SAMSKRUTI COLLEGE OF ENGINEERING & TECHNOLOGY
(Approved by AICTE, New Delhi & Affiliated to JNTUH.)
Kondapur (V), Ghatkesar (M), Medchal (Dist)

Course Hand Out

Subject Name: Cryptography and Network Security

Prepared by: G. Radha Devi, Assistant Professor, CSE

Year, Semester, Regulation: III Year- II Sem (R16)

UNIT-I

- **Computer Security** - generic name for the collection of tools designed to protect data and to thwart hackers
- **Network Security** - measures to protect data during their transmission
- **Internet Security** - measures to protect data during their transmission over a collection of interconnected networks

Security Attacks, Services and Mechanisms

To assess the security needs of an organization effectively, the manager responsible for security needs some systematic way of defining the requirements for security and characterization of approaches to satisfy those requirements. One approach is to consider three aspects of information security:

Security attack – Any action that compromises the security of information owned by an organization.

Security mechanism – A mechanism that is designed to detect, prevent or recover from a security attack.

Security service – A service that enhances the security of the data processing systems and the information transfers of an organization. The services are intended to counter security attacks and they make use of one or more security mechanisms to provide the service.

Basic Concepts

Cryptography The art or science encompassing the principles and methods of transforming an intelligible message into one that is unintelligible, and then retransforming that message back to its original form

Plaintext The original intelligible message

Cipher text The transformed message

Cipher An algorithm for transforming an intelligible message into one that is unintelligible by transposition and/or substitution methods

Key Some critical information used by the cipher, known only to the sender& receiver

Encipher (encode) The process of converting plaintext to cipher text using a cipher and a key

Decipher (decode) the process of converting cipher text back into plaintext using a cipher and a key

Cryptanalysis The study of principles and methods of transforming an unintelligible message back into an intelligible message *without* knowledge of the key. Also called **code breaking**

Cryptology Both cryptography and cryptanalysis

Code An algorithm for transforming an intelligible message into an unintelligible one using a code-book

Cryptography

Cryptographic systems are generally classified along 3 independent dimensions:

Type of operations used for transforming plain text to cipher text

All the encryption algorithms are based on two general principles: **substitution**, in which each element in the plaintext is mapped into another element, and **transposition**, in which elements in the plaintext are rearranged.

The number of keys used

If the sender and receiver uses same key then it is said to be **symmetric key (or) single key (or) conventional encryption**.

If the sender and receiver use different keys then it is said to be **public key encryption**.

The way in which the plain text is processed

A **block cipher** processes the input and block of elements at a time, producing output block for each input block.

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

Cryptanalysis

The process of attempting to discover X or K or both is known as cryptanalysis. The strategy used by the cryptanalysis depends on the nature of the encryption scheme and the information available to the cryptanalyst.

There are various types of cryptanalytic attacks based on the amount of information known to the cryptanalyst.

Cipher text only – A copy of cipher text alone is known to the cryptanalyst.

Known plaintext – The cryptanalyst has a copy of the cipher text and the corresponding plaintext.

Chosen plaintext – The cryptanalysts gains temporary access to the encryption machine. They cannot open it to find the key, however; they can encrypt a large number of suitably chosen plaintexts and try to use the resulting cipher texts to deduce the key.

Chosen cipher text – The cryptanalyst obtains temporary access to the decryption machine, uses it to decrypt several string of symbols, and tries to use the results to deduce the key.

STEGANOGRAPHY

A plaintext message may be hidden in any one of the two ways. The methods of steganography conceal the existence of the message, whereas the methods of cryptography render the message unintelligible to outsiders by various transformations of the text.

A simple form of steganography, but one that is time consuming to construct is one in which an arrangement of words or letters within an apparently innocuous text spells out the real message.

e.g., (i) the sequence of first letters of each word of the overall message spells out the real (Hidden) message.
(ii) Subset of the words of the overall message is used to convey the hidden message.

Various other techniques have been used historically, some of them are

Character marking – selected letters of printed or typewritten text are overwritten in pencil. The marks are ordinarily not visible unless the paper is held to an angle to bright light.

