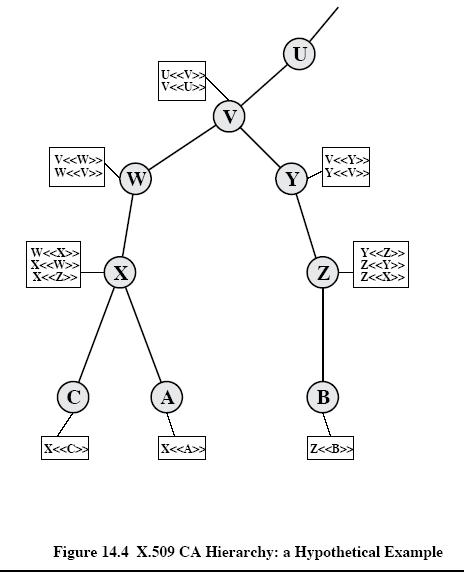
The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

**Obtaining a Certificate**

* any user with access to CA can get any certificate from it
* only the CA can modify a certificate
* because cannot be forged, certificates can be placed in a public directory

**CA Hierarchy**

* if both users share a common CA then they are assumed to know its public key
* otherwise CA's must form a hierarchy
* use certificates linking members of hierarchy to validate other CA's – each CA has certificates for clients (forward) and parent (backward)
* each client trusts parents certificates
* enable verification of any certificate from one CA by users of all other CAs in hierarchy



**Certificate Revocation**

* Certificates have a period of validity
* may need to revoke before expiry, eg: 1. user's private key is compromised 2. user is no longer certified by this CA 3. CA's certificate is compromised
* CA’s maintain list of revoked certificates – the Certificate Revocation List (CRL)
* users should check certs with CA’s CRL

**Authentication Procedures**

* X.509 includes three alternative authentication procedures:
  + One-Way Authentication
  + Two-Way Authentication
  + Three-Way Authentication
  + all use public-key signatures

**One-Way Authentication**

* 1 message ( A->B) used to establish – the identity of A and that message is from A – message was intended for B – integrity & originality of message
* Message must include timestamp, nonce, B's identity and is signed by A

**Two-Way Authentication**

* 2 messages (A->B, B->A) which also establishes in addition: – the identity of B and that reply is from B – that reply is intended for A – integrity & originality of reply
* Reply includes original nonce from A, also timestamp and nonce from B

**Three-Way Authentication**

* 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
* has reply from A back to B containing signed copy of nonce from B
* means that timestamps need not be checked or relied upon

**6.3. Public-Key Infrastructure**

RFC 2822 (Internet Security Glossary) defines public-key infrastructure (PKI) as the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography. The principal objective for developing a PKI is to enable secure, convenient, and efficient acquisition of public keys. The Internet Engineering Task Force (IETF) Public Key Infrastructure X.509 (PKIX) working group has been the driving force behind setting up a formal (and generic) model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet. This section describes the PKIX model.

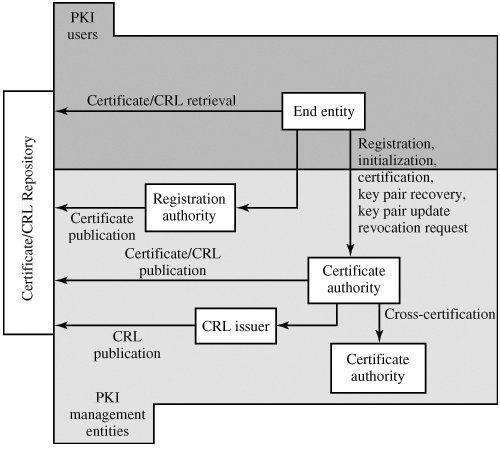


Fig: PKIX Architectural Model

Figure shows the interrelationship among the key elements of the PKIX model. These elements are

* End entity: A generic term used to denote end users, devices (e.g., servers, routers), or any other entity that can be identified in the subject field of a public key certificate. End entities typically consume and/or support PKI-related services. Certification authority (CA): The issuer of certificates and (usually) certificate revocation lists (CRLs). It may also support a variety of administrative functions, although these are often delegated to one or more Registration Authorities.
* Registration authority (RA): An optional component that can assume a number of administrative functions from the CA. The RA is often associated with the End Entity registration process, but can assist in a number of other areas as well.
* CRL issuer: An optional component that a CA can delegate to publish CRLs.
* Repository: A generic term used to denote any method for storing certificates and CRLs so that they can be retrieved by End Entities.

**PKIX Management Functions**

PKIX identifies a number of management functions that potentially need to be supported by management protocols. These are indicated in the above Figure and include the following:

* Registration: This is the process whereby a user first makes itself known to a CA (directly, or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI. Registration usually involves some offline or online procedure for mutual authentication. Typically, the end entity is issued one or more shared secret keys used for subsequent authentication. Initialization: Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.

For example, the client needs to be securely initialized with the public key and other assured information of the trusted CA(s), to be used in validating certificate paths.

* **Certification:** This is the process in which a CA issues a certificate for a user's public key, and returns thatcertificate to the user's client system and/or posts that certificate in a repository.
* **Key pair recovery**: Key pairs can be used to support digital signature creation and verification, encryption anddecryption, or both. When a key pair is used for encryption/decryption, it is important to provide a mechanism to recover the necessary decryption keys when normal access to the keying material is no longer possible, otherwise it will not be possible to recover the encrypted data. Loss of access to the decryption key can result from forgotten passwords/PINs, corrupted disk drives, damage to hardware tokens, and so on. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility (typically, the CA that issued the End Entity's certificate).
* **Key pair update**: All key pairs need to be updated regularly (i.e., replaced with a new key pair) and newcertificates issued. Update is required when the certificate lifetime expires and as a result of certificate revocation.
* **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificaterevocation. Reasons for revocation include private key compromise, change in affiliation, and name change.
* **Cross certification**: Two CAs exchange information used in establishing a cross-certificate. A cross-certificateis a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

**PKIX Management Protocols**

The PKIX working group has defines two alternative management protocols between PKIX entities that support the management functions listed in the preceding subsection. RFC 2510 defines the certificate management protocols (CMP). Within CMP, each of the management functions is explicitly identified by specific protocol exchanges. CMP is designed to be a flexible protocol able to accommodate a variety of technical, operational, and business models.

RFC 2797 defines certificate management messages over CMS (CMC), where CMS refers to RFC 2630, cryptographic message syntax. CMC is built on earlier work and is intended to leverage existing implementations. Although all of the PKIX functions are supported, the functions do not all map into specific protocol exchanges.

**6.4. Biometric Authentication**

Biometrics refers to metrics related to human characteristics. Biometrics authentication (or realistic authentication)is used in computer science as a form of identification and access control.It is also used to identify individuals in groups that are under surveillance.

Biometric identifiers are the distinctive, measurable characteristics used to label and describe individuals. Biometric identifiers are often categorized as physiological versus behavioral characteristics. Physiological characteristics are related to the shape of the body. Examples include, but are not limited to fingerprint, palm veins, face recognition, DNA, palm print, hand geometry, iris recognition, retina and odor/scent. Behavioral characteristics are related to the pattern of behavior of a person, including but not limited to typing rhythm, gait, and voice. Some researchers have coined the term behaviometrics to describe the latter class of biometrics.

More traditional means of access control include token-based identification systems, such as a driver's license or passport, and knowledge-based identification systems, such as a password or personal identification number. Since biometric identifiers are unique to individuals, they are more reliable in verifying identity than token and knowledge-based methods; however, the collection of biometric identifiers raises privacy concerns about the ultimate use of this information.

 **\* \* \* \* \* \* \* \* \* \* \* \* \* \* END OF UNIT III \* \* \* \* \* \* \* \* \* \* \* \* \* \***

1

**AUTHENTICATION SERVICES**

**KERBEROS**

Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Kerberos relies exclusively on conventional encryption, making no use of public-key encryption.

The following are the requirements for Kerberos:

1. **Secure:** A network eavesdropper should not be able to obtain the necessaryinformation to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
2. **Reliable:** For all services that rely on Kerberos for access control, lack ofavailability of the Kerberos service means lack of availability of the supported services. Hence, Kerberos should be highly reliable and should employ a distributed server architecture, with one system able to back up another.
3. **Transparent:** Ideally, the user should not be aware that authentication is takingplace, beyond the requirement to enter a password.
4. **Scalable:** The system should be capable of supporting large numbers of clientsand servers. This suggests a modular, distributed architecture.

To support these requirements, the overall scheme of Kerberos is that of a trusted third-party authentication service that uses a protocol based on that proposed by Needham and Schroeder [NEED78] It is trusted in the sense that clients and servers trust Kerberos to mediate their mutual authentication. Assuming the Kerberos protocol is well designed, then the authentication service is secure if the Kerberos server itself is secure.

**A simple authentication dialogue**

In an unprotected network environment, any client can apply to any server for service. The obvious security risk is that of impersonation. To counter this threat, servers must be able to confirm the identities of clients who request service. But in an open environment, this places a substantial burden on each server.

2

An alternative is to use an authentication server (AS) that knows the passwords of all users and stores these in a centralized database. In addition, the AS shares a unique secret key with each server. The simple authentication dialogue is as follows:

* C >> AS: IDc||Pc||IDv
* AS >> C: Ticket
* C >> V: IDc||Ticket Ticket= EKv(IDc||ADc||IDv)

|  |  |
| --- | --- |
| C: Client, |  |
| AS: Authentication Server, |  |
| V: Server, | IDc : ID of the client, |
| Pc:Password of the client, |  |
| ADc: Address of client, | IDv : ID of the server, |

Kv: secret key shared by AS and V, ||: concatenation.

**A more secure authentication dialogue**

There are two major problems associated with the previous approach:

* Plaintext transmission of the password.
* Each time a user has to enter the password.

To solve these problems, we introduce a scheme for avoiding plaintext passwords, and anew server, known as ticket granting server (TGS). The hypothetical scenario is as follows: **Once per user logon session:**

* C >> AS: IDc||IDtgs
* AS >> C: Ekc (Tickettgs)

**Once per type of service:**

* C >> TGS: IDc||IDv||Tickettgs
* TGS >> C: ticketv

**Once per service session:**

5. C >> V: IDc||ticketv

Tickettgs= Ektgs(IDc||ADc||IDtgs||TS1||Lifetime1) Ticketv= Ekv(IDc||ADc||IDv||TS2||Lifetime2)

3

C: Client, AS: Authentication Server, V: Server, IDc : ID of the client, Pc:Password of the client, ADc: Address of client, IDv : ID of the server, Kv: secret key shared by AS and V, ||: concatenation, IDtgs: ID of the TGS server, TS1, TS2: time stamps, lifetime: lifetime of the ticket.

The new service, TGS, issues tickets to users who have been authenticated to AS. Thus, the user first requests a ticket-granting ticket (Tickettgs) from the AS. The client module in the user workstation saves this ticket. Each time the user requires access to a new service, the client applies to the TGS, using the ticket to authenticate itself. The TGS then grants a ticket for the particular service. The client saves each service-granting ticket and uses it to authenticate its user to a server each time a particular service is requested. Let us look at the details of this scheme:

1. The client requests a ticket-granting ticket on behalf of the user by sending its user's ID and password to the AS, together with the TGS ID, indicating a request to use the TGS service.
2. The AS responds with a ticket that is encrypted with a key that is derived from the user's password.

When this response arrives at the client, the client prompts the user for his or her password, generates the key, and attempts to decrypt the incoming message.

If the correct password is supplied, the ticket is successfully recovered.

Because only the correct user should know the password, only the correct user can recover the ticket. Thus, we have used the password to obtain credentials from Kerberos without having to transmit the password in plaintext.

Now that the client has a ticket-granting ticket, access to any server can be obtained with steps 3 and 4:

1. The client requests a service-granting ticket on behalf of the user. For this purpose, the client transmits a message to the TGS containing the user's ID, the ID of the desired service, and the ticket-granting ticket.
2. The TGS decrypts the incoming ticket and verifies the success of the decryption by the presence of its ID. It checks to make sure that the lifetime has not expired. Then it

4

compares the user ID and network address with the incoming information to authenticate the user. If the user is permitted access to the server V, the TGS issues a ticket to grant access to the requested service.

The service-granting ticket has the same structure as the ticket-granting ticket. Indeed, because the TGS is a server, we would expect that the same elements are needed to authenticate a client to the TGS and to authenticate a client to an application server. Again, the ticket contains a timestamp and lifetime. If the user wants access to the same service at a later time, the client can simply use the previously acquired service-granting ticket and need not bother the user for a password. Note that the ticket is encrypted with a secret key (Kv) known only to the TGS and the server, preventing alteration.

Finally, with a particular service-granting ticket, the client can gain access to the corresponding service with step 5:

1. The client requests access to a service on behalf of the user. For this purpose, the client transmits a message to the server containing the user's ID and the service-granting ticket. The server authenticates by using the contents of the ticket.

This new scenario satisfies the two requirements of only one password query per user session and protection of the user password.

**Kerbero V4 Authentication Dialogue Message Exchange**

Two additional problems remain in the more secure authentication dialogue:

* Lifetime associated with the ticket granting ticket. If the lifetime is very short, then the user will be repeatedly asked for a password. If the lifetime is long, then the opponent has the greater opportunity for replay.
* Requirement for the servers to authenticate themselves to users.

The actual Kerberos protocol version 4 is as follows :

1. a basic third-party authentication scheme
2. have an Authentication Server (AS)

– users initially negotiate with AS to identify self

– AS provides a non-corruptible authentication credential (ticket granting ticket TGT)

1. have a Ticket Granting server (TGS)

– users subsequently request access to other services from TGS on basis of users TGT

5

**The table given below illustrates the mode of dialogue in V4**

|  |  |
| --- | --- |
| **Message (1)** | **Client requests ticket-granting ticket** |
|  |  |
| IDC | Tells AS identity of user from this client |
|  |  |
| IDtgs | Tells AS that user requests access to TGS |
|  |  |
| TS1 | Allows AS to verify that client's clock is synchronized with that of AS |
|  |  |

**Message (2)**

Kc

Kc,tgs

**AS returns ticket-granting ticket**

Encryption is based on user's password, enabling AS and client to verify password, and protecting contents of message (2)

Copy of session key accessible to client created by AS to permit secure exchange between client and TGS without requiring them to share a permanent key

|  |  |  |
| --- | --- | --- |
|  |  |  |
| IDtgs | Confirms that this ticket is for the TGS |  |
|  |  |  |

|  |  |
| --- | --- |
| TS2 | Informs client of time this ticket was issued |
|  |  |
| Lifetime2 | Informs client of the lifetime of this ticket |
|  |  |
| Tickettgs | Ticket to be used by client to access TGS |
|  |  |
|  | (a) Authentication Service Exchange |
|  |  |
| **Message (3)** | **Client requests service-granting ticket** |
|  |  |
| IDV | Tells TGS that user requests access to server V |
|  |  |
| Tickettgs | Assures TGS that this user has been authenticated by AS |
|  |  |
| Authenticatorc | Generated by client to validate ticket |
|  |  |
| **Message (4)** | **TGS returns service-granting ticket** |
|  |  |
| Kc,tgs | Key shared only by C and TGS protects contents of message (4) |
|  |  |
| Kc,v | Copy of session key accessible to client created by TGS to permit secure |
|  | exchange between client and server without requiring them to share a |
|  | permanent key |
|  |  |
| IDv | Confirms that this ticket is for server V |
|  |  |
| TS4 | Informs client of time this ticket was issued |
|  |  |
| Ticketv | Ticket to be used by client to access server V |
|  |  |
| Tickettgs | Reusable so that user does not have to reenter password |
|  |  |

Ktgs

Kc,tgs

Ticket is encrypted with key known only to AS and TGS, to prevent tampering

Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket

|  |  |
| --- | --- |
| IDC | Indicates the rightful owner of this ticket |
|  |  |
| ADC | Prevents use of ticket from workstation other than one that initially |
|  | requested the ticket |
|  |  |
| IDtgs | Assures server that it has decrypted ticket properly |
|  |  |
| TS2 | Informs TGS of time this ticket was issued |
|  |  |
| Lifetime2 | Prevents replay after ticket has expired |
|  |  |

Authenticatorc

Kc,tgs

Assures TGS that the ticket presenter is the same as the client for whom the ticket was issued has very short lifetime to prevent replay

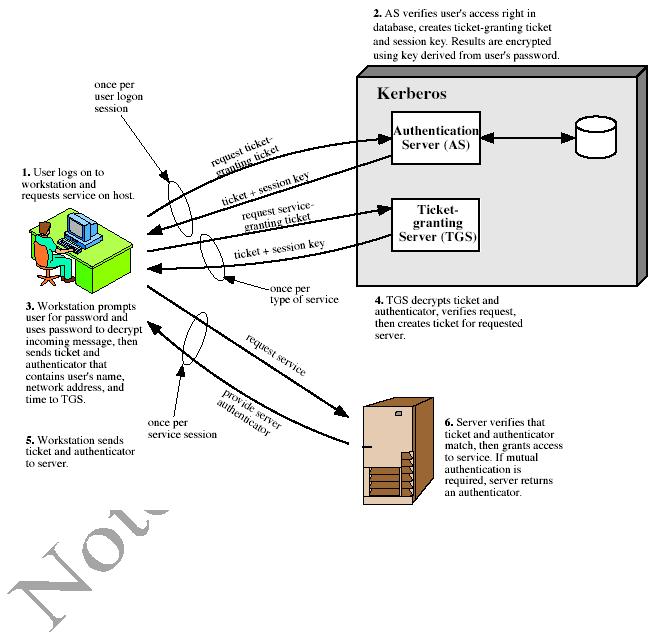
Authenticator is encrypted with key known only to client and TGS, to prevent tamperig

7



8

**Kerberos 4 Overview**



***Kerberos Realms and Multiple Kerberi***

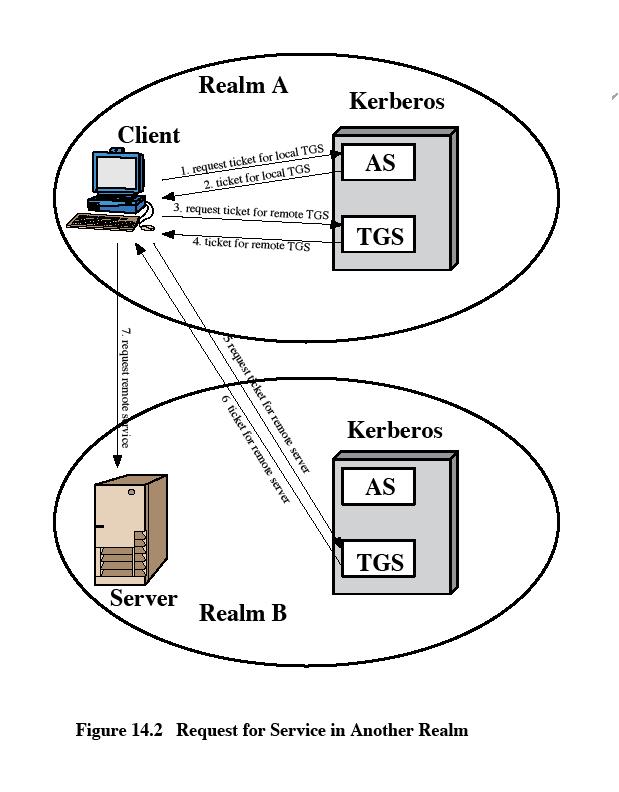
A full-service Kerberos environment consisting of a Kerberos server, a number of clients, and a number of application servers requires the following:

1. The Kerberos server must have the user ID and hashed passwords of all participating users in its database. All users are registered with the Kerberos server.
2. The Kerberos server must share a secret key with each server. All servers are registered with the Kerberos server.

Such an environment is referred to as a **Kerberos realm**.