Invisible ink – a number of substances can be used for writing but leave no visible trace until heat or some chemical is applied to the paper.

Pin punctures – small pin punctures on selected letters are ordinarily not visible unless the paper is held in front of the light. Typewritten correction ribbon – used between the lines typed with a black ribbon, the results of typing with the correction tape are visible only under a strong light.

Drawbacks of steganography

Requires a lot of overhead to hide a relatively few bits of information.

Once the system is discovered, it becomes virtually worthless.

SECURITY SERVICES

The classification of security services are as follows:

Confidentiality: Ensures that the information in a computer system and transmitted information are accessible only for reading by authorized parties.

E.g. Printing, displaying and other forms of disclosure.

Authentication: Ensures that the origin of a message or electronic document is correctly identified, with an assurance that the identity is not false.

Integrity: Ensures that only authorized parties are able to modify computer system assets and transmitted information. Modification includes writing, changing status, deleting, creating and delaying or replaying of transmitted messages.

Non repudiation: Requires that neither the sender nor the receiver of a message be able to deny the transmission.

Access control: Requires that access to information resources may be controlled by or the target system.

Availability: Requires that computer system assets be available to authorized parties when needed.

SECURITY MECHANISMS

One of the most specific security mechanisms in use is cryptographic techniques. Encryption or encryption-like transformations of information are the most common means of providing security. Some of the mechanisms are

1 Encipherment
2 **Digital Signature**
3 **Access Control**

Cryptographic Attacks

Passive Attacks

Passive attacks are in the nature of eavesdropping on, or monitoring of, transmissions. The goal of the opponent is to obtain information that is being transmitted. Passive
attacks are of two types:

Release of message contents: A telephone conversation, an e-mail message and a transferred file may contain sensitive or confidential information. We would like to prevent the opponent from learning the contents of these transmissions.

Traffic analysis: If we had encryption protection in place, an opponent might still be able to observe the pattern of the message. The opponent could determine the location and identity of communication hosts and could observe the frequency and length of messages being exchanged. This information might be useful in guessing the nature of communication that was taking place.

Passive attacks are very difficult to detect because they do not involve any alteration of data. However, it is feasible to prevent the success of these attacks.

Active attacks

These attacks involve some modification of the data stream or the creation of a false stream. These attacks can be classified in to four categories:

Masquerade – One entity pretends to be a different entity.

Replay – involves passive capture of a data unit and its subsequent transmission to produce an unauthorized effect.

Modification of messages – Some portion of message is altered or the messages are delayed or recorded, to produce an unauthorized effect.

Denial of service – Prevents or inhibits the normal use or management of communication facilities. Another form of service denial is the disruption of an entire network, either by disabling the network or overloading it with messages so as to degrade performance.

It is quite difficult to prevent active attacks absolutely, because to do so would require physical protection of all communication facilities and paths at all times. Instead, the goal is to detect them and to recover from any disruption or delays caused by them.