9

**The concept of *realm* can be explained as follows.**



A Kerberos realm is a set of managed nodes that share the same Kerberos database. The Kerberos database resides on the Kerberos master computer system, which should be kept in a physically secure room.

10

A read-only copy of the Kerberos database might also reside on other Kerberos computer systems.

However, all changes to the database must be made on the master computer system. Changing or accessing the contents of a Kerberos database requires the Kerberos master password.

A related concept is that of a Kerberos principal, which is a service or user that is known to the Kerberos system.

Each Kerberos principal is identified by its principal name. Principal names consist of three parts: a service or user name, an instance name, and a realm name

Networks of clients and servers under different administrative organizations typically constitute different realms.

That is, it generally is not practical, or does not conform to administrative policy, to have users and servers in one administrative domain registered with a Kerberos server elsewhere.

However, users in one realm may need access to servers in other realms, and some servers may be willing to provide service to users from other realms, provided that those users are authenticated.

Kerberos provides a mechanism for supporting such interrealm authentication. For two realms to support interrealm authentication, a third requirement is added:

1. The Kerberos server in each interoperating realm shares a secret key with the server in the other realm. The two Kerberos servers are registered with each other.

The scheme requires that the Kerberos server in one realm trust the Kerberos server in the other realm to authenticate its users. Furthermore, the participating servers in the second realm must also be willing to trust the Kerberos server in the first realm.

**Kerberos version 5**

Version 5 of Kerberos provides a number of improvements over version 4.

* developed in mid 1990’s
* provides improvements over v4

– addresses environmental shortcomings

– and technical deficiencies

* specified as Internet standard RFC 1510

11

**Differences between version 4 and 5**

Version 5 is intended to address the limitations of version 4 in two areas:

* **Environmental shortcomings**

o encryption system dependence o internet protocol dependence o message byte ordering

o ticket lifetime

o authentication forwarding

* 1. inter-realm authenticaiton
* **Technical deficiencies** 
  1. double encryption

o PCBC encryption

o Session keys

o Password attacks

**The version 5 authentication dialogue**

12

First, consider the authentication service exchange. Message (1) is a client request for a ticket-granting ticket. As before, it includes the ID of the user and the TGS. The following new elements are added:

* Realm: Indicates realm of user
* Options: Used to request that certain flags be set in the returned ticket
* Times: Used by the client to request the following time settings in the ticket: from: the desired start time for the requested ticket

till: the requested expiration time for the requested ticket rtime: requested renew-till time

* [**Nonc**](mk:@MSITStore:D:cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app03.html#gloss01_051)**e**:A random value to be repeated in message (2) to assure that the responseis fresh and has not been replayed by an opponent

Message (2) returns a ticket-granting ticket, identifying information for the client, and a block encrypted using the encryption key based on the user's password.

13

This block includes the session key to be used between the client and the TGS, times specified in message (1), the nonce from message (1), and TGS identifying information.

The ticket itself includes the session key, identifying information for the client, the requested time values, and flags that reflect the status of this ticket and the requested options.

These flags introduce significant new functionality to version 5. For now, we defer a discussion of these flags and concentrate on the overall structure of the version 5 protocol.

Let us now compare the ticket-granting service exchange for versions 4 and 5. We see that message (3) for both versions includes an authenticator, a ticket, and the name of the requested service.

In addition, version 5 includes requested times and options for the ticket and a nonce, all with functions similar to those of message (1).

The authenticator itself is essentially the same as the one used in version 4.

Message (4) has the same structure as message (2), returning a ticket plus information needed by the client, the latter encrypted with the session key now shared by the client and the TGS.

Finally, for the client/server authentication exchange, several new features appear in version 5. In message (5), the client may request as an option that mutual authentication is required. The authenticator includes several new fields as follows:

* **Subkey**: The client's choice for an encryption key to be used to protect this

specific application session. If this field is omitted, the session key from the ticket (Kc,v) is used.

* **Sequence number**: An optional field that specifies the starting sequence numberto be used by the server for messages sent to the client during this session. Messages may be sequence numbered to detect replays.

If mutual authentication is required, the server responds with message (6). This message includes the timestamp from the authenticator. Note that in version 4, the timestamp was incremented by one. This is not necessary in version 5 because the nature of the format of messages is such that it is not possible for an opponent to create message (6) without knowledge of the appropriate encryption keys.

***Ticket Flags***

The flags field included in tickets in version 5 supports expanded functionality compared to that available in version 4.

14

**X.509 Certificates**

**Overview:**

* **issued by a Certification Authority (CA), containing:**

– version (1, 2, or 3)

– serial number (unique within CA) identifying certificate

– signature algorithm identifier

– issuer X.500 name (CA)

– period of validity (from - to dates)

– subject X.500 name (name of owner)

– subject public-key info (algorithm, parameters, key)

– issuer unique identifier (v2+)

– subject unique identifier (v2+)

– extension fields (v3)

– signature (of hash of all fields in certificate)

* **notation CA<<A>> denotes certificate for A signed by CA**

X.509 defines a framework for the provision of authentication services by the X.500 directory to its users. The directory may serve as a repository of public-key certificates. Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority. In addition, X.509 defines alternative authentication protocols based on the use of public-key certificates.

X.509 is an important standard because the certificate structure and authentication protocols defined in X.509 are used in a variety of contexts. For example, the X.509 certificate format is used in S/MIME), IP Security and SSL/TLS and SET

X.509 is based on the use of public-key cryptography and digital signatures. The standard does not dictate the use of a specific algorithm but recommends RSA. The digital signature scheme is assumed to require the use of a hash function.

15

**Certificates**

The heart of the X.509 scheme is the public-key certificate associated with each user. These user certificates are assumed to be created by some trusted certification authority (CA) and placed in the directory by the CA or by the user.

* **Version:**

Differentiates among successive versions of the certificate format; the default is version 1. If the Issuer Unique Identifier or Subject Unique Identifier are present, the value must be version 2. If one or more extensions are present, the version must be version 3.

* **Serial number**:

An integer value, unique within the issuing CA, that is unambiguously associated with this certificate.

* **Signature algorithm identifier**:

The algorithm used to sign the certificate, together with any associated parameters. Because this information is repeated in the Signature field at the end of the certificate, this field has little, if any, utility.

* **Issuer name**:

X.500 name of the CA that created and signed this certificate.

* **Period of validity**:

Consists of two dates: the first and last on which the certificate is valid.

* **Subject name**:

The name of the user to whom this certificate refers. That is, this certificate certifies the public key of the subject who holds the corresponding private key.

* **Subject's public-key information**:

The public key of the subject, plus an identifier of the algorithm for which this key is to be used, together with any associated parameters.

* **Issuer unique identifier**:

An optional bit string field used to identify uniquely the issuing CA in the event the X.500 name has been reused for different entities.

16

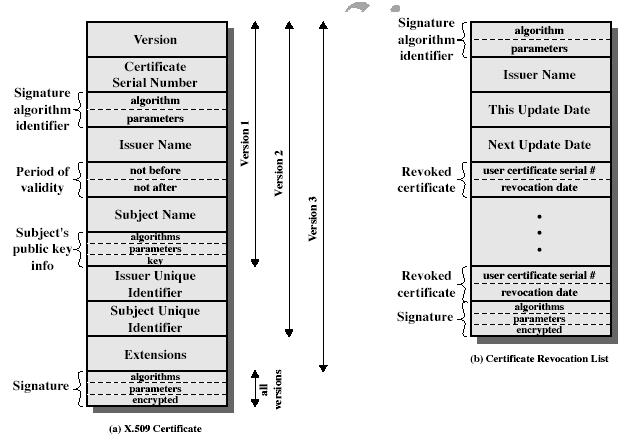
* **Subject unique identifier**:

An optional bit string field used to identify uniquely the subject in the event the X.500 name has been reused for different entities.

* **Extensions:**

A set of one or more extension fields. Extensions were added in version 3 and are discussed later in this section.

* **Signature:**

Covers all of the other fields of the certificate; it contains the hash code of the other fields, encrypted with the CA's private key. This field includes the signature algorithm identifie

17

The standard uses the following notation to define a certificate:

CA<<A>> = CA {V, SN, AI, CA, TA, A, Ap}

where

Y <<X>> = the certificate of user X issued by certification authority Y

Y {I} = the signing of I by Y. It consists of I with an encrypted

hash code appended

The CA signs the certificate with its private key. If the corresponding public key is known to a user, then that user can verify that a certificate signed by the CA is valid.

***Obtaining a User's Certificate***

User certificates generated by a CA have the following characteristics:

* Any user with access to the public key of the CA can verify the user public key that was certified.
* No party other than the certification authority can modify the certificate without this being detected.

Because certificates are unforgeable, they can be placed in a directory without the need for the directory to make special efforts to protect them.

If all users subscribe to the same CA, then there is a common trust of that CA. All user certificates can be placed in the directory for access by all users.

If there is a large community of users, it may not be practical for all users to subscribe to the same CA. Because it is the CA that signs certificates, each participating user must have a copy of the CA's own public key to verify signatures. This public key must be provided to each user in an absolutely secure (with respect to integrity and authenticity) way so that the user has confidence in the associated certificates. Thus, with many users, it may be more practical for there to be a number of CAs, each of which securely provides its public key to some fraction of the users.

Now suppose that A has obtained a certificate from certification authority X1 and B has obtained a certificate from CA X2. If A does not securely know the public key of X 2, then B's certificate, issued by X2, is useless to A.

A can read B's certificate, but A cannot verify the signature. However, if the two CAs have securely exchanged their own public keys, the following procedure will enable A to obtain B's public key:

18

1. A obtains, from the directory, the certificate of X2 signed by X1. Because A securely knows X1's public key, A can obtain X2's public key from its certificate and verify it by means of X1's signature on the certificate.
2. A then goes back to the directory and obtains the certificate of B signed by X2 Because A now has a trusted copy of X2's public key, A can verify the signature and securely obtain B's public key.

A has used a chain of certificates to obtain B's public key. In the notation of X.509, this chain is expressed as

X1<<X2>> X2 <<B>>

In the same fashion, B can obtain A's public key with the reverse chain:

X2<<X1>> X1 <<A>>

This scheme need not be limited to a chain of two certificates. An arbitrarily long path of CAs can be followed to produce a chain. A chain with N elements would be expressed as

X1<<X2>> X2 <<X3>>... XN<<B>>

In this case, each pair of CAs in the chain (Xi, Xi+1) must have created certificates for each other.

All these certificates of CAs by CAs need to appear in the directory, and the user needs to know how they are linked to follow a path to another user's public-key certificate. X.509 suggests that CAs be arranged in a hierarchy so that navigation is straightforward.

[Figure 14.5,](mk:@MSITStore:D:\\\\\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch14lev1sec2.html#ch14fig05) taken from X.509, is an example of such a hierarchy. The connected circles indicate the hierarchical relationship among the CAs; the associated boxes indicate certificates maintained in the directory for each CA entry. The directory entry for each CA includes two types of certificates:

* Forward certificates: Certificates of X generated by other CAs
* Reverse certificates: Certificates generated by X that are the certificates of other CAs

**CA Hierarchy Use**

In the example given below , user A can acquire the following certificates from the directory to establish a certification path to B:

X<<W>> W <<V>> V <<Y>> <<Z>> Z <<B>>

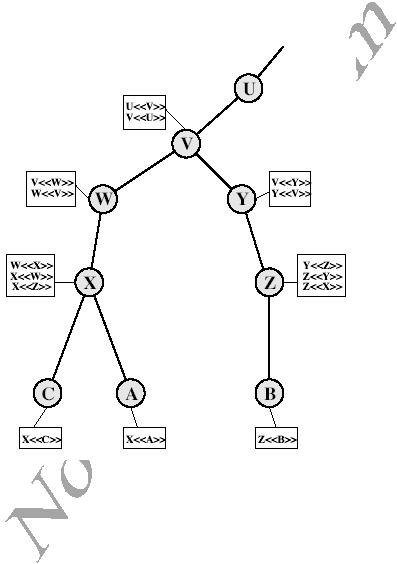
When A has obtained these certificates, it can unwrap the certification path in sequence to recover a trusted copy of B's public key. Using this public key, A can send encrypted

19

messages to B. If A wishes to receive encrypted messages back from B, or to sign messages sent to B, then B will require A's public key, which can be obtained from the following certification path:

Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>

B can obtain this set of certificates from the directory, or A can provide them as part of its initial message to B.



**Certificate Revocation**

* certificates have a period of validity
* may need to revoke before expiry, for the following reasons eg:
  1. user's private key is compromised
  2. user is no longer certified by this CA
  3. CA's certificate is compromised
* CA’s maintain list of revoked certificates
  1. the Certificate Revocation List (CRL)

20

* users should check certs with CA’s CRL

**Authentication Procedures**

X.509 includes three alternative authentication procedures:

* **One-Way Authentication**
* **Two-Way Authentication**
* **Three-Way Authentication**
* all use public-key signatures

**One-Way Authentication**

* 1 message ( A->B) used to establish

– the identity of A and that message is from A

– message was intended for B

– integrity & originality of message

* message must include timestamp, nonce, B's identity and is signed by A

**Two-Way Authentication**

* 2 messages (A->B, B->A) which also establishes in addition:

– the identity of B and that reply is from B

– that reply is intended for A

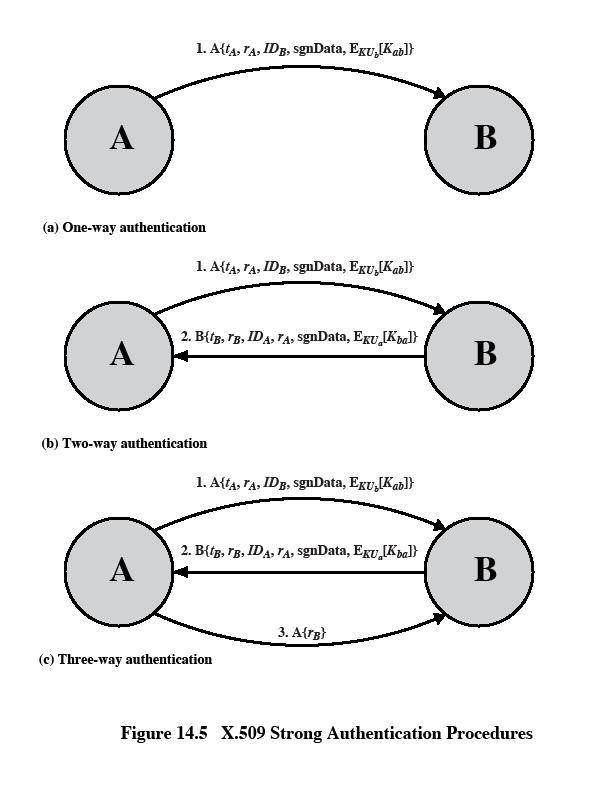
– integrity & originality of reply

* reply includes original nonce from A, also timestamp and nonce from B

**Three-Way Authentication**

* 3 messages (A->B, B->A, A->B) which enables above authentication without synchronized clocks
* has reply from A back to B containing signed copy of nonce from B
* means that timestamps need not be checked or relied upon

21



22

**X.509 Version 3**

The X.509 version 2 format does not convey all of the information that recent design and implementation experience has shown to be needed. [ [FORD9](mk:@MSITStore:D:cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_086)5] lists the following requirements not satisfied by version 2:

1. The Subject field is inadequate to convey the identity of a key owner to a public-key user.
2. The Subject field is also inadequate for many applications, which typically recognize entities by an Internet e-mail address, a URL, or some other Internet-related identification.
3. There is a need to indicate security policy information. There is a need to limit the damage that can result from a faulty or malicious CA by setting constraints on the applicability of a particular certificate.
4. It is important to be able to identify different keys used by the same owner at

different times.

The certificate extensions fall into three main categories: key and policy information, subject and issuer attributes, and certification path constraints.

***Key and Policy Information***

These extensions convey additional information about the subject and issuer keys, plus indicators of certificate policy.. For example, a policy might be applicable to the authentication of electronic data interchange (EDI) transactions for the trading of goods within a given price range.

This area includes the following:

* **Authority key identifier:** Identifies the public key to be used to verify thesignature on this certificate or CRL.
* **Subject key identifier:** Identifies the public key being certified. Useful forsubject key pair updating.
* **Key usage:** Indicates a restriction imposed as to the purposes for which, and thepolicies under which, the certified public key may be used.
* **Private-key usage period:** Indicates the period of use of the private keycorresponding to the public key.. For example, with digital signature keys, the usage period for the signing private key is typically shorter than that for the verifying public key.
* **Certificate policies:** Certificates may be used in environments where multiplepolicies apply.
* **Policy mappings:** Used only in certificates for CAs issued by other CAs.

23

***Certificate Subject and Issuer Attributes***

These extensions support alternative names, in alternative formats, for a certificate subject or certificate issuer and can convey additional information about the certificate subject, to increase a certificate user's confidence that the certificate subject is a particular person or entity. For example, information such as postal address, position within a corporation, or picture image may be required.

The extension fields in this area include the following:

* **Subject alternative name:** Contains one or more alternative names, using any ofa variety of forms
* **Subject directory attributes:** Conveys any desired X.500 directory attributevalues for the subject of this certificate.

***Certification Path Constraints***

These extensions allow constraint specifications to be included in certificates issued for CAs by other CAs.

The extension fields in this area include the following:

* **Basic constraints:** Indicates if the subject may act as a CA. If so, a certificationpath length constraint may be specified.
* **Name constraints**: Indicates a name space within which all subject names insubsequent certificates in a certification path must be located.
* **Policy constraints**: Specifies constraints that may require explicit certificatepolicy identification or inhibit policy mapping for the remainder of the certification path.

24

**ELECTRONIC MAIL SECURITY**

**PRETTY GOOD PRIVACY (PGP)**

**PGP provides the confidentiality and authentication service that can be used for electronic mail and file storage applications.** The steps involved in PGP are

* Select the best available cryptographic algorithms as building blocks.
* Integrate these algorithms into a general purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands.
* Make the package and its documentation, including the source code, freely available via the internet, bulletin boards and commercial networks.
* Enter into an agreement with a company to provide a fully compatible, low cost commercial version of PGP.

**PGP has grown explosively and is now widely used. A number of reasons can be**

**cited for this growth.**

* It is available free worldwide in versions that run on a variety of platform.
* It is based on algorithms that have survived extensive public review and are considered extremely secure.

e.g., RSA, DSS and Diffie Hellman for public key encryption CAST-128, IDEA and 3DES for conventional encryption SHA-1 for hash coding.

* It has a wide range of applicability.
* It was not developed by, nor it is controlled by, any governmental or standards organization.

**Operational description**

The actual operation of PGP consists of five services: authentication, confidentiality, compression, e-mail compatibility and segmentation.

1. **Authentication**

The sequence for authentication is as follows:

* + The sender creates the message

25

* 1. SHA-1 is used to generate a 160-bit hash code of the message
  2. The hash code is encrypted with RSA using the sender’s private key and the result is prepended to the message
  3. The receiver uses RSA with the sender’s public key to decrypt and recover the hash code.
  4. The receiver generates a new hash code for the message and compares it with the decrypted hash code. If the two match, the message is accepted as authentic.
* **Confidentiality**

Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files. In both cases, the conventional encryption algorithm CAST-128 may be used. The 64-bit cipher feedback (CFB) mode is used.

In PGP, each conventional key is used only once. That is, a new key is generated as a random 128-bit number for each message. Thus although this is referred to as **a session key**, it is in reality a **one time key**. To protect the key, it is encryptedwith the receiver’s public key.

The sequence for confidentiality is as follows:

* 1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.
  2. The message is encrypted using CAST-128 with the session key.
  3. The session key is encrypted with RSA, using the receiver’s public key and is prepended to the message.
  4. The receiver uses RSA with its private key to decrypt and recover the session key.
  5. The session key is used to decrypt the message.

**Confidentiality and authentication**

Here both services may be used for the same message. First, a signature is generated for the plaintext message and prepended to the message. Then the

26

plaintext plus the signature is encrypted using CAST-128 and the session key is encrypted using RSA.

**3. Compression**

As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space for both e-mail transmission and for file storage.

The signature is generated before compression for two reasons:

* It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification. If one signed a compressed document, then it would be

27

necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.

* Even if one were willing to generate dynamically a recompressed message fro verification, PGP’s compression algorithm presents a difficulty. The algorithm is not deterministic; various implementations of the algorithm

achieve different tradeoffs in running speed versus compression ratio and as a result, produce different compression forms.

Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult. The compression algorithm used is ZIP.

**4. e-mail compatibility**

Many electronic mail systems only permit the use of blocks consisting of ASCII texts. To accommodate this restriction, PGP provides the service of

converting the raw 8-bit binary stream to a stream of printable ASCII characters. The scheme used for this purpose is **radix-64 conversion**. Each group of three octets of binary data is mapped into four ASCII characters.

e.g., consider the 24-bit (3 octets) raw text sequence 00100011 01011100 10010001, we can express this input in block of 6-bits to produce 4 ASCII characters.

|  |  |  |  |
| --- | --- | --- | --- |
| 001000 | 110101 | 110010 | 010001 |
| I | L | Y | R => corresponding ASCII |
|  |  |  | characters |

**5. Segmentation and reassembly**

E-mail facilities often are restricted to a maximum length. E.g., many of the facilities accessible through the internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately.

To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via e-mail. The segmentation is done after all the other processing, including the radix-64

28

conversion. At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block before performing the other steps.

PGP Operation Summary:

**Cryptographic keys and key rings**

Three separate requirements can be identified with respect to these keys:

* A means of generating unpredictable session keys is needed.
* It must allow a user to have multiple public key/private key pairs.
* Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

We now examine each of the requirements in turn.

**1. Session key generation**

Each session key is associated with a single message and is used only for the purpose of encryption and decryption of that message. Random 128-bit numbers are generated using CAST-128 itself. The input to the random number

29

generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key. The plaintext input to CAST-128 is itself derived from a stream of 128-bit randomized numbers. These numbers are based on the keystroke input from the user.

**2. Key identifiers**

If multiple public/private key pair are used, then how does the recipient know which of the public keys was used to encrypt the session key? One simple solution would be to transmit the public key with the message but, it is unnecessary wasteful of space. Another solution would be to associate an identifier with each public key that is unique at least within each user.

The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID. The key ID associated with each public key consists of its least significant 64 bits. i.e., the key ID of public key KUa is (KUa mod 264).

**A message consists of three components**.

1. **Message component** –includes actual data to be transmitted, as well asthe filename and a timestamp that specifies the time of creation.
2. **Signature component** –includes the following

o Timestamp – time at which the signature was made. o Message digest – hash code.

1. Two octets of message digest – to enable the recipient to determine if the correct public key was used to decrypt the message.
   1. Key ID of sender’s public key – identifies the public key
2. **Session key component** –includes session key and the identifier of therecipient public key.

30

1. **Key rings**

PGP provides a pair of data structures at each node, one to store the public/private key pair owned by that node and one to store the public keys of the other users known at that node. These data structures are referred to as private key ring and public key ring.

**The general structures of the private and public key rings are shown below:**

**Timestamp** –the date/time when this entry was made.

**Key ID** –the least significant bits of the public key.

**Public key –** public key portion of the pair.

**Private key** –private key portion of the pair.

**User ID** –the owner of the key.

31

**Key legitimacy field** –indicates the extent to which PGP will trust that this is a validpublic key for this user.

**Signature trust field –** indicates the degree to which this PGP user trusts the signer to

certify public key.

**Owner trust field** –indicates the degree to which this public key is trusted to sign other

public key certificates.

**PGP message generation**

First consider message transmission and assume that the message is to be both signed and encrypted. The sending PGP entity performs the following steps:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | 32 | | |  |
|  |  |  |  |  |  |  |  |  |  | Public-key ring | | | | | | |  |  |  |  |
|  | Passphrase | | | | | H |  | IDB | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Private-key ring | | | | | | |  |  |  |  |  |  |  |  | Key ID | | |  |
|  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| IDA |  |  |  |  |  | E(KRa) | DC | |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Key ID |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  | Public key | | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | Private |  |  | RNG |  |  |  |  | KUb | | |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | key KRa |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | Session | | | | | | |  |  |  |  |
|  |  | H | | | | EP | || | |  | key Ks | | | | | | |  |  | Output |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |
| Message |  |  |  |  |  |  |  |  |  |  |  |  | EP | | | | || | | |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| M |  |  |  |  |  |  |  |  | EC |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | Signature+ | |  | Encrypted signature | | | | | | | | |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | message | |  |  | + message | | | | | |  |  |  |  |

**Figure: PGP message generation**

1. **signing the message** 
   * PGP retrieves the sender’s private key from the private key ring using user ID as an index. If user ID was not provided, the first private key from the ring is retrieved.
   * PGP prompts the user for the passpharse (password) to recover the unencrypted private key.
   * The signature component of the message is constructed.
2. **encrypting the message** 
   * PGP generates a session key and encrypts the message.
   * PGP retrieves the recipient’s public key from the public key ring using user

ID as index.

* + The session key component of the message is constructed.

The receiving PGP entity performs the following steps:

passphrase

33

H

Private-key ring Public-key ring

Select

E( KRb)

DC

Private key

KRb

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | Public key | | | |  |
| Receiver’s | |  |  |  |  |
|  |  |  |  | KUa | | | |  |
|  |  | Sender’s |  |  |
| key ID | |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  | DP | key ID |  |  |  |  |  |  |
| Encrypted | |  | DP | |  |  |  |
|  |  |  |  |  |  |  |
|  |  | Encrypted |  |  |  |  |
| session key | |  | Ks |  |  |  |  |
|  |  |  |  |
|  |  |  | digest |  |  |  |  |  |  |
| Encrypted | |  | DC |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
| message + | |  |  |  |  | Compare | | |  |
|  |  |  |  |  |  |
| signature | |  |  | Message |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  | H | | | |  |
|  |  |  |  |  |  |  |  |  |  |  |

**Figure: PGP message reception**

1. **decrypting the message** 
   * PGP retrieves the receiver’s private key from the private key ring, using the key ID field in the session key component of the message as an index.
   * PGP prompts the user for the passpharse (password) to recover the unencrypted private key.
   * PGP then recovers the session key and decrypts the message.
2. **Authenticating the message** 
   * PGP retrieves the sender’s public key from the public key ring, using the key

ID field in the signature key component of the message as an index.

* + PGP recovers the transmitted message digest.
  + PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

34

**Public-Key Management**

This whole business of protecting public keys from tampering is the single most difficult problem in practical public key applications. PGP provides a structure for solving this problem, with several suggested options that may be used.

***Approaches to Public-Key Management***

The essence of the problem is this: User A must build up a public-key ring containing the public keys of other users to interoperate with them using PGP. Suppose that A's key ring contains a public key attributed to B but that the key is, in fact, owned by C. This could happen if, for example, A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C. The result is that two threats now exist. First, C can send messages to A and forge B's signature, so that A will accept the message as coming from B. Second, any encrypted message from A to B can be read by C.

A number of approaches are possible for minimizing the risk that a user's public-key ring contains false public keys. Suppose that A wishes to obtain a reliable public key for B. The following are some approaches that could be used:

1. Physically get the key from B. B could store her public key (PUb) on a floppy disk and hand it to A..
2. Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate the key, in radix-64 format, over the phone.
3. Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer, D, creates a signed certificate. The certificate includes B's public key, the time of creation of the key, and a validity period for the key.
4. Obtain B's public key from a trusted certifying authority. Again, a public key certificate is created and signed by the authority. A could then access the authority, providing a user name and receiving a signed certificate.

For cases 3 and 4, A would already have to have a copy of the introducer's public key and trust that this key is valid. Ultimately, it is up to A to assign a level of trust to anyone who is to act as an introducer.

***The Use of Trust***

Although PGP does not include any specification for establishing certifying authorities or for establishing trust, it does provide a convenient means of using trust, associating trust with public keys, and exploiting trust information.

35

The basic structure is as follows. Each entry in the public-key ring is a public-key certificate.

Associated with each such entry is a key legitimacy field that indicates the extent to which PGP will trust that this is a valid public key for this user; the higher the level of trust, the stronger is the binding of this user ID to this key. This field is computed by PGP.

Also associated with the entry are zero or more signatures that the key ring owner has collected that sign this certificate. In turn, each signature has associated with it a signature trust field that indicates the degree to which this PGP user trusts the signer to certify public keys.

The key legitimacy field is derived from the collection of signature trust fields in the entry.

Finally, each entry defines a public key associated with a particular owner, and an owner trust field is included that indicates the degree to which this public key is trusted to sign other public-key certificates; this level of trust is assigned by the user.

.

The three fields mentioned in the previous paragraph are each contained in a structure referred to as a trust flag byte.

Suppose that we are dealing with the public-key ring of user A. We can describe the operation of the trust processing as follows:

1. When A inserts a new public key on the public-key ring, PGP must assign a value to the trust flag that is associated with the owner of this public key. If the owner is A, and therefore this public key also appears in the private-key ring, then a value of ultimate trust is automatically assigned to the trust field. Otherwise, PGP asks A for his assessment of the trust to be assigned to the owner of this key, and A must enter the desired level. The user can specify that this owner is unknown, untrusted, marginally trusted, or completely trusted.
2. When the new public key is entered, one or more signatures may be attached to it. More signatures may be added later. When a signature is inserted into the entry, PGP searches the public-key ring to see if the author of this signature is among the known public-key owners. If so, the OWNERTRUST value for this owner is assigned to the SIGTRUST field for this signature. If not, an unknown user value is assigned.

36

* The value of the key legitimacy field is calculated on the basis of the signature trust fields present in this entry. If at least one signature has a signature trust value of ultimate, then the key legitimacy value is set to complete.

The node labeled "You" refers to the entry in the public-key ring corresponding to this user. This key is legitimate and the OWNERTRUST value is ultimate trust. Each other node in the key ring has an OWNERTRUST value of undefined unless some other value is assigned by the user. In this example, this user has specified that it always trusts the following users to sign other keys: D, E, F, L. This user partially trusts users A and B to sign other keys.

So the shading, or lack thereof, of the nodes in  [Figure 15.7](mk:@MSITStore:C:\\\\Documents%20and%20Settings\\\\sethukarasi\\\\Desktop\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec1.html#ch15fig07) indicates the level of trust assigned by this user. The tree structure indicates which keys have been signed by which

37

other users. If a key is signed by a user whose key is also in this key ring, the arrow joins the signed key to the signatory. If the key is signed by a user whose key is not present in this key ring, the arrow joins the signed key to a question mark, indicating that the signatory is unknown to this user.

Several points are illustrated in this  [Figure 15.7](mk:@MSITStore:C:\\\\\\\\Documents%20and%20Settings\\\\\\\\sethukarasi\\\\\\\\Desktop\\\\\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec1.html#ch15fig07):

Note that all keys whose owners are fully or partially trusted by this user have been signed by this user, with the exception of node L.

1. We assume that two partially trusted signatures are sufficient to certify a key. Hence, the key for user H is deemed legitimate by PGP because it is signed by A and B, both of whom are partially trusted.
2. A key may be determined to be legitimate because it is signed by one fully trusted or two partially trusted signatories, but its user may not be trusted to sign other keys. For example, N's key is legitimate because it is signed by E, whom this user trusts, but N is not trusted to sign other keys because this user has not assigned N that trust value. Therefore, although R's key is signed by N, PGP does not consider R's key legitimate. This situation makes perfect sense. If you wish to send a private message to some individual, it is not necessary that you trust that individual in any respect. It is only necessary that you are sure that you have the correct public key for that individual.
3. [Figure 15.7](mk:@MSITStore:C:\\\\Documents%20and%20Settings\\\\sethukarasi\\\\Desktop\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec1.html#ch15fig07) also shows an example of a detached "orphan" node S, with two unknown signatures. Such a key may have been acquired from a key server. PGP cannot assume that this key is legitimate simply because it came from a reputable server. The user must declare the key legitimate by signing it or by telling PGP that it is willing to trust fully one of the key's signatories.

38

**S/MIME**

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851.

**Multipurpose Internet Mail Extensions**

MIME is an extension to the RFC 822 framework that is intended to address some of the problems and limitations of the use of SMTP (Simple Mail Transfer Protocol) or some other mail transfer protocol and RFC 822 for electronic mail. Following are the limitations of SMTP/822 scheme:

1. SMTP cannot transmit executable files or other binary objects.
2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII and the character code EBCDIC do not use a consistent set of mappings, resulting in translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standards defined in RFC 821. Common problems include:

o Deletion, addition, or reordering of carriage return and linefeed o Truncating or wrapping lines longer than 76 characters

o Removal of trailing white space (tab and space characters) o Padding of lines in a message to the same length

o Conversion of tab characters into multiple space characters

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations. The specification is provided in RFCs 2045 through 2049.

39

***Overview***

The MIME specification includes the following elements:

1. **Five new message header** fields are defined, which may be included in an RFC822 header. These fields provide information about the body of the message.
2. **A number of content formats** are defined, thus standardizing representationsthat support multimedia electronic mail.
3. **Transfer encodings** are defined that enable the conversion of any content formatinto a form that is protected from alteration by the mail system.

In this subsection, we introduce the five message header fields. The next two subsections deal with content formats and transfer encodings.

**The five header fields defined in MIME are as follows:**

* **MIME-Version**: Must have the parameter value 1.0. This field indicates that themessage conforms to RFCs 2045 and 2046.
* **Content-Type**: Describes the data contained in the body with sufficient detail
* **Content-Transfer-Encoding**: Indicates the type of transformation that has beenused to represent the body of the message in a way that is acceptable for mail transport.
* **Content-ID:** Used to identify MIME entities uniquely in multiple contexts.
* **Content-Description**: A text description of the object with the body; this is usefulwhen the object is not readable (e.g., audio data).

***MIME Content Types***

The bulk of the MIME specification is concerned with the definition of a variety of content types. This reflects the need to provide standardized ways of dealing with a wide variety of information representations in a multimedia environment.

[Table 15.3](mk:@MSITStore:C:\\\\Documents%20and%20Settings\\\\sethukarasi\\\\Desktop\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15table03) lists the content types specified in RFC 2046. There are seven different major types of content and a total of 15 subtypes

40

***Table 15.3. MIME Content Types***

***(This item is displayed on page 461 in the print version)***

For the text type of body, no special software is required to get the full meaning of the text, aside from support of the indicated character set. The primary subtype is plain text,

41

which is simply a string of ASCII characters or ISO 8859 characters. The enriched subtype allows greater formatting flexibility.

The multipart type indicates that the body contains multiple, independent parts. The Content-Type header field includes a parameter, called boundary, that defines the delimiter between body parts.

The multipart/digest subtype is used when each of the body parts is interpreted as an RFC 822 message with headers. This subtype enables the construction of a message whose parts are individual messages. For example, the moderator of a group might collect e-mail messages from participants, bundle these messages, and send them out in one encapsulating MIME message.

The message type provides a number of important capabilities in MIME. The message/rfc822 subtype indicates that the body is an entire message, including header and body. Despite the name of this subtype, the encapsulated message may be not only a simple RFC 822 message, but also any MIME message.

The message/partial subtype enables fragmentation of a large message into a number of parts, which must be reassembled at the destination. For this subtype, three parameters are specified in the Content-Type: Message/Partial field: an id common to all fragments of the same message, a sequence number unique to each fragment, and the total number of fragments.

The message/external-body subtype indicates that the actual data to be conveyed in this message are not contained in the body. Instead, the body contains the information needed to access the data. As with the other message types, the message/external-body subtype has an outer header and an encapsulated message with its own header. The only necessary field in the outer header is the Content-Type field, which identifies this as a message/external-body subtype. The inner header is the message header for the encapsulated message. The Content-Type field in the outer header must include an access-type parameter, which indicates the method of access, such as FTP (file transfer protocol).

The application type refers to other kinds of data, typically either uninterpreted binary data or information to be processed by a mail-based application.

***MIME Transfer Encodings***

The other major component of the MIME specification, in addition to content type specification, is a definition of transfer encodings for message bodies. The objective is to provide reliable delivery across the largest range of environments.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values, as listed in  [Table 15.4.](mk:@MSITStore:C:\\\\\\\\Documents%20and%20Settings\\\\\\\\sethukarasi\\\\\\\\Desktop\\\\\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15table04) For SMTP transfer, it is safe to use the 7bit form. The 8bit and binary forms may be usable in other mail

42

transport contexts. Another Content-Transfer-Encoding value is x-token, which indicates that some other encoding scheme is used, for which a name is to be supplied. The two actual encoding schemes defined are quoted-printable and base64.

7bit

8bit

binary

quoted-printable

base64

x-token

***Table 15.4. MIME Transfer Encodings***

The data are all represented by short lines of ASCII characters.

The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).

Not only may non-ASCII characters be present but the lines are not necessarily short enough for SMTP transport.

Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.

Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.

A named nonstandard encoding.

The quoted-printable transfer encoding is useful when the data consists largely of octets that correspond to printable ASCII characters. In essence, it represents nonsafe characters by the hexadecimal representation of their code and introduces reversible (soft) line breaks to limit message lines to 76 characters.

The base64 transfer encoding, also known as radix-64 encoding, is a common one for encoding arbitrary binary data in such a way as to be invulnerable to the processing by mail transport programs.

***Canonical Form***

An important concept in MIME and S/MIME is that of canonical form. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form, which is a format that may be peculiar to a particular system.

**S/MIME Functionality**

In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages. In this subsection, we briefly summarize S/MIME capability. We then look in more detail at this capability by examining message formats and message preparation.

43

***Functions***

S/MIME provides the following functions:

* **Enveloped data:** This consists of encrypted content of any type and encrypted-content encryption keys for one or more recipients.
* **Signed data**: A digital signature is formed by taking the message digest of thecontent to be signed and then encrypting that with the private key of the signer. The content plus signature are then encoded using base64 encoding. A signed data message can only be viewed by a recipient with S/MIME capability.
* **Clear-signed data:** As with signed data, a digital signature of the content isformed. However, in this case, only the digital signature is encoded using base64. As a result, recipients without S/MIME capability can view the message content, although they cannot verify the signature.
* **Signed and enveloped data:** Signed-only and encrypted-only entities may benested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

***Cryptographic Algorithms***

1. hash functions: SHA-1 & MD5
2. digital signatures: DSS & RSA
3. session key encryption: ElGamal & RSA
4. message encryption: Triple-DES, RC2/40 and others
5. have a procedure to decide which algorithms to use.

[Table 15.6](mk:@MSITStore:C:\\\\Documents%20and%20Settings\\\\sethukarasi\\\\Desktop\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15table06) summarizes the cryptographic algorithms used in S/MIME. S/MIME uses the following terminology, taken from RFC 2119 to specify the requirement level:

* Must: The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.
* Should: There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

44

***Table 15.6. Cryptographic Algorithms Used in S/MIME***

**Function**

Create a message digest to be used in forming a digital signature.

Encrypt message digest to form digital signature.

Encrypt session key for transmission with message.

Encrypt message for transmission with one-time session key.

Create a message authentication code

**Requirement**

MUST support SHA-1.

Receiver SHOULD support MD5 for backward compatibility.

Sending and receiving agents MUST support DSS.

Sending agents SHOULD support RSA encryption.

Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.

Sending and receiving agents SHOULD support Diffie-Hellman.

Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.

Sending and receiving agents MUST support encryption with triple DES

Sending agents SHOULD support encryption with AES.