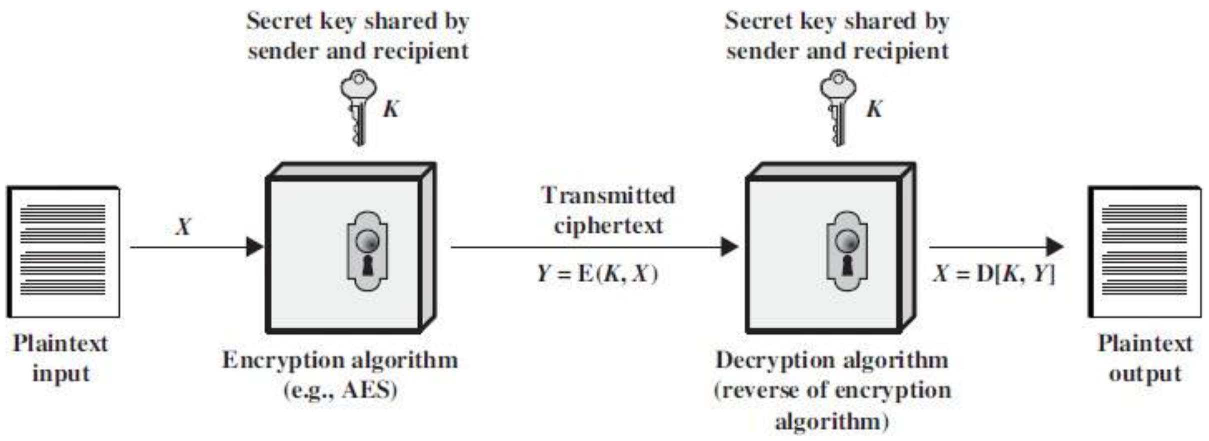
CONVENTIONAL ENCRYPTION

- Referred conventional / private-key / single-key
- Sender and recipient share a common key

All classical encryption algorithms are private-key was only type prior to invention of public-key in 1970‟**plaintext** - the original message
Some basic terminologies used:

- **cipher text** - the coded message
- **Cipher** - algorithm for transforming plaintext to cipher text
- **Key** - info used in cipher known only to sender/receiver
- **encipher (encrypt)** - converting plaintext to cipher text

- **decipher (decrypt)** - recovering cipher text from plaintext
- **Cryptography** - study of encryption principles/methods
- **Cryptanalysis (code breaking)** - the study of principles/ methods of deciphering cipher text *without* knowing key
- **Cryptology** - the field of both cryptography and cryptanalysis


Secret key shared by sender and recipient
K
Secret key shared by sender and recipient
K
X
Transmitted ciphertext
Y = E(K, X)
X = D[K, Y]
Plaintext input
Encryption algorithm (e.g., AES)
Decryption algorithm (reverse of encryption algorithm)
Plaintext output

CLASSICAL ENCRYPTION TECHNIQUES

There are two basic building blocks of all encryption techniques: substitution and transposition.

SUBSTITUTION TECHNIQUES

A substitution technique is one in which the letters of plaintext are replaced by other letters or by numbers or symbols. If the plaintext is viewed as a sequence of bits, then substitution involves replacing plaintext bit patterns with cipher text bit patterns.

TRANSPOSITION TECHNIQUES

All the techniques examined so far involve the substitution of a cipher text symbol for a plaintext symbol. A very different kind of mapping is achieved by performing some sort of permutation on the plaintext letters. This technique is referred to as a transposition cipher.

IMPORTANT QUESTIONS:

1. What is the need for security?
2. What are the types of security attacks?
3. What are the various security mechanisms?
4. Explain network security model?
5. Compare substitution ciphers with transposition ciphers?
6. Explain various substitution techniques with suitable examples?
7. Explain the Caesar cipher?
8. Explain Hill cipher with example?

UNIT - II

BLOCK CIPHER PRINCIPLES

Virtually, all symmetric block encryption algorithms in current use are based on a structure referred to as Fiestel block cipher. For that reason, it is important to examine the design principles of the Fiestel cipher. We begin with a **comparison of stream cipher with block cipher.**

- **A stream cipher** is one that encrypts a digital data stream one bit or one byte at a time. E.g, vigenere cipher. **A block cipher** is one in which a block of plaintext is treated as a whole and used to produce a cipher text block of equal length. Typically a block size of 64 or 128 bits is used.

Block cipher principles

- most symmetric block ciphers are based on a **Feistel Cipher Structure** needed since must be able to **decrypt** ciphertext to recover messages efficiently. block ciphers look like an extremely large substitution
- would need table of 264 entries for a 64-bit block
- Instead create from smaller building blocks
- using idea of a product cipher in 1949 Claude Shannon introduced idea of substitu- tion-permutation (S-P) networks called modern substitution-transposition product cipher
these form the basis of modern block ciphers

- S-P networks are based on the two primitive cryptographic operations we have seen before:
  - *substitution* (S-box)
  - *permutation* (P-box)
  - provide *confusion* and *diffusion* of message
  - **diffusion** – dissipates statistical structure of plaintext over bulk of ciphertext