Sending agents SHOULD support encryption with RC2/40.

Receiving agents MUST support HMAC with SHA-1.

Receiving agents SHOULD support HMAC with SHA-1.

**S/MIME Messages**

S/MIME makes use of a number of new MIME content types, which are shown in  [Tabl](mk:@MSITStore:C:Documents%20and%20SettingssethukarasiDesktopcryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15table07)e  [15.7.](mk:@MSITStore:C:\\\\Documents%20and%20Settings\\\\sethukarasi\\\\Desktop\\\\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15table07) All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories and made available for the S/MIME effort.

45

***Table 15.7. S/MIME Content Types***

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Subtype** | **smime Parameter** | **Description** |
|  |  |  |  |
| Multipart | Signed |  | A clear-signed message in two parts: one |
|  |  |  | is the message and the other is the |
|  |  |  | signature. |
|  |  |  |  |
| Application | pkcs 7- | signedData | A signed S/MIME entity. |
|  | mime |  |  |
|  |  |  |  |
|  | pkcs 7- | envelopedData | An encrypted S/MIME entity. |
|  | mime |  |  |
|  |  |  |  |
|  | pkcs 7- | degenerate | An entity containing only public- key |
|  | mime | signedData | certificates. |
|  |  |  |  |
|  | pkcs 7- | CompressedData | A compressed S/MIME entity |
|  | mime |  |  |
|  |  |  |  |
|  | pkcs 7- | signedData | The content type of the signature subpart |
|  | signature |  | of a multipart/signed message. |
|  |  |  |  |

We examine each of these in turn after first looking at the general procedures for S/MIME message preparation.

***SECURING A MIME ENTITY***

S/MIME secures a MIME entity with a signature, encryption, or both. A MIME entity may be an entire message (except for the RFC 822 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message. In all cases, the message to be sent is converted to canonical form. In particular, for a given type and subtype, the appropriate canonical form is used for the message content. For a multipart message, the appropriate canonical form is used for each subpart.

The use of transfer encoding requires special attention.

***i)EnvelopedData***

An application/pkcs7-mime subtype is used for one of four categories of S/MIME processing, each with a unique smime-type parameter. In all cases, the resulting entity, referred to as an object, is represented in a form known as Basic Encoding Rules (BER), which is defined in ITU-T Recommendation X.209. The steps for preparing an envelopedData MIME entity are as follows:

46

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or tripleDES).
2. For each recipient, encrypt the session key with the recipient's public RSA key.
3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate, [[3]](mk:@MSITStore:C:Documents%20and%20SettingssethukarasiDesktopcryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch15lev1sec2.html#ch15fn3) an identifier of the algorithm used to encrypt the session key, and the encrypted session key.

This is an X.509 certificate, discussed later in this section.

1. Encrypt the message content with the session key.

The RecipientInfo blocks followed by the encrypted content constitute the envelopedData. This information is then encoded into base64. To recover the encrypted message, the recipient first strips off the base64 encoding. Then the recipient's private key is used to recover the session key. Finally, the message content is decrypted with the session key.

***ii)SignedData***

The signedData smime-type can actually be used with one or more signers. For clarity, we confine our description to the case of a single digital signature. The steps for preparing a signedData MIME entity are as follows:

* Select a message digest algorithm (SHA or MD5).
* Compute the message digest, or hash function, of the content to be signed.
* Encrypt the message digest with the signer's private key.
* Prepare a block known as SignerInfo that contains the signer's public-key certificate, an identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

The signedData entity consists of a series of blocks, including a message digest algorithm identifier, the message being signed, and SignerInfo. The signedData entity may also include a set of public-key certificates sufficient to constitute a chain from a recognized root or top-level certification authority to the signer. This information is then encoded into base64.

47

To recover the signed message and verify the signature, the recipient first strips off the base64 encoding. Then the signer's public key is used to decrypt the message digest. The recipient independently computes the message digest and compares it to the decrypted message digest to verify the signature.

***iii)Clear Signing***

1. Clear signing is achieved using the multipart content type with a signed subtype.
2. As was mentioned, this signing process does not involve transforming the message to be signed, so that the message is sent "in the clear."
3. Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.

A multipart/signed message has two parts. The first part can be any MIME type but must be prepared so that it will not be altered during transfer from source to destination. This means that if the first part is not 7bit, then it needs to be encoded using base64 or quoted-printable. Then this part is processed in the same manner as signedData, but in this case an object with signedData format is created that has an empty message content field. This object is a detached signature. It is then transfer encoded using base64 to become the second part of the multipart/signed message. This second part has a MIME content type of application and a subtype of pkcs7-signature

The protocol parameter indicates that this is a two-part clear-signed entity. The receiver can verify the signature by taking the message digest of the first part and comparing this to the message digest recovered from the signature in the second part.

***Registration Request***

* Typically, an application or user will apply to a certification authority for a public-key certificate.
* The application/pkcs10 S/MIME entity is used to transfer a certification request. The certification request includes certificationRequestInfo block, followed by an

identifier of the public-key encryption algorithm, followed by the signature of the certificationRequestInfo block, made using the sender's private key.

* The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the

user's public key.

48

***Certificates-Only Message***

A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request. The message is an application/pkcs7-mime type/subtype with an smime-type parameter of degenerate. The steps involved are the same as those for creating a signedData message, except that there is no message content and the signerInfo field is empty.

**S/MIME Certificate Processing**

S/MIME uses public-key certificates that conform to version 3 of X.509 The key-management scheme used by S/MIME is in some ways a hybrid between a strict X.509 certification hierarchy and PGP's web of trust. As with the PGP model, S/MIME managers and/or users must configure each client with a list of trusted keys and with certificate revocation lists.

***\*User Agent Role***

An S/MIME user has **several key-management functions** to perform:

* **Key generation:** The user of some related administrative utility (e.g., oneassociated with LAN management) MUST be capable of generating a key pair from a good source of nondeterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.
* **Registration:** A user's public key must be registered with a certification authorityin order to receive an X.509 public-key certificate.
* **Certificate storage and retrieval:** A user requires access to a local list ofcertificates in order to verify incoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

***\*VeriSign Certificates***

There are several companies that provide certification authority (CA) services. For example, Nortel has designed an enterprise CA solution and can provide S/MIME support within an organization. There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service. Of these, the most widely used is the VeriSign CA service, a brief description of which we now provide.

49

VeriSign provides a CA service that is intended to be compatible with S/MIME and a variety of other applications. VeriSign issues X.509 certificates with the product name VeriSign Digital ID. As of early 1998, over 35,000 commercial Web sites were using VeriSign Server Digital IDs, and over a million consumer Digital IDs had been issued to users of Netscape and Microsoft browsers.

**The information contained in a Digital ID** depends on the type of Digital ID and itsuse. At a minimum, each Digital ID contains

* Owner's public key
* Owner's name or alias
* Expiration date of the Digital ID
* Serial number of the Digital ID
* Name of the certification authority that issued the Digital ID
* Digital signature of the certification authority that issued the Digital ID

**Digital IDs can also contain other user-supplied information, including**

* Address
* E-mail address
* Basic registration information (country, zip code, age, and gender)

**VeriSign** provides three levels, or classes, of security for public-key certificates. A userrequests a certificate online at VeriSign's Web site or other participating Web sites. Class 1 and Class 2 requests are processed on line, and in most cases take only a few seconds to approve. Briefly, the following procedures are used:

* For Class 1 Digital IDs, VeriSign confirms the user's e-mail address by sending a PIN and Digital ID pick-up information to the e-mail address provided in the application.
* For Class 2 Digital IDs, VeriSign verifies the information in the application through an automated comparison with a consumer database in addition to performing all of the checking associated with a Class 1 Digital ID. Finally, confirmation is sent to the specified postal address alerting the user that a Digital ID has been issued in his or her name.
* For Class 3 Digital IDs, VeriSign requires a higher level of identity assurance. An individual must prove his or her identity by providing notarized credentials or applying in person.

50

**Enhanced Security Services**

As of this writing, three enhanced security services have been proposed in an Internet draft.:

* **Signed receipts:** A signed receipt may be requested in a SignedData object.Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message.
* **Security labels**: A security label may be included in the authenticated attributesof a SignedData object. A security label is a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. The labels may be used for access control, by indicating which users are permitte access to an object.
* **Secure mailing lists:** When a user sends a message to multiple recipients, acertain amount of per-recipient processing is required, including the use of each recipient's public key. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA, with encryption performed using the MLA's public key.

**\* \* \* \* \* \* \* \* \* \* \* \* \* \* END OF UNIT IV \* \* \* \* \* \* \* \* \* \* \* \* \* \***

**INTRUDERS**

One of the most publicized attacks to security is the intruder, generally referred to as hacker or cracker. Three classes of intruders are as follows:

**Masquerader** – an individual who is not authorized to use the computer and who penetrates a system’s access controls to exploit a legitimate user’s account.

 **Misfeasor** – a legitimate user who accesses data, programs, or resources for which such access is not authorized, or who is authorized for such access but misuse his or her privileges.

 **Clandestine user** – an individual who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

The masquerader is likely to be an outsider; the misfeasor generally is an insider;

and the clandestine user can be either an outsider or an insider.

Intruder attacks range from the benign to the serious. At the benign end of the scale, there are many people who simply wish to explore internets and see what is out there. At the serious end are individuals who are attempting to read privileged data, perform unauthorized modifications to data, or disrupt the system. Benign intruders might be tolerable, although they do consume resources and may slow performance for legitimate users. However there is no way in advance to know whether an intruder will be benign or malign.

**An analysis of previous attack revealed that there were two levels of hackers:**

 The high levels were sophisticated users with a thorough knowledge of the technology.

 The low levels were the ‘foot soldiers’ who merely use the supplied cracking

programs with little understanding of how they work.

one of the results of the growing awareness of the intruder problem has been the establishment of a number of Computer Emergency Response Teams (CERT). these co- operative ventures collect information about system vulnerabilities and disseminate it to systems managers. Unfortunately, hackers can also gain access to CERT reports.

In addition to running password cracking programs, the intruders attempted to modify login software to enable them to capture passwords of users logging onto the systems.

**Intrusion techniques**

The objective of the intruders is to gain access to a system or to increase the range of privileges accessible on a system. Generally, this requires the intruders to acquire information that should be protected. In most cases, the information is in the form of a user password.

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

 **One way encryption** – the system stores only an encrypted form of user’s password. In practice, the system usually performs a one way transformation (not reversible) in which the password is used to generate a key for the encryption function and in which a fixed length output is produced.

 **Access control** – access to the password file is limited to one or a very few accounts.

**The following techniques are used for learning passwords.**

 Try default passwords used with standard accounts that are shipped with the system. Many administrators do not bother to change these defaults.

 Exhaustively try all short passwords.

 Try words in the system’s online dictionary or a list of likely passwords.

 Collect information about users such as their full names, the name of their spouse and children, pictures in their office and books in their office that are related to hobbies.

 Try user’s phone number, social security numbers and room numbers.

 Try all legitimate license plate numbers.

 Use a torjan horse to bypass restriction on access.

 Tap the line between a remote user and the host system. Two principle countermeasures:

 Detection – concerned with learning of an attack, either before or after its success.

 Prevention – challenging security goal and an uphill bottle at all times.

**INTRUSION DETECTION:**

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

1. If an intrusion is detected quickly enough, the intruder can be identified and ejected from the system before any damage is done or any data are compromised.

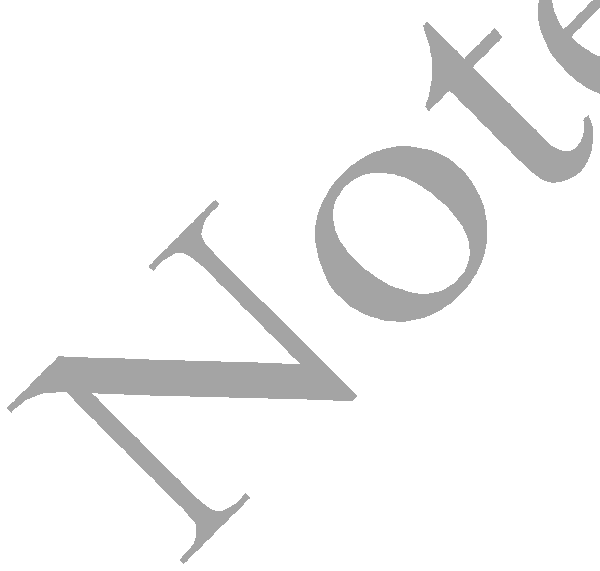
2. An effective intrusion detection system can serve as a deterrent, so acting to prevent intrusions.

3. Intrusion detection enables the collection of information about intrusion techniques that can be used to strengthen the intrusion prevention facility.

Intrusion detection is based on the assumption that the behavior of the intruder differs from that of a legitimate user in ways that can be quantified.

[Figure 18.1 s](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec2.html#ch18fig01)uggests, in very abstract terms, the nature of the task confronting the designer of an intrusion detection system. Although the typical behavior of an intruder differs from the typical behavior of an authorized user, there is an overlap in these behaviors. Thus, a loose interpretation of intruder behavior, which will catch more intruders, will also lead to a number of "false positives," or authorized users identified as intruders. On the other hand, an attempt to limit false positives by a tight interpretation of intruder behavior will lead to an increase in false negatives, or intruders not identified as intruders. Thus, there is an element of compromise and art in the practice of intrusion detection.

**[**[**PORR92]**](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_218) **identifies the following approaches to intrusion detection:**

1. **Statistical anomaly detection**: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

a. **Threshold detection**: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.

b. **Profile based:** A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. **Rule-based detection**: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

a. **Anomaly detection**: Rules are developed to detect deviation from previous

usage patterns.

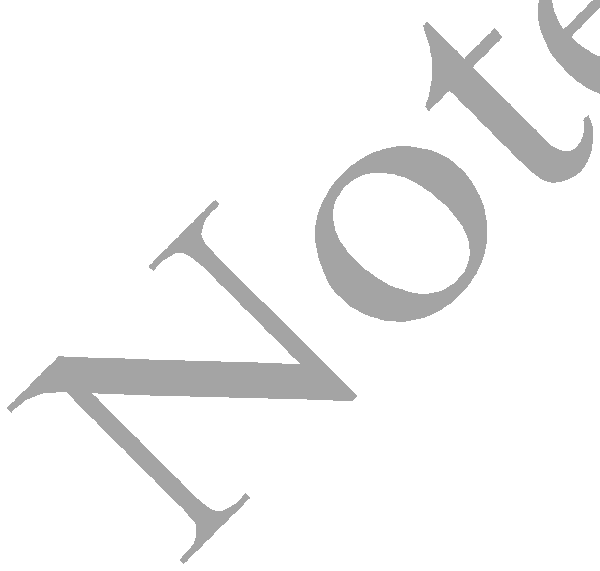
b. **Penetration identification**: An expert system approach that searches for suspicious behavior.

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders. On the other hand, such techniques may be unable to deal with misfeasors. For such attacks, rule-based approaches may be able to recognize events and sequences that, in context, reveal penetration. In practice, a system may exhibit a combination of both approaches to be effective against a broad range of attacks.

**Audit Records**

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

 **Native audit records**: Virtually all multiuser operating systems include accounting software that collects information on user activity. The advantage of using this information is that no additional collection software is needed. The disadvantage is that the native audit records may not contain the needed information or may not contain it in a convenient form.

 **Detection-specific audit records**: A collection facility can be implemented that generates audit records containing only that information required by the intrusion detection system. One advantage of such an approach is that it could be made vendor independent and ported to a variety of systems. The disadvantage is the extra overhead involved in having, in effect, two accounting packages running on a machine.

**Each audit record contains the following fields:**

 **Subject:** Initiators of actions. A subject is typically a terminal user but might also be a process acting on behalf of users or groups of users.

 **Object:** Receptors of actions. Examples include files, programs, messages, records,

terminals, printers, and user- or program-created structures

 **Resource-Usage**: A list of quantitative elements in which each element gives the amount used of some resource (e.g., number of lines printed or displayed, number of records read or written, processor time, I/O units used, session elapsed time).

 **Time-Stamp**: Unique time-and-date stamp identifying when the action took place.

Most user operations are made up of a number of elementary actions. For example, a file copy involves the execution of the user command, which includes doing access validation and setting up the copy, plus the read from one file, plus the write to another file. Consider the command

COPY GAME.EXE TO <Library>GAME.EXE

issued by Smith to copy an executable file GAME from the current directory to the

<Library> directory. The following audit records may be generated:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Smith | execute | <Library>COPY.EXE | 0 | CPU = 00002 | 11058721678 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Smith | read | <Smith>GAME.EXE | 0 | RECORDS = 0 | 11058721679 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Smith | execute | <Library>COPY.EXE | write-viol | RECORDS = 0 | 11058721680 |

In this case, the copy is aborted because Smith does not have write permission to <Library>. The decomposition of a user operation into elementary actions has three advantages:

1. Because objects are the protectable entities in a system, the use of elementary actions enables an audit of all behavior affecting an object. Thus, the system can detect attempted subversions of access

2. Single-object, single-action audit records simplify the model and the implementation.

3. Because of the simple, uniform structure of the detection-specific audit records, it may be relatively easy to obtain this information or at least part of it by a straightforward mapping from existing native audit records to the detection-specific audit records.

 **Statistical Anomaly Detection:**

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

Threshold analysis, by itself, is a crude and ineffective detector of even moderately sophisticated attacks. Both the threshold and the time interval must be determined.

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

The foundation of this approach is an analysis of audit records. The audit records provide input to the intrusion detection function in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. Examples of metrics that are useful for profile-based intrusion detection are the following:

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

 **Gauge:** A nonnegative integer that may be incremented or decremented. Typically, a gauge is used to measure the current value of some entity. Examples include the

number of logical connections assigned to a user application and the number of outgoing messages queued for a user process.

 **Interval timer**: The length of time between two related events. An example is the length of time between successive logins to an account.

 **Resource utilization**: Quantity of resources consumed during a specified period.

Examples include the number of pages printed during a user session and total time consumed by a program execution.

Given these general metrics, various tests can be performed to determine whether current activity fits within acceptable limits. [[DENN87]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_065) lists the following approaches that may be taken:

 Mean and standard deviation

 Multivariate

 Markov process

 Time series

 Operational

The simplest statistical test is to measure the mean and standard deviation of a parameter over some historical period. This gives a reflection of the average behavior and its variability.

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

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

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

Finally, an operational model is based on a judgment of what is considered abnormal, rather

than an automated analysis of past audit records. Typically, fixed limits are defined and intrusion is suspected for an observation that is outside the limits.

 **Rule-Based Intrusion Detection**

Rule-based techniques detect intrusion by observing events in the system and applying a set of rules that lead to a decision regarding whether a given pattern of activity is or is not suspicious.

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

As with statistical anomaly detection, rule-based anomaly detection does not require knowledge of security vulnerabilities within the system. Rather, the scheme is based on observing past behavior and, in effect, assuming that the future will be like the past

**Rule-based penetration identification** takes a very different approach to intrusion detection, one based on expert system technology. The key feature of such systems is the use of rules for identifying known penetrations or penetrations that would exploit known weaknesses.

Example heuristics are the following:

1. Users should not read files in other users' personal directories.

2. Users must not write other users' files.

3. Users who log in after hours often access the same files they used earlier.

4. Users do not generally open disk devices directly but rely on higher-level operating system utilities.

5. Users should not be logged in more than once to the same system.

6. Users do not make copies of system programs.

**The Base-Rate Fallacy**

To be of practical use, an intrusion detection system should detect a substantial percentage of intrusions while keeping the false alarm rate at an acceptable level. If only a modest percentage of actual intrusions are detected, the system provides a false sense of security. On the other hand, if the system frequently triggers an alert when there is no intrusion (a false alarm), then either system managers will begin to ignore the alarms, or much time will be wasted analyzing the false alarms.

Unfortunately, because of the nature of the probabilities involved, it is very difficult to meet the standard of high rate of detections with a low rate of false alarms. In general, if the actual numbers of intrusions is low compared to the number of legitimate uses of a system, then the false alarm rate will be high unless the test is extremely discriminating.

**Distributed Intrusion Detection**

Until recently, work on intrusion detection systems focused on single-system stand-alone facilities. The typical organization, however, needs to defend a distributed collection of hosts supported by a LAN Porras points out the following major issues in the design of a distributed intrusion detection system

 A distributed intrusion detection system may need to deal with different audit record formats. In a heterogeneous environment, different systems will employ different native audit collection systems and, if using intrusion detection, may employ different formats for security-related audit records.

 One or more nodes in the network will serve as collection and analysis points for the data from the systems on the network. Thus, either raw audit data or summary data must be transmitted across the network. Therefore, there is a requirement to assure the integrity and confidentiality of these data.

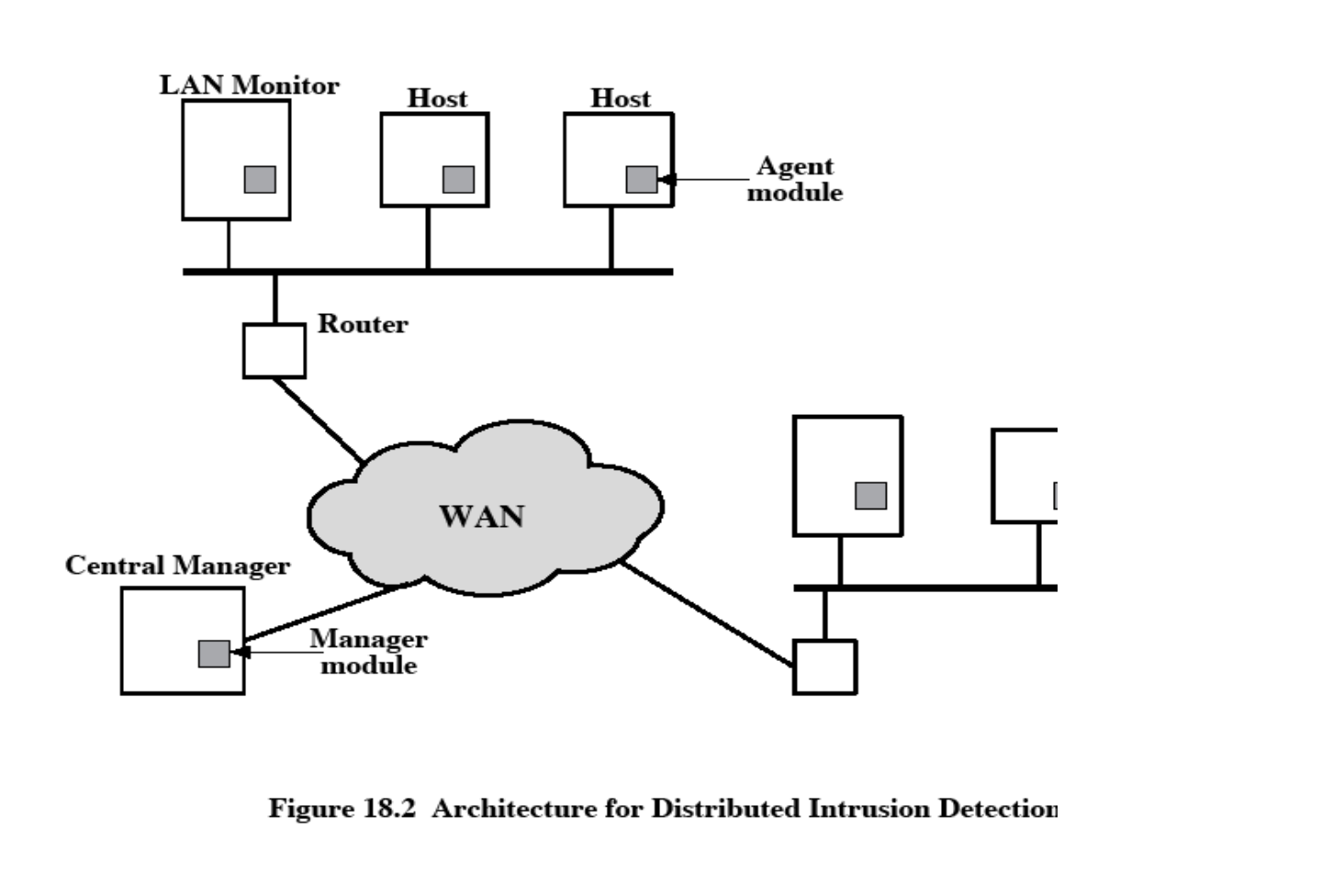
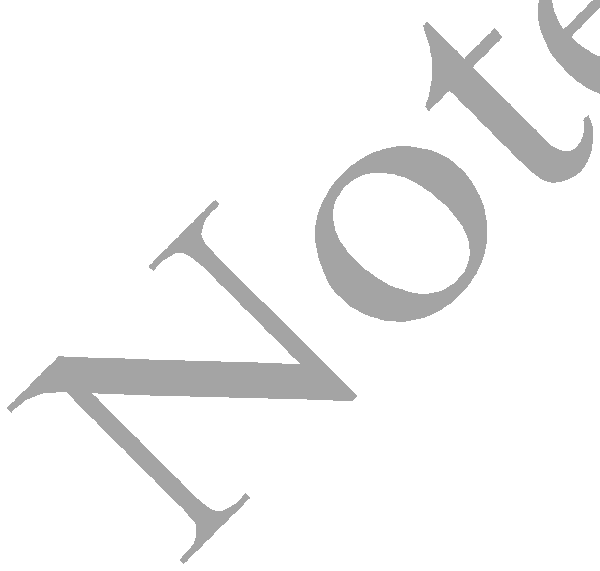
 Either a centralized or decentralized architecture can be used.

[Figure 18.2 shows](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec2.html#ch18fig02) the overall architecture, which consists of three main components:

 **Host agent module:** An audit collection module operating as a background process on a monitored system. Its purpose is to collect data on security-related events on the host and transmit these to the central manager.

 **LAN monitor agent module:** Operates in the same fashion as a host agent module except that it analyzes LAN traffic and reports the results to the central manager.

 **Central manager module:** Receives reports from LAN monitor and host agents and



processes and correlates these reports to detect intrusion.

The scheme is designed to be independent of any operating system or system auditing implementation. [Figure 18.3 shows](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec2.html#ch18fig03) the general approach that is taken.

 The agent captures each audit record produced by the native audit collection system.

 A filter is applied that retains only those records that are of security interest.

 These records are then reformatted into a standardized format referred to as the host audit record (HAR).

 Next, a template-driven logic module analyzes the records for suspicious activity.

 At the lowest level, the agent scans for notable events that are of interest independent of any past events.

 Examples include failed file accesses, accessing system files, and changing a file's access control.

 At the next higher level, the agent looks for sequences of events, such as known attack patterns (signatures).

 Finally, the agent looks for anomalous behavior of an individual user based on a historical profile of that user, such as number of programs executed, number of files accessed, and the like.

 When suspicious activity is detected, an alert is sent to the central manager.

 The central manager includes an expert system that can draw inferences from received data.

 The manager may also query individual systems for copies of HARs to correlate with those from other agents.

 The LAN monitor agent also supplies information to the central manager.

 The LAN monitor agent audits host-host connections, services used, and volume of traffic.

 It searches for significant events, such as sudden changes in network load, the use of security-related services, and network activities such as rlogin.

The architecture depicted in [Figures 18.2 a](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec2.html#ch18fig02)nd [18.3 is](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec2.html#ch18fig03) quite general and flexible. It offers a foundation for a machine-independent approach that can expand from stand-alone intrusion detection to a system that is able to correlate activity from a number of sites and networks to detect suspicious activity that would otherwise remain undetected.

**Honeypots**

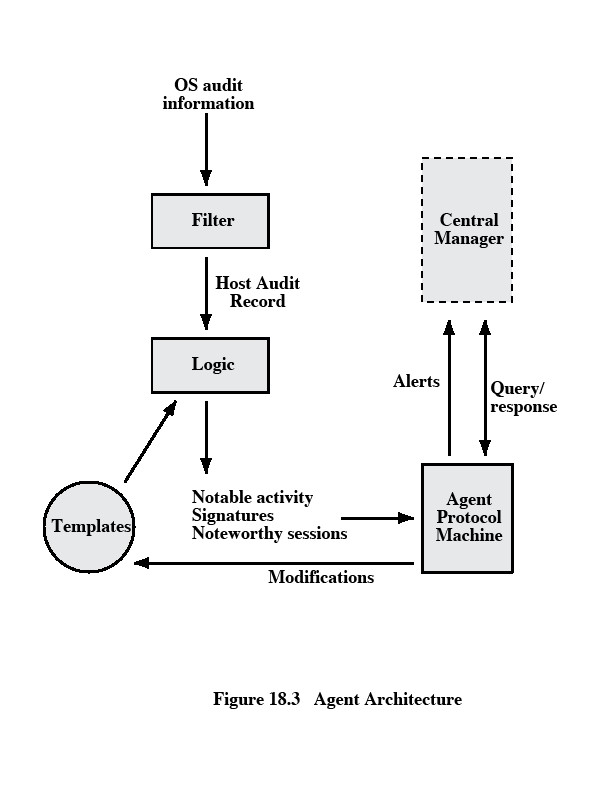
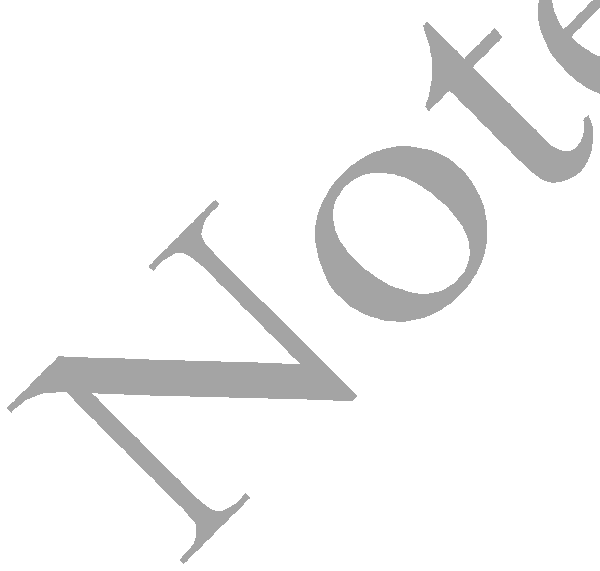
A relatively recent innovation in intrusion detection technology is the honeypot. Honeypots

are decoy systems that are designed to lure a potential attacker away from critical systems. Honeypots are designed to

 divert an attacker from accessing critical systems

 collect information about the attacker's activity

 encourage the attacker to stay on the system long enough for administrators to respond



These systems are filled with fabricated information designed to appear valuable but that a legitimate user of the system wouldn't access. Thus, any access to the honeypot is suspect.

**Intrusion Detection Exchange Format**

To facilitate the development of distributed intrusion detection systems that can function across a wide range of platforms and environments, standards are needed to support interoperability. Such standards are the focus of the IETF Intrusion Detection Working Group. The outputs of this working group include the following:

1. A requirements document, which describes the high-level functional requirements for communication between intrusion detection systems and with management systems, including the rationale for those requirements.

2. A common intrusion language specification, which describes data formats that satisfy the requirements.

3. A framework document, which identifies existing protocols best used for communication between intrusion detection systems, and describes how the devised

data formats relate to them.

**Password Protection**

**PASSWORD MANAGEMENT**

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

 The ID determines whether the user is authorized to gain access to a system.

 The ID determines the privileges accorded to the user.

 The ID is used in ,what is referred to as discretionary access control. For example, by listing the IDs of the other users, a user may grant permission to them to read files owned by that user.

***The Vulnerability of Passwords***

To understand the nature of the threat to password-based systems, let us consider a scheme that is widely used on UNIX, the following procedure is employed ([Figure 18.4a).](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec3.html#ch18fig04)

 Each user selects a password of up to eight printable characters in length.

 This is converted into a 56-bit value (using 7-bit ASCII) that serves as the key input to an encryption routine.

 The encryption routine, known as crypt(3), is based on DES. The DES algorithm is modified using a 12-bit "salt" value.

 Typically, this value is related to the time at which the password is assigned to the user.

The modified DES algorithm is exercised with a data input consisting of a 64-bit block of zeros.

 The output of the algorithm then serves as input for a second encryption.

 This process is repeated for a total of 25 encryptions.

 The resulting 64-bit output is then translated into an 11-character sequence.

 The hashed password is then stored, together with a plaintext copy of the salt, in the password file for the corresponding user ID.

 This method has been shown to be secure against a variety of cryptanalytic attacks

**The salt serves three purposes:**

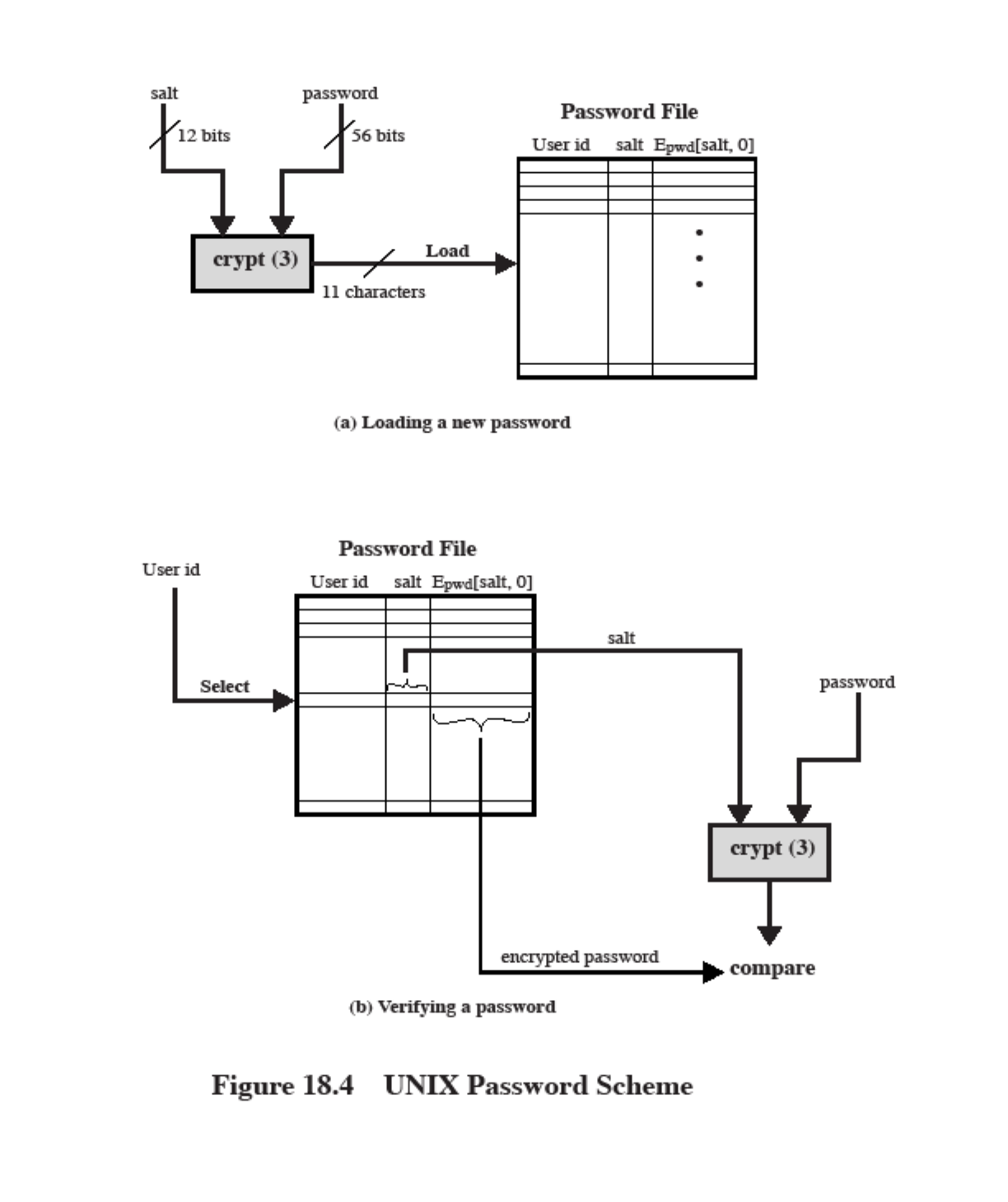
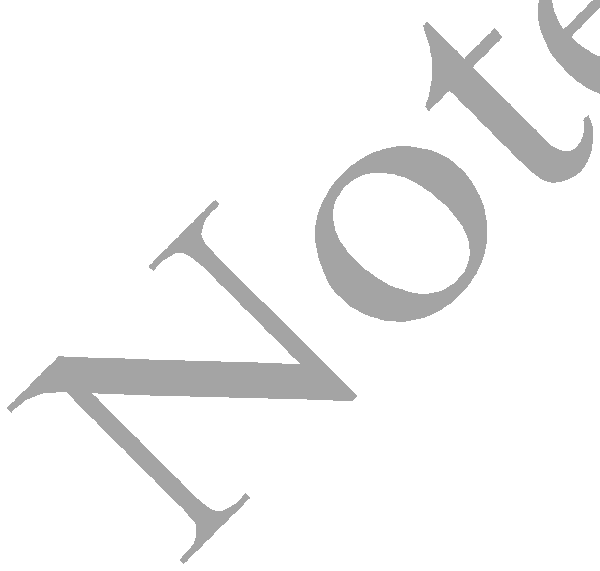
 It prevents duplicate passwords from being visible in the password file. Even if two users choose the same password, those passwords will be assigned at different times. Hence, the "extended" passwords of the two users will differ.

 It effectively increases the length of the password without requiring the user to remember two additional characters.

 It prevents the use of a hardware implementation of DES, which would ease the difficulty of a brute-force guessing attack.

When a user attempts to log on to a UNIX system, the user provides an ID and a password. The operating system uses the ID to index into the password file and retrieve the plaintext salt and the encrypted password. The salt and user-supplied password are used as input to the

encryption routine. If the result matches the stored value, the password is accepted.The encryption routine is designed to discourage guessing attacks. Software implementations of DES are slow compared to hardware versions, and the use of 25 iterations multiplies the time required by 25.



Thus, there are two threats to the UNIX password scheme. First, a user can gain access on a machine using a guest account or by some other means and then run a password guessing program, called a password cracker, on that machine.

As an example**, a password cracker was reported on the Internet in** August 1993 [[MADS93]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_165). Using a Thinking Machines Corporation parallel computer, a performance of

1560 encryptions per second per vector unit was achieved. With four vector units per processing node (a standard configuration), this works out to 800,000 encryptions per second on a 128-node machine (which is a modest size) and 6.4 million encryptions per second on a

1024-node machine.

Password length is only part of the problem. Many people, when permitted to choose their own password, pick a password that is guessable, such as their own name, their street name, a common dictionary word, and so forth. This makes the job of password cracking straightforward. Following strategy was used:

1. Try the user's name, initials, account name, and other relevant personal information.

In all, 130 different permutations for each user were tried.

2. Try words from various dictionaries.

3. Try various permutations on the words from step 2.

4. Try various capitalization permutations on the words from step 2 that were not considered in step 3. This added almost 2 million additional words to the list.

***Access Control***

One way to thwart a password attack is to deny the opponent access to the password file. If the encrypted password portion of the file is accessible only by a privileged user, then the opponent cannot read it without already knowing the password of a privileged user.