- **confusion** – makes relationship between ciphertext and key as complex as possible

DATA ENCRYPTION STANDARD (DES)

In May 1973, and again in Aug 1974 the NBS (now NIST) called for possible encryption algorithms for use in unclassified government applications response was mostly disappointing, however IBM submitted their Lucifer design following a period of redesign and comment it became the Data Encryption Standard (DES)

it was adopted as a (US) federal standard in Nov 76, published by NBS as a hardware only scheme in Jan 77 and by ANSI for both hardware and software standards in ANSI X3.92-1981 (also X3.106-1983 modes of use) subsequently it has been widely adopted and is now published in many standards around the world cf Australian Standard AS2805.5-1985one of the largest users of the DES is the banking industry, particularly with EFT, and EFTPOS

it is for this use that the DES has primarily been standardized, with ANSI having twice reconfirmed its recommended use for 5 year periods - a further extension is not expected however although the standard is public, the design criteria used are classified and have yet to be released there has been considerable controversy over the design, particularly in the choice of a 56-bit key

- recent analysis has shown despite this that the choice was appropriate, and that DES is well designed
- rapid advances in computing speed though have rendered the 56 bit key susceptible to exhaustive key search, as predicted by Diffie & Hellman
- the DES has also been theoretically broken using a method called Differential Cryptanalysis, however in practice this is unlikely to be a problem (yet)

DES Modes of Use

- DES encrypts 64-bit blocks of data, using a 56-bit key
- we need some way of specifying how to use it in practise, given that we usually have an arbitrary amount of information to encrypt
- the way we use a block cipher is called its **Mode of Use** and four have been defined for the DES by ANSI in the standard: ANSI X3.106-1983 Modes of Use)
- modes are either:

Block Modes

Splits messages in blocks (ECB, CBC)

Electronic Codebook Book (ECB)

- Where the message is broken into independent 64-bit blocks which are encrypted

C_(i) = DES_(K1) (P_(i))

Cipher Block Chaining (CBC)

Again the message is broken into 64-bit blocks, but they are linked together in the encryption operation with an IV C_(i) = DES_(K1) (P_(i)(+)C_(i-1)) C_(-1)=IV

Stream Modes

On bit stream messages (CFB, OFB)

Cipher Feedback (CFB)

- Where the message is treated as a stream of bits, added to the output of the DES, with the result being feedback for the next stage
C_(i) = P_(i)(+) DES_(K1) (C_(i-1)) C_(-1)=IV

Output Feedback (OFB)

- Where the message is treated as a stream of bits, added to the message, but with the feedback being independent of the message
C_(i) = P_(i)(+) O_(i) O_(i) = DES_(K1)(O_(i-1)) O_(-1)=IV

- each mode has its advantages and disadvantages

Limitations of Various Modes

ECB

- repetitions in message can be reflected in ciphertext
  - if aligned with message block
  - particularly with data such graphics
  - or with messages that change very little, which become a code-book analysis problem
- weakness is because enciphered message blocks are independent of each other

CBC

- use result of one encryption to modify input of next
  - hence each ciphertext block is dependent on **all** message blocks before it
  - thus a change in the message affects the ciphertext block after the change as well as the original block

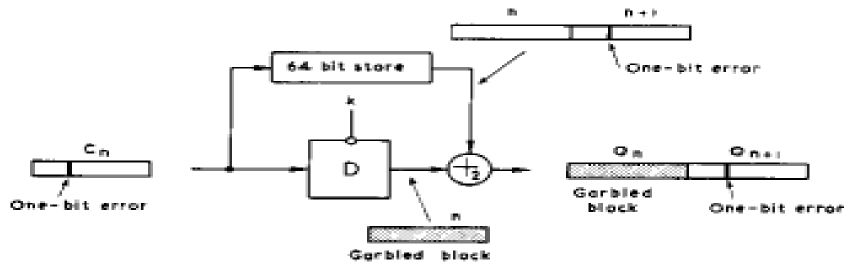


Fig. 4.5 One-bit error in cipher block chaining

to start need an **Initial Value** (IV) which must be known by both sender and receiver