**Password Selection Strategies**

Four basic techniques are in use:

 User education

 Computer-generated passwords

 Reactive password checking

 Proactive password checking

Users can be told the importance of using hard-to-guess passwords and can be provided with guidelines for selecting strong passwords. This **user education** strategy is unlikely to succeed at most installations, particularly where there is a large user population or a lot of turnover. Many users will simply ignore the guidelines

**Computer-generated passwords** also have problems. If the passwords are quite random in nature, users will not be able to remember them. Even if the password is pronounceable, the user may have difficulty remembering it and so be tempted to write it down

**A reactive password** checking strategy is one in which the system periodically runs its own password cracker to find guessable passwords.

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

We look at possible approaches to proactive password checking.

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

 All passwords must be at least eight characters long.

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

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

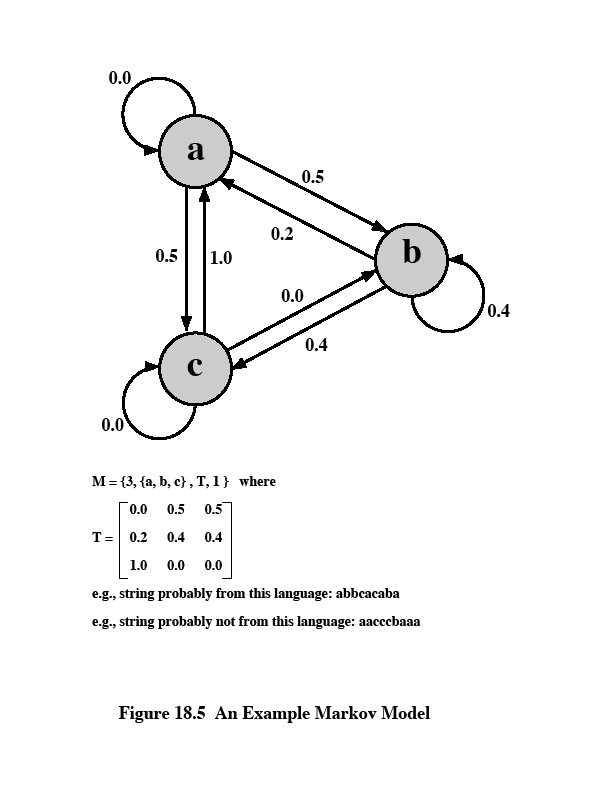
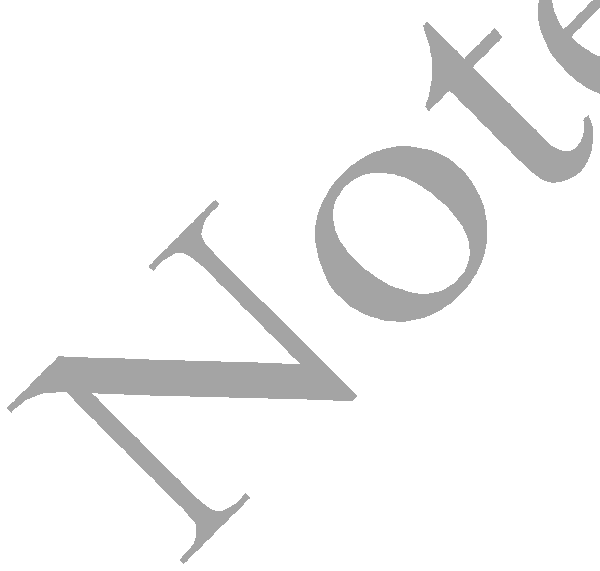
Another possible procedure is simply to compile a large dictionary of possible "bad" passwords. When a user selects a password, the system checks to make sure that it is not on the disapproved list. There are two problems with this approach:

 Space: The dictionary must be very large to be effective..

 Time: The time required to search a large dictionary may itself be large

Two techniques for developing an effective and efficient proactive password checker that is based on rejecting words on a list show promise. One of these develops a Markov model for the generation of guessable passwords [[DAVI93]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_060). [Figure 18.5 shows](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec3.html#ch18fig05) a simplified version of such a model. This model shows a language consisting of an alphabet of three characters. The state of the system at any time is the identity of the most recent letter. The value on the transition from one state to another represents the probability that one letter follows another. Thus, the probability that the next letter is b, given that the current letter is a, is 0.5.

In general, a Markov model is a quadruple [m, A, T, k], where m is the number of states in the model, A is the state space, T is the matrix of transition probabilities, and k is the order of the model. For a kth-order model, the probability of making a transition to a particular letter depends on the previous k letters that have been generated. [Figure 18.5 shows](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch18lev1sec3.html#ch18fig05) a simple first- order model.



The authors report on the development and use of a second-order model. To begin, a dictionary of guessable passwords is constructed. Then the transition matrix is calculated as follows:

1. Determine the frequency matrix f, where f(i, j, k) is the number of occurrences of the trigram consisting of the ith, jth, and kth character. For example, the password parsnips yields the trigrams par, ars, rsn, sni, nip, and ips.

2. For each bigram ij, calculate f(i, j,∞) as the total number of trigrams beginning with

ij. For example, f(a, b,∞) would be the total number of trigrams of the form aba, abb, abc, and so on.

3. Compute the entries of T as follows: T(i,j,k) = f(i, j, k) / f(i, j,∞)

The result is a model that reflects the structure of the words in the dictionary.

A quite different approach has been reported by Spafford [[SPAF92a,](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_257) [SPAF92b]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_258). It is based on the use of a Bloom filter [[BLOO70]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_034). To begin, we explain the operation of the Bloom filter. A Bloom filter of order k consists of a set of k independent hash functions H1(x), H2(x),..., Hk(x), where each function maps a password into a hash value in the range 0 to N -

1 That is,

Hi(Xj) = y 1 ≤i ≤k; 1 ≤j ≤D; 0 ≤y ≤N- 1 where

Xj = jth word in password dictionary

D = number of words in password dictionary

The following procedure is then applied to the dictionary:

**1** A hash table of N bits is defined, with all bits initially set to 0.

**2** For each password, its k hash values are calculated, and the corresponding bits in the hash table are set to 1. Thus, if Hi(Xj) = 67 for some (i, j), then the sixty-seventh bit of the hash table is set to 1; if the bit already has the value 1, it remains at 1.

When a new password is presented to the checker, its k hash values are calculated. If all the corresponding bits of the hash table are equal to 1, then the password is rejected.

**FIREWALLS**

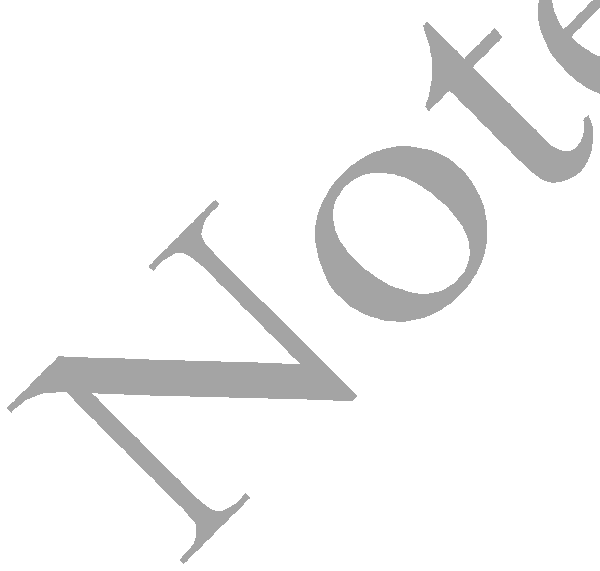
**Firewall design principles**

Internet connectivity is no longer an option for most organizations. However, while internet access provides benefits to the organization, it enables the outside world to reach and interact with local network assets. This creates the threat to the organization. While it is possible to equip each workstation and server on the premises network with strong security features, such as intrusion protection, this is not a practical approach. The alternative, increasingly accepted, is the firewall.

The firewall is inserted between the premise network and internet to establish a controlled link and to erect an outer security wall or perimeter. The aim of this perimeter is to protect the premises network from internet based attacks and to provide a single choke point where security and audit can be imposed. The firewall can be a single computer system or a set of two or more systems that cooperate to perform the firewall function.

**Firewall characteristics:**

 All traffic from inside to outside, and vice versa, must pass through the firewall.

This is achieved by physically blocking all access to the local network except via the firewall. Various configurations are possible.

 Only authorized traffic, as defined by the local security policy, will be allowed to pass. Various types of firewalls are used, which implement various types of security policies.

 The firewall itself is immune to penetration. This implies that use of a trusted system with a secure operating system. This implies that use of a trusted system with a secure operating system.

Four techniques that firewall use to control access and enforce the site’s security policy is

as follows:

 Service control – determines the type of internet services that can be accessed, inbound or outbound. The firewall may filter traffic on this basis of IP address and TCP port number; may provide proxy software that receives and interprets each service request before passing it on; or may host the server software itself, such as web or mail service.

 Direction control – determines the direction in which particular service request may be initiated and allowed to flow through the firewall.

 User control – controls access to a service according to which user is attempting to access it.

 Behavior control – controls how particular services are used.

**Capabilities of firewall**

A firewall defines a single choke point that keeps unauthorized users out of the protected network, prohibits potentially vulnerable services from entering or leaving the network, and provides protection from various kinds of IP spoofing and routing attacks.

 A firewall provides a location for monitoring security related events. Audits and alarms can be implemented on the firewall system.

 A firewall is a convenient platform for several internet functions that are not security related.

 A firewall can serve as the platform for IPsec.

**Limitations of firewall**

 The firewall cannot protect against attacks that bypass the firewall. Internal systems may have dial-out capability to connect to an ISP. An internal LAN may support a modem pool that provides dial-in capability for traveling employees and telecommuters.

 The firewall does not protect against internal threats. The firewall does not protect against internal threats, such as a disgruntled employee or an employee who unwittingly cooperates with an external attacker.

 The firewall cannot protect against the transfer of virus-infected programs or files.

Because of the variety of operating systems and applications supported inside the perimeter, it would be impractical and perhaps impossible for the firewall to scan all incoming files, e-mail, and messages for viruses.

**Types of firewalls**

There are 3 common types of firewalls.

 Packet filters

 Application-level gateways

 Circuit-level gateways

**Packet filtering router**

A packet filtering router applies a set of rules to each incoming IP packet and then forwards or discards the packet. The router is typically configured to filter packets going in both directions. Filtering rules are based on the information contained in a network packet:

 Source IP address – IP address of the system that originated the IP packet.

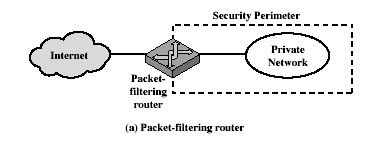
 Destination IP address – IP address of the system, the IP is trying to reach.

 Source and destination transport level address – transport level port number.

 IP protocol field – defines the transport protocol.

 Interface – for a router with three or more ports, which interface of the router the packet come from or which interface of the router the packet is destined for.

The packet filter is typically set up as a list of rules based on matches to fields in the IP or TCP header. If there is a match to one of the rules, that rule is invoked to determine whether to forward or discard the packet. If there is no match to any rule, then a default action is taken.



Two default policies are possible:

 Default = discard: That which is not expressly permitted is prohibited.

 Default = forward: That which is not expressly prohibited is permitted.

The default discard policy is the more conservative. Initially everything is blocked, and services must be added on a case-by-case basis. This policy is more visible to users, who are most likely to see the firewall as a hindrance. The default forward policy increases ease of use for end users but provides reduced security.

**Advantages of packet filter router**

 Simple

 Transparent to users

 Very fast

**Weakness of packet filter firewalls**

 Because packet filter firewalls do not examine upper-layer data, they cannot prevent attacks that employ application specific vulnerabilities or functions.

 Because of the limited information available to the firewall, the logging functionality present in packet filter firewall is limited.

 It does not support advanced user authentication schemes.

 They are generally vulnerable to attacks such as layer address spoofing.

Some of the attacks that can be made on packet filtering routers and the appropriate counter measures are the following:

 IP address spoofing – the intruders transmit packets from the outside with a source IP address field containing an address of an internal host.

Countermeasure: to discard packet with an inside source address if the packet arrives on an external interface.

 Source routing attacks – the source station specifies the route that a packet should take as it crosses the internet; i.e., it will bypass the firewall.

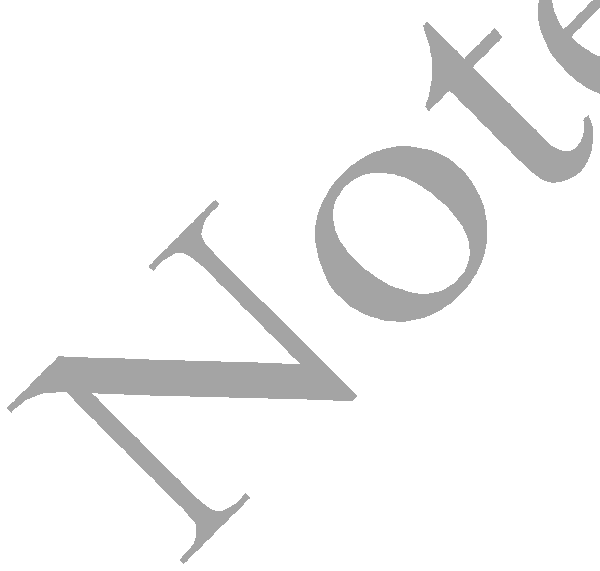
Countermeasure: to discard all packets that uses this option.

 Tiny fragment attacks – the intruder create extremely small fragments and force the TCP header information into a separate packet fragment. The attacker hopes that only the first fragment is examined and the remaining fragments are passed through.

Countermeasure: to discard all packets where the protocol type is TCP and the IP

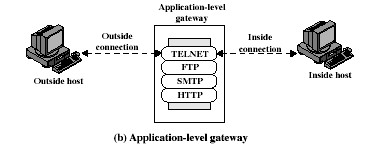
fragment offset is equal to 1.

**Application level gateway**

An Application level gateway, also called a proxy server, acts as a relay of application level traffic. The user contacts the gateway using a TCP/IP application, such as Telnet or FTP, and the gateway asks the user for the name of the remote host to be accessed. When the user responds and provides a valid user ID and authentication information, the gateway contacts the application on the remote host and relays TCP segments containing the application data between the two endpoints.

Application level gateways tend to be more secure than packet filters. It is easy to log and audit all incoming traffic at the application level. A prime disadvantage is the additional processing overhead on each connection.

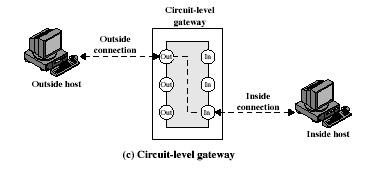
**Circuit level gateway**



Circuit level gateway can be a stand-alone system or it can be a specified function performed by an application level gateway for certain applications. A Circuit level gateway does not permit an end-to-end TCP connection; rather, the gateway sets up two TCP connections, one between itself and a TCP user on an inner host and one between itself and a TCP user on an outer host. Once the two connections are established, the gateway typically relays TCP segments from one connection to the other without examining the contents. The security function consists of determining which connections will be allowed.

A typical use of Circuit level gateways is a situation in which the system administrator trusts the internal users. The gateway can be configured to support application level or proxy service on inbound connections and circuit level functions for outbound connections.

**Basiton host**



It is a system identified by the firewall administrator as a critical strong point in the network’s security. The Bastion host serves as a platform for an application level and circuit level gateway.

Common characteristics of a Basiton host are as follows:

 The Bastion host hardware platform executes a secure version of its operating system, making it a trusted system.

 Only the services that the network administrator considers essential are installed on the Bastion host.

 It may require additional authentication before a user is allowed access to the proxy services.

 Each proxy is configured to support only a subset of standard application’s

command set.

 Each proxy is configured to allow access only to specific host systems.

 Each proxy maintains detailed audit information by logging all traffic, each connection and the duration of each connection.

 Each proxy is independent of other proxies on the Bastion host.

 A proxy generally performs no disk access other than to read its initial configuration file.

 Each proxy runs on a non privileged user in a private and secured directory on the

Bastion host.

**Firewall configurations**

There are 3 common firewall configurations.

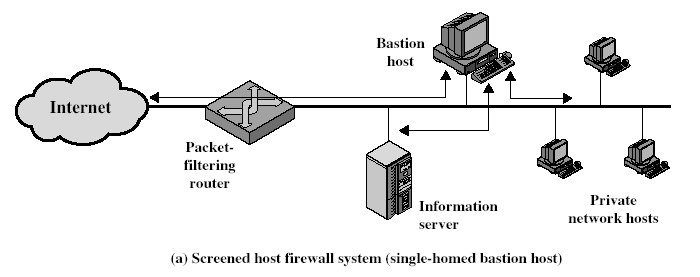
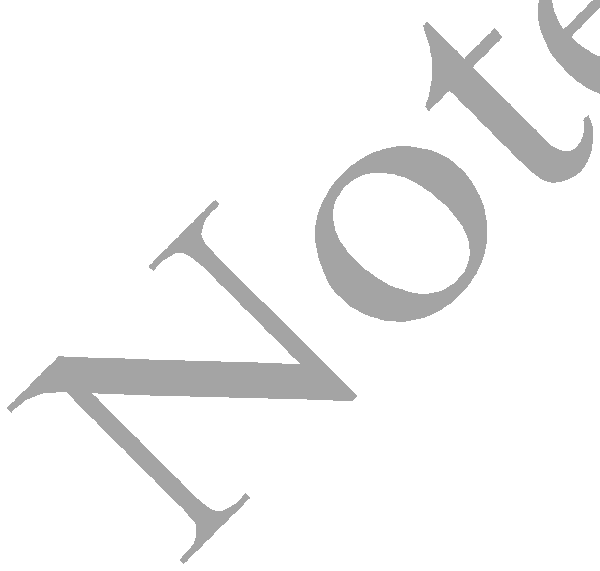
**1. Screened host firewall, single-homed basiton configuration**

In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that

 For traffic from the internet, only IP packets destined for the basiton host are allowed in.

 For traffic from the internal network, only IP packets from the basiton host are allowed out.

The basiton host performs authentication and proxy functions. This configuration has greater security than simply a packet filtering router or an application level gateway alone, for two reasons:



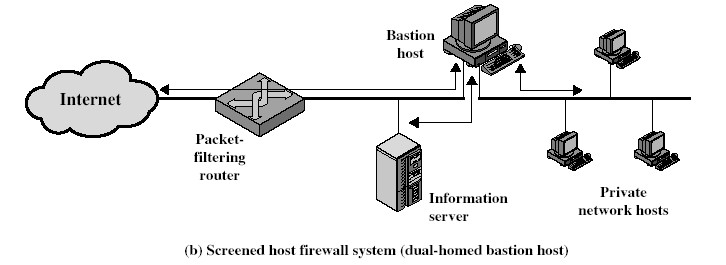
 This configuration implements both packet level and application level filtering, allowing for considerable flexibility in defining security policy.

 An intruder must generally penetrate two separate systems before the security of the internal network is compromised.

**2. Screened host firewall, dual homed basiton configuration**

In the previous configuration, if the packet filtering router is compromised, traffic could flow directly through the router between the internet and the other hosts on the private network. This configuration physically prevents such a security break.

**3. Screened subnet firewall configuration**



In this configuration, two packet filtering routers are used, one between the basiton host and internet and one between the basiton host and the internal network. This configuration creates an isolated subnetwork, which may consist of simply the basiton host but may also include one or more information servers and modems for dial-in capability. Typically both the internet and the internal network have access to hosts on the screened subnet, but traffic across the screened subnet is blocked. This configuration offers several advantages:

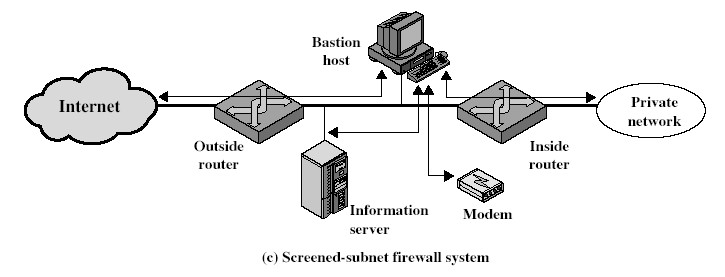
 There are now three levels of defense to thwart intruders.