- however if IV is sent in the clear, an attacker can change bits of the first block, and change IV to compensate
- hence either IV must be a fixed value (as in EFTPOS) or it must be sent encrypted in ECB mode before rest of message

- also at the end of the message, have to handle a possible last short block
  - either pad last block (possible with count of pad size), or use some fiddling to double up last two blocks
  - see Davies for examples

CFB

- when data is bit or byte oriented, want to operate on it at that level, so use a stream mode
- the block cipher is use in **encryption** mode at **both** ends, with input being a feed-back copy of the ciphertext
- can vary the number of bits feed back, trading off efficiency for ease of use
- again errors propogate for several blocks after the erro

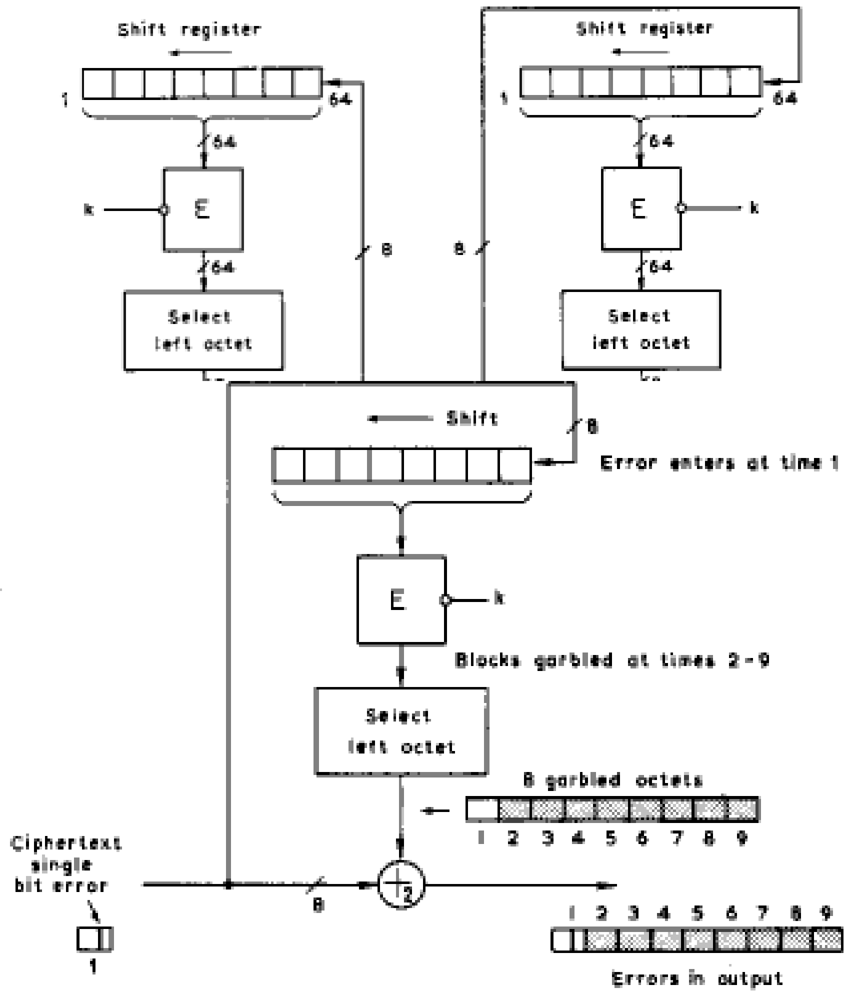


Fig. 4.8 One-bit error in 8-bit cipher feedback

OFB

- also a stream mode, but intended for use where the error feedback is a problem, or where the encryptions want to be done before the message is available
- is superficially similar to CFB, but the feedback is from the output of the block cipher and is independent of the message, a variation of a Vernam cipher
- again an IV is needed
- sender and receiver must remain in sync, and some recovery method is needed to ensure this occurs
- although originally specified with varying m-bit feedback in the standards, subsequent research has shown