 The outside router advertises only the existence of the screened subnet to the internet; therefore the internal network is invisible to the internet.

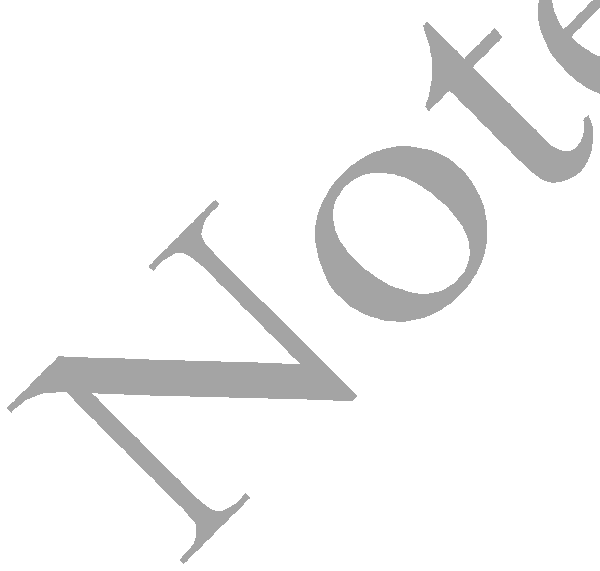
 Similarly, the inside router advertises only the existence of the screened subnet to the internal network; therefore the systems on the internal network cannot construct direct routes to the internet.

**Trusted systems**

One way to enhance the ability of a system to defend against intruders and malicious programs is to implement trusted system technology.



**Data access control**

Following successful logon, the user has been granted access to one or set of hosts and applications. This is generally not sufficient for a system that includes sensitive data in its database. Through the user access control procedure, a user can be identified to the system. Associated with each user, there can be a profile that specifies permissible operations and file accesses. The operating system can then enforce rules based on the user profile. The database management system, however, must control access to specific records or even portions of records. The operating system may grant a user permission to access a file or use an application, following which there are no further security checks, the database management system must make a decision on each individual access attempt. That decision will depend not only on the user’s identity but also on the specific parts of the data being accessed and even on the information already divulged to the user.

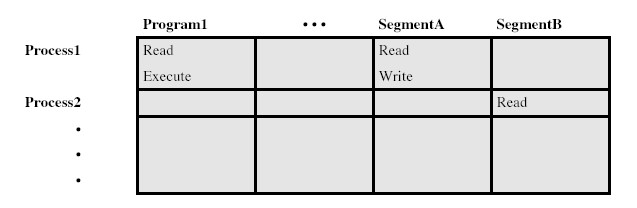
A general model of access control as exercised by an file or database management system is that of an access matrix. The basic elements of the model are as follows:

 **Subject**: An entity capable of accessing objects. Generally, the concept of subject equates with that of process.

 **Object**: Anything to which access is controlled. Examples include files, portion of files, programs, and segments of memory.

 **Access right:** The way in which the object is accessed by a subject. Examples are read, write and execute.

One axis of the matrix consists of identified subjects that may attempt data access. Typically, this list will consist of individual users or user groups. The other axis lists the objects that may be accessed. Objects may be individual data fields. Each entry in the matrix indicates the access rights of that subject for that object. The matrix may be decomposed by columns, yielding **access control lists.** Thus, for each object, an access control list lists users and their permitted access rights. The access control list may contain a default, or public, entry.



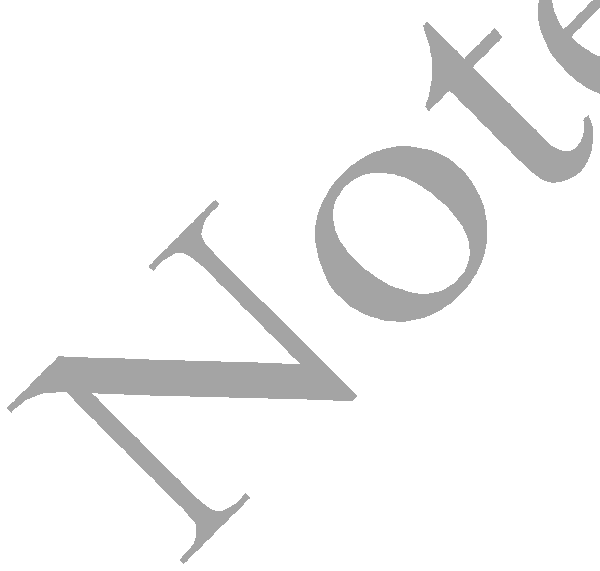
**a. Access matrix**

**Access control list for Program1:**

Process1 (Read, Execute)

**Access control list for Segment A:**

Process1 (Read, Write)



**Access control list for Segment B:**

Process2 (Read)

**b. Access control list**

**Capability list for Process1:** Program1 (Read, Execute) Segment A (Read)

**Capability list for Process2:**

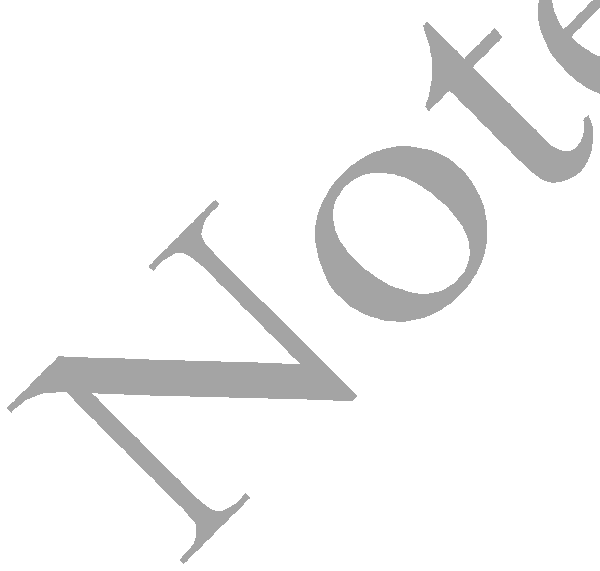
Segment B (Read)

**c. Capability list**

**Fig: Access Control Structure**

Decomposition by rows yields **capability tickets**. A capability ticket specifies authorized objects and operations for a user. Each user has a number of tickets and may be authorized to loan or give them to others. Because tickets may be dispersed around the system, they present a greater security problem than access control lists. In particular, the ticket must be unforgeable. One way to accomplish this is to have the operating system hold all tickets on behalf of users. These tickets would have to be held in a region of memory inaccessible to users.

**The concept of Trusted Systems**

When multiple categories or levels of data are defined, the requirement is referred to as multilevel security. The general statement of the requirement for multilevel security is that a subject at a high level may not convey information to a subject at a lower or noncomparable level unless that flow accurately reflects the will of an authorized user. For implementation purposes, this requirement is in two parts and is simply stated. A multilevel secure system must enforce:

 **No read up:** A subject can only read an object of less or equal security level. This is referred to as **simple security property.**

 **No write down:** A subject can only write into an object of greater or equal security level. This is referred to as **\*-property (star property).**

These two rules, if properly enforced, provide multilevel security.

**Reference Monitor concept**

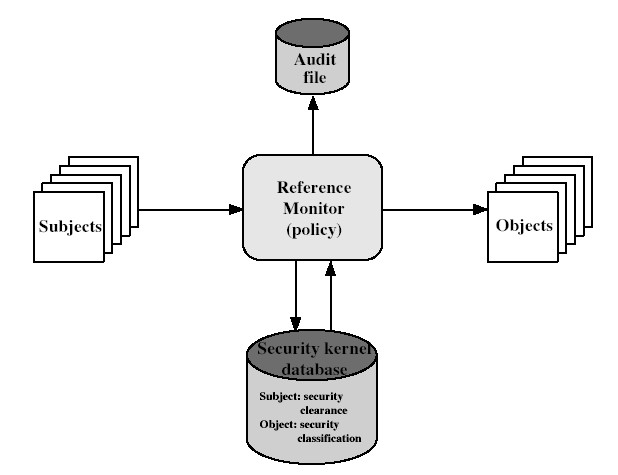
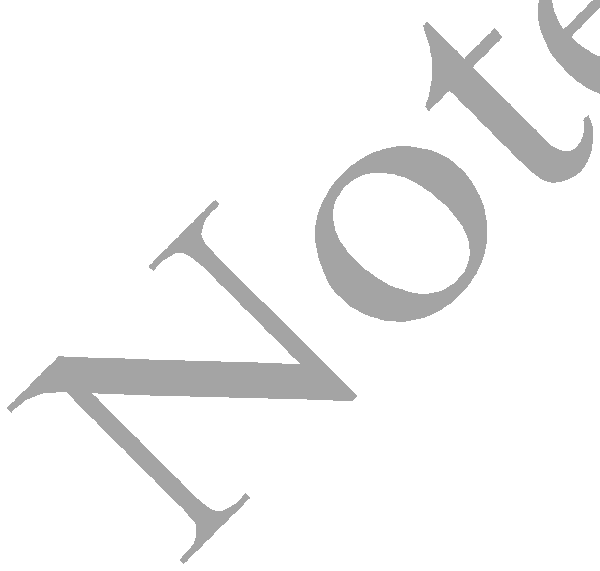
The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects to objects on the basis of

security parameters of the subject and object. The reference monitor has access to a file, known as the security kernel database that lists the access privileges (security clearance) of each subject and the protection attributes (classification level) of each object. The reference monitor enforces the security rules and has the following properties:

 Complete mediation: The security rules are enforced on every access, not just, fr example, when a file is opened.

 Isolation: The reference monitor and database are protected from unauthorised modification.

 Verifiability: The reference monitor’s correctness must be provable. That is, it must be possible to demonstrate mathematically that the reference monitor enforces the security rules and provides complete mediation and isolation. Important security events, such as detected security violations and



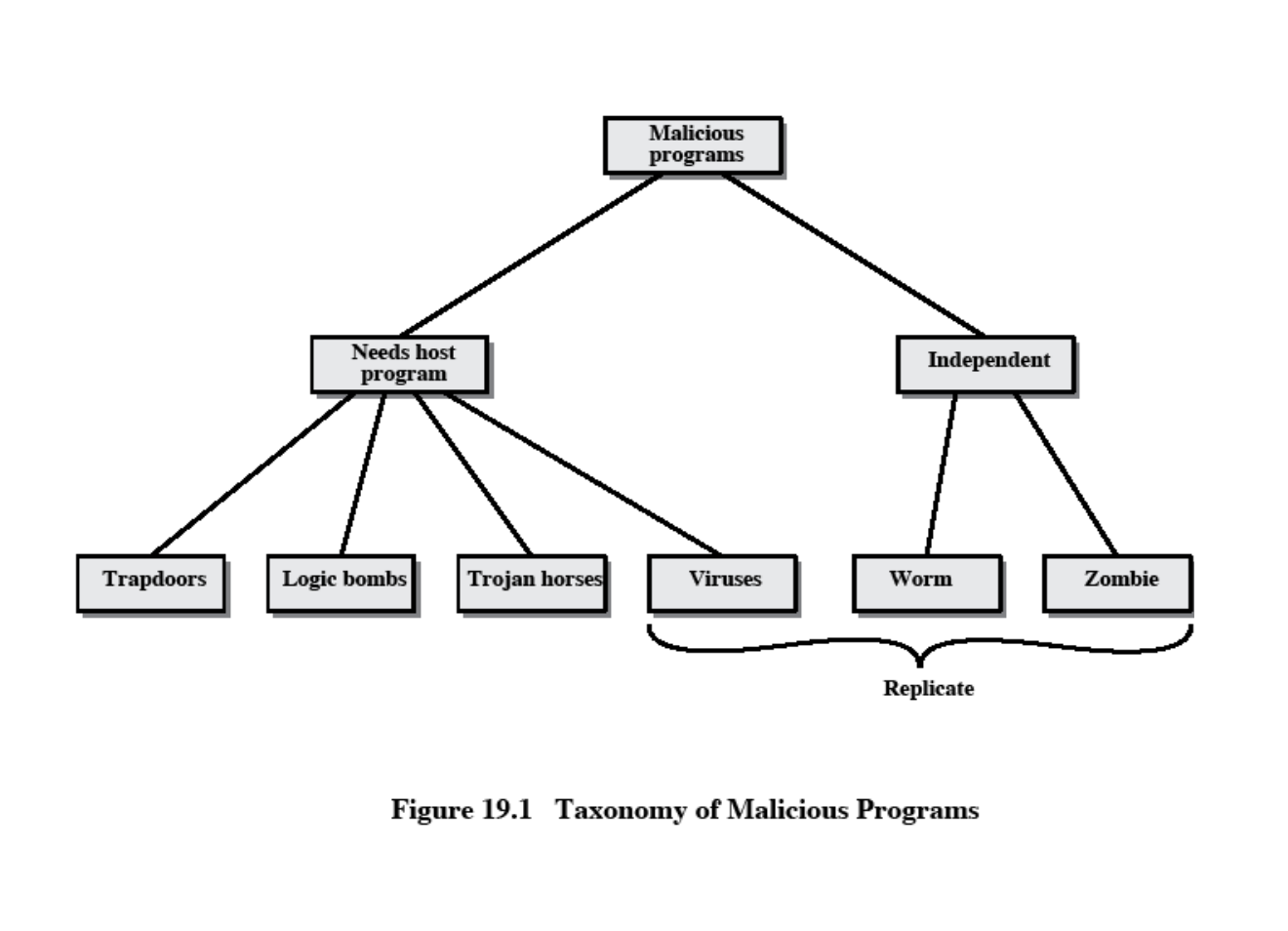
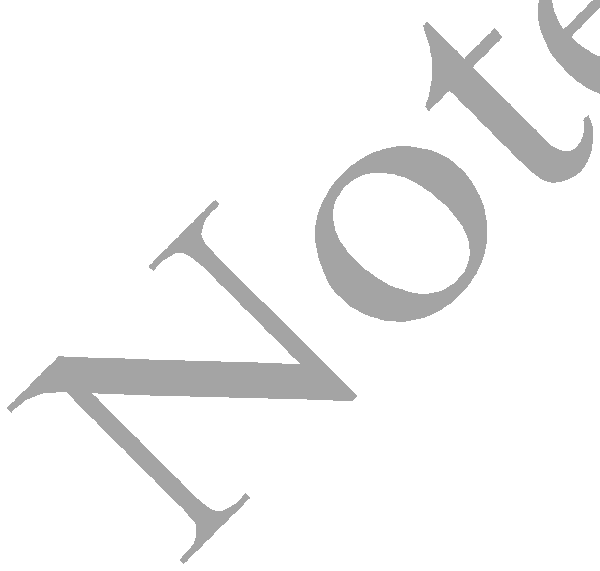
authorized changes to the security kernel database, are stored in the audit file.

**Fig: Reference Monitor Concept**

**VIRUSES AND RELATED THREATS**

Perhaps the most sophisticated types of threats to computer systems are presented by programs that exploit vulnerabilities in computing systems.

**Malicious Programs**



|  |  |
| --- | --- |
| **Name** | **Description** |
| Virus | Attaches itself to a program and propagates copies of itself to other programs |
| Worm | Program that propagates copies of itself to other computers |
| Logic bomb | Triggers action when condition occurs |

|  |  |
| --- | --- |
| Trojan horse | Program that contains unexpected additional functionality |
| Backdoor  (trapdoor) | Program modification that allows unauthorized access to functionality |
| Exploits | Code specific to a single vulnerability or set of vulnerabilities |
| Downloaders | Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail. |
| Auto-rooter | Malicious hacker tools used to break into new machines remotely |
| Kit (virus generator) | Set of tools for generating new viruses automatically |
| Spammer programs | Used to send large volumes of unwanted e-mail |
| Flooders | Used to attack networked computer systems with a large volume of traffic to carry out a denial of service (DoS) attack |
| Keyloggers | Captures keystrokes on a compromised system |
| Rootkit | Set of hacker tools used after attacker has broken into a computer system and gained root-level access |
| Zombie | Program activated on an infected machine that is activated to launch attacks on other machines |

Malicious software can be divided into two categories:

those that need a host program, and those that are independent.

The former are essentially fragments of programs that cannot exist independently of some actual application program, utility, or system program. Viruses, logic bombs, and backdoors are examples. The latter are self-contained programs that can be scheduled and run by the operating system. Worms and zombie programs are examples.

**The Nature of Viruses**

A virus is a piece of software that can "infect" other programs by modifying them; the modification includes a copy of the virus program, which can then go on to infect other programs.

A virus can do anything that other programs do. The only difference is that it attaches itself to another program and executes secretly when the host program is run. Once a virus is executing, it can perform any function, such as erasing files and programs.

During its lifetime, a typical virus goes through the following four phases:

 **Dormant phase**: The virus is idle. The virus will eventually be activated by some event, such as a date, the presence of another program or file, or the capacity of the disk exceeding some limit. Not all viruses have this stage.

 **Propagation phase:** The virus places an identical copy of itself into other programs or into certain system areas on the disk. Each infected program will now contain a clone of the virus, which will itself enter a propagation phase.

 **Triggering phase:** The virus is activated to perform the function for which it was intended. As with the dormant phase, the triggering phase can be caused by a variety of system events, including a count of the number of times that this copy of the virus has made copies of itself.

 **Execution phase:** The function is performed. The function may be harmless, such as a message on the screen, or damaging, such as the destruction of programs and data files.

***Virus Structure***

A virus can be prepended or postpended to an executable program, or it can be embedded in some other fashion. The key to its operation is that the infected program, when invoked, will first execute the virus code and then execute the original code of the program.

**An infected program begins with the virus code and works as follows.**

The first line of code is a jump to the main virus program. The second line is a special marker

that is used by the virus to determine whether or not a potential victim program has already been infected with this virus.

When the program is invoked, control is immediately transferred to the main virus program. The virus program first seeks out uninfected executable files and infects them. Next, the virus may perform some action, usually detrimental to the system.

This action could be performed every time the program is invoked, or it could be a logic bomb that triggers only under certain conditions.

Finally, the virus transfers control to the original program. If the infection phase of the program is reasonably rapid, a user is unlikely to notice any difference between the execution of an infected and uninfected program.

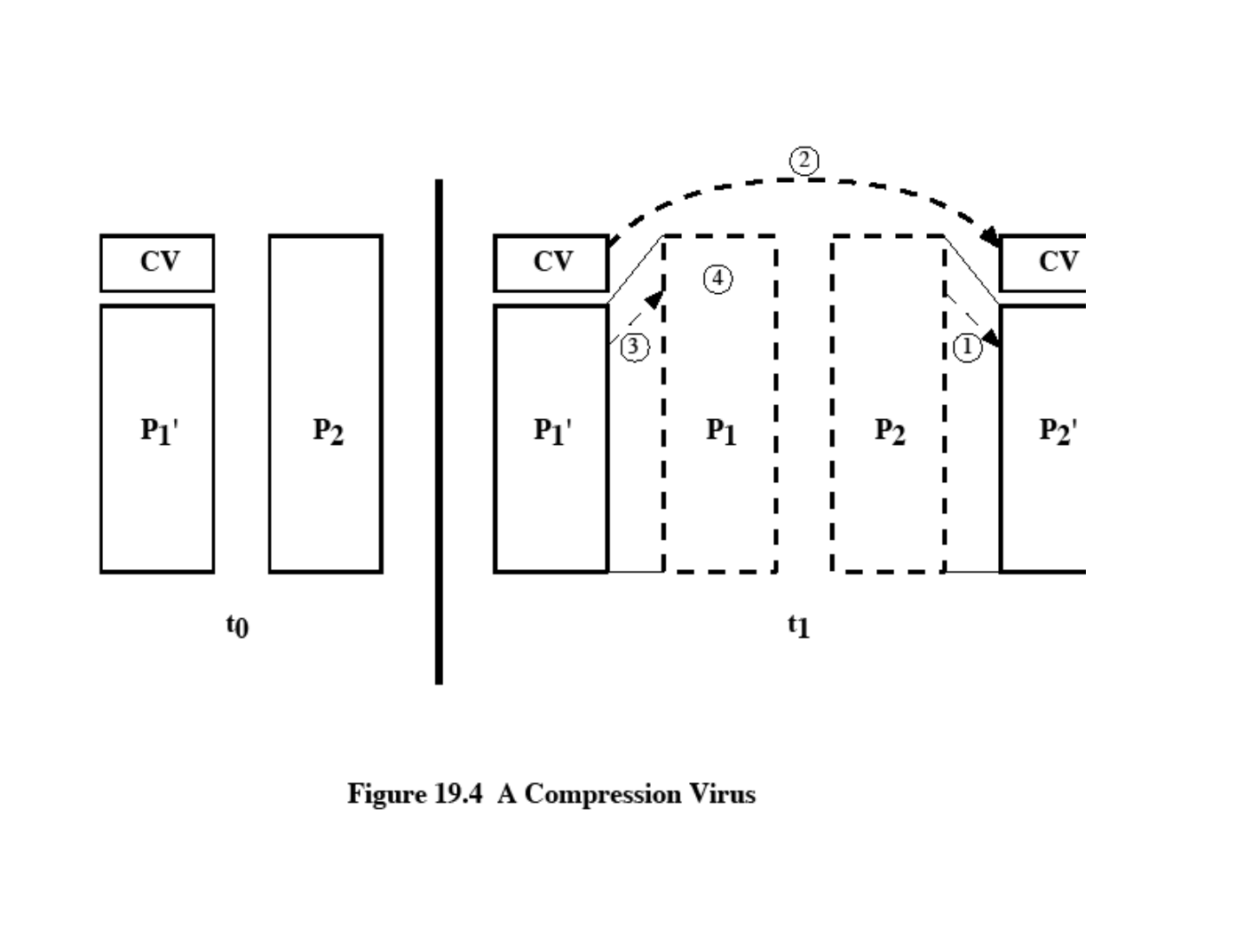
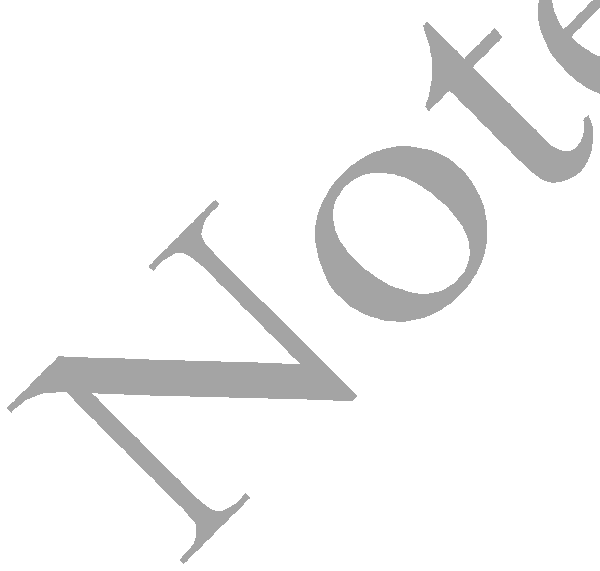
A virus such as the one just described is easily detected because an infected version of a program is longer than the corresponding uninfected one. A way to thwart such a simple means of detecting a virus is to compress the executable file so that both the infected and uninfected versions are of identical length.. The key lines in this virus are numbered, and [Figure 19.3](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch19lev1sec1.html#ch19fig03) [[COHE94]](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/app04lev1sec1.html#biblio01_051) illustrates the operation. We assume that program P1 is infected with the virus CV. When this program is invoked, control passes to its virus, which performs the following steps:

**1.** For each uninfected file P2 that is found, the virus first compresses that file to produce P'2, which is shorter than the original program by the size of the virus.

**2.** A copy of the virus is prepended to the compressed program.

**3.** The compressed version of the original infected program, P'1, is uncompressed.

**4.** The uncompressed original program is executed.



In this example, the virus does nothing other than propagate. As in the previous example, the virus may include a logic bomb.

***Initial Infection***

Once a virus has gained entry to a system by infecting a single program, it is in a position to infect some or all other executable files on that system when the infected program executes. Thus, viral infection can be completely prevented by preventing the virus from gaining entry in the first place. Unfortunately, prevention is extraordinarily difficult because a virus can be part of any program outside a system. Thus, unless one is content to take an absolutely bare piece of

iron and write all one's own system and application programs, one is vulnerable.

**Types of Viruses**

following categories as being among the most significant types of viruses:

 **Parasitic virus**: The traditional and still most common form of virus. A parasitic virus attaches itself to executable files and replicates, when the infected program is executed, by finding other executable files to infect.

 **Memory-resident virus**: Lodges in main memory as part of a resident system program.

From that point on, the virus infects every program that executes.

 **Boot sector virus**: Infects a master boot record or boot record and spreads when a system is booted from the disk containing the virus.

 **Stealth virus**: A form of virus explicitly designed to hide itself from detection by antivirus software.

 **Polymorphic virus**: A virus that mutates with every infection, making detection by the

"signature" of the virus impossible.

 **Metamorphic virus**: As with a polymorphic virus, a metamorphic virus mutates with every infection. The difference is that a metamorphic virus rewrites itself completely at each iteration, increasing the difficulty of detection. Metamorphic viruses my change their behavior as well as their appearance.

One example of a **stealth virus** was discussed earlier: a virus that uses compression so that the infected program is exactly the same length as an uninfected version. Far more sophisticated techniques are possible. For example, a virus can place intercept logic in disk I/O routines, so that when there is an attempt to read suspected portions of the disk using these routines, the virus will present back the original, uninfected program.

A **polymorphic virus** creates copies during replication that are functionally equivalent but have distinctly different bit patterns.

[Page 606]

**Macro Viruses**

In the mid-1990s, macro viruses became by far the most prevalent type of virus. Macro viruses are particularly threatening for a number of reasons:

1. A macro virus is platform independent. Virtually all of the macro viruses infect Microsoft Word documents. Any hardware platform and operating system that supports Word can be infected.

2. Macro viruses infect documents, not executable portions of code. Most of the information introduced onto a computer system is in the form of a document rather than a program.

3. Macro viruses are easily spread. A very common method is by electronic mail.

Macro viruses take advantage of a feature found in Word and other office applications such as Microsoft Excel, namely the macro. In essence, a macro is an executable program embedded in a word processing document or other type of file. Typically, users employ macros to automate repetitive tasks and thereby save keystrokes. The macro language is usually some form of the Basic programming language. A user might define a sequence of keystrokes in a macro and set it up so that the macro is invoked when a function key or special short combination of keys is input.

Successive releases of Word provide increased protection against macro viruses. For example, Microsoft offers an optional Macro Virus Protection tool that detects suspicious Word files and alerts the customer to the potential risk of opening a file with macros. Various antivirus product vendors have also developed tools to detect and correct macro viruses.

**E-mail Viruses**

A more recent development in malicious software is the e-mail virus. The first rapidly spreading e-mail viruses, such as Melissa, made use of a Microsoft Word macro embedded in an attachment. If the recipient opens the e-mail attachment, the Word macro is activated. Then

1. The e-mail virus sends itself to everyone on the mailing list in the user's e-mail package.

2. The virus does local damage.

**Worms**

A worm is a program that can replicate itself and send copies from computer to computer across network connections. Upon arrival, the worm may be activated to replicate and propagate again.

Network worm programs use network connections to spread from system to system. Once active within a system, a network worm can behave as a computer virus or bacteria, or it could implant Trojan horse programs or perform any number of disruptive or destructive actions.

To replicate itself, a network worm uses some sort of network vehicle. Examples include the following:

 Electronic mail facility: A worm mails a copy of itself to other systems.

 Remote execution capability: A worm executes a copy of itself on another system.

 Remote login capability: A worm logs onto a remote system as a user and then uses commands to copy itself from one system to the other.

The new copy of the worm program is then run on the remote system where, in addition to any functions that it performs at that system, it continues to spread in the same fashion.

A network worm exhibits the same characteristics as a computer virus: a dormant phase, a propagation phase, a triggering phase, and an execution phase. The propagation phase generally performs the following functions:

**1.** Search for other systems to infect by examining host tables or similar repositories of remote system addresses.

**2.** Establish a connection with a remote system.

**3.** Copy itself to the remote system and cause the copy to be run.

As with viruses, network worms are difficult to counter.

***The Morris Worm***

The Morris worm was designed to spread on UNIX systems and used a number of different techniques for propagation.

1. It attempted to log on to a remote host as a legitimate user. In this method, the worm first attempted to crack the local password file, and then used the discovered passwords and corresponding user IDs. The assumption was that many users would use the same password on different systems. To obtain the passwords, the worm ran a password- cracking program that tried

a. Each user's account name and simple permutations of it

b. A list of 432 built-in passwords that Morris thought to be likely candidates c. All the words in the local system directory

2. It exploited a bug in the finger protocol, which reports the whereabouts of a remote user.

3. It exploited a trapdoor in the debug option of the remote process that receives and sends mail.

If any of these attacks succeeded, the worm achieved communication with the operating system command interpreter.

***Recent Worm Attacks***

In late 2001, a more versatile worm appeared, known as Nimda. Nimda spreads by multiple mechanisms:

 from client to client via e-mail

 from client to client via open network shares

 from Web server to client via browsing of compromised Web sites

 from client to Web server via active scanning for and exploitation of various Microsoft

IIS 4.0 / 5.0 directory traversal vulnerabilities

 from client to Web server via scanning for the back doors left behind by the "Code Red

II" worms

The worm modifies Web documents (e.g., .htm, .html, and .asp files) and certain executable files found on the systems it infects and creates numerous copies of itself under various filenames.

In early 2003, the SQL Slammer worm appeared. This worm exploited a buffer overflow vulnerability in Microsoft SQL server.

Mydoom is a mass-mailing e-mail worm that appeared in 2004

**Antivirus Approaches**

**VIRUS COUNTERMEASURES**

The ideal solution to the threat of viruses is prevention: The next best approach is to be able to do the following:

 **Detection:** Once the infection has occurred, determine that it has occurred and locate the virus.

 **Identification**: Once detection has been achieved, identify the specific virus that has infected a program.

 **Removal**: Once the specific virus has been identified, remove all traces of the virus from the infected program and restore it to its original state. Remove the virus from all infected systems so that the disease cannot spread further.

If detection succeeds but either identification or removal is not possible, then the alternative is to discard the infected program and reload a clean backup version.

There are four generations of antivirus software:

 First generation: simple scanners

 Second generation: heuristic scanners

 Third generation: activity traps

 Fourth generation: full-featured protection

**A first-generation scanner** requires a virus signature to identify a virus.. Such signature- specific scanners are limited to the detection of known viruses. Another type of first-generation scanner maintains a record of the length of programs and looks for changes in length.

**A second-generation scanner** does not rely on a specific signature. Rather, the scanner uses heuristic rules to search for probable virus infection. One class of such scanners looks for fragments of code that are often associated with viruses.

Another second-generation approach is integrity checking. A checksum can be appended to each program. If a virus infects the program without changing the checksum, then an integrity check will catch the change. To counter a virus that is sophisticated enough to change the checksum when it infects a program, an encrypted hash function can be used. The encryption key is stored separately from the program so that the virus cannot generate a new hash code and encrypt that. By using a hash function rather than a simpler checksum, the virus is prevented from adjusting the program to produce the same hash code as before.

**Third-generation programs** are memory-resident programs that identify a virus by its actions rather than its structure in an infected program. Such programs have the advantage that it is not necessary to develop signatures and heuristics for a wide array of viruses. Rather, it is necessary only to identify the small set of actions that indicate an infection is being attempted and then to intervene.

**Fourth-generation products** are packages consisting of a variety of antivirus techniques used in conjunction. These include scanning and activity trap components. In addition, such a package includes access control capability, which limits the ability of viruses to penetrate a system and then limits the ability of a virus to update files in order to pass on the infection.

The arms race continues. With fourth-generation packages, a more comprehensive defense strategy is employed, broadening the scope of defense to more general-purpose computer security measures.

**Advanced Antivirus Techniques**

More sophisticated antivirus approaches and products continue to appear. In this subsection, we highlight two of the most important.

***Generic Decryption***

Generic decryption (GD) technology enables the antivirus program to easily detect even the most complex polymorphic viruses, while maintaining fast scanning speeds . In order to detect such a structure, executable files are run through a GD scanner, which contains the following elements:

 **CPU emulator:** A software-based virtual computer. Instructions in an executable file are interpreted by the emulator rather than executed on the underlying processor. The emulator includes software versions of all registers and other processor hardware, so that the underlying processor is unaffected by programs interpreted on the emulator.

 **Virus signature scanner:** A module that scans the target code looking for known virus signatures.

 **Emulation control module:** Controls the execution of the target code.

***Digital Immune System***

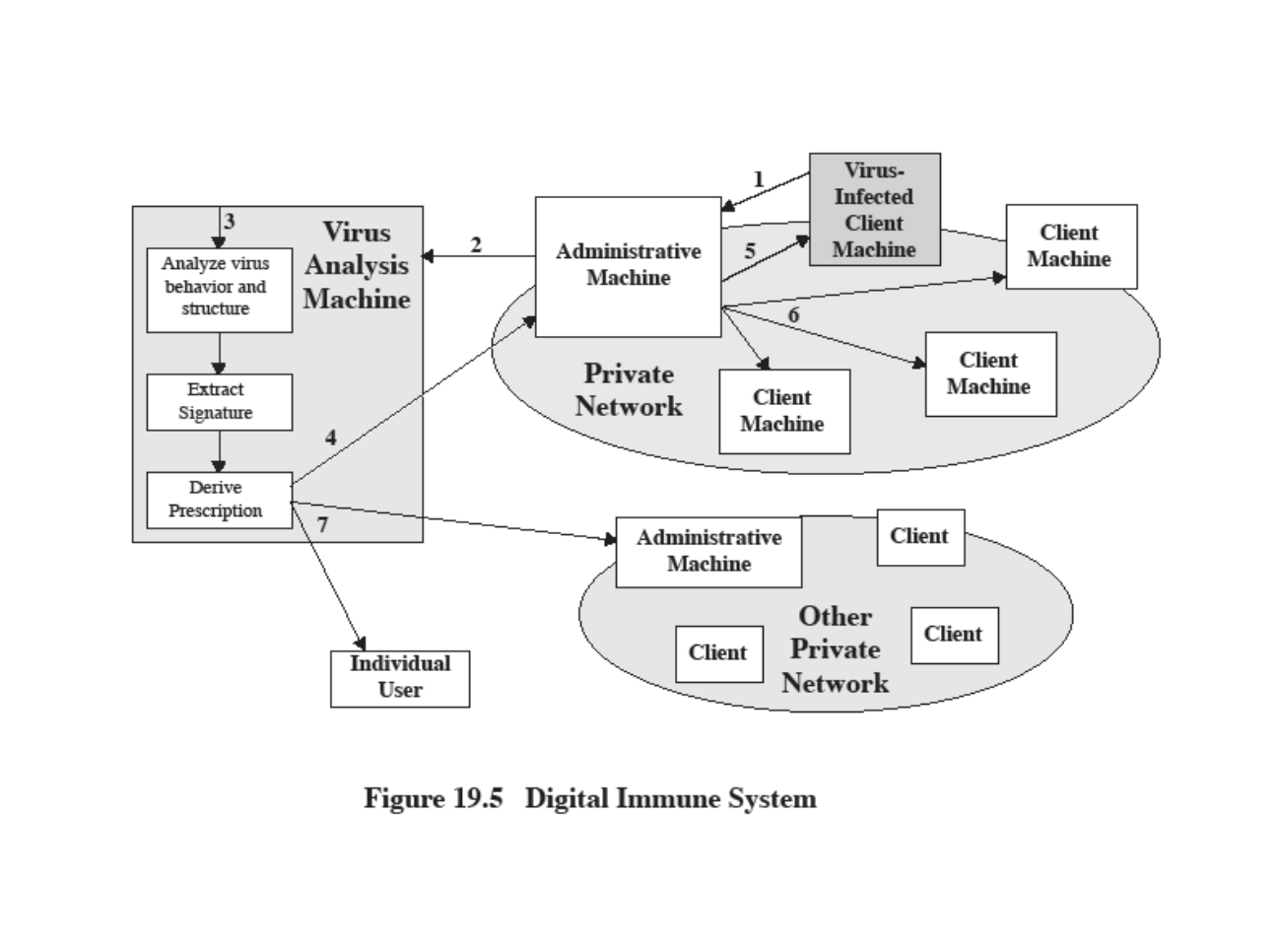
The digital immune system is a comprehensive approach to virus protection developed by IBM]. The motivation for this development has been the rising threat of Internet-based virus propagation.Two major trends in Internet technology have had an increasing impact on the rate of virus propagation in recent years:

 Integrated mail systems: Systems such as Lotus Notes and Microsoft Outlook make it very simple to send anything to anyone and to work with objects that are received.

 Mobile-program systems: Capabilities such as Java and ActiveX allow programs to move on their own from one system to another.

[Figure 19.4 i](mk:@MSITStore:C:\Documents%20and%20Settings\sethukarasi\Desktop\cryptography-and-network-security-4th-edition.9780131873162.25360.chm::/0131873164/ch19lev1sec2.html#ch19fig04)llustrates the typical steps in digital immune system operation:

**1.** A monitoring program on each PC uses a variety of heuristics based on system behavior, suspicious changes to programs, or family signature to infer that a virus may be present. The monitoring program forwards a copy of any program thought to be infected to an administrative machine within the organization.



**2.** The administrative machine encrypts the sample and sends it to a central virus analysis machine.

**3.** This machine creates an environment in which the infected program can be safely run for analysis. Techniques used for this purpose include emulation, or the creation of a protected environment within which the suspect program can be executed and monitored. The virus analysis machine then produces a prescription for identifying and removing the

virus.

**4.** The resulting prescription is sent back to the administrative machine.

**5.** The administrative machine forwards the prescription to the infected client.

**6.** The prescription is also forwarded to other clients in the organization.

**7.** Subscribers around the world receive regular antivirus updates that protect them from the new virus.

The success of the digital immune system depends on the ability of the virus analysis machine to detect new and innovative virus strains. By constantly analyzing and monitoring the viruses found in the wild, it should be possible to continually update the digital immune software to keep up with the threat.

**Behavior-Blocking Software**

Unlike heuristics or fingerprint-based scanners, behavior-blocking software integrates with the operating system of a host computer and monitors program behavior in real-time for malicious actions. Monitored behaviors can include the following:

 Attempts to open, view, delete, and/or modify files;

 Attempts to format disk drives and other unrecoverable disk operations;

 Modifications to the logic of executable files or macros;

 Modification of critical system settings, such as start-up settings;

 Scripting of e-mail and instant messaging clients to send executable content; and

 Initiation of network communications.

If the behavior blocker detects that a program is initiating would-be malicious behaviors as it runs, it can block these behaviors in real-time and/or terminate the offending software. This gives it a fundamental advantage over such established antivirus detection techniques as fingerprinting or heuristics.

**\* \* \* \* \* \* \* \* \* \* \* \* \* \* END OF UNIT V \* \* \* \* \* \* \* \* \* \* \* \* \* \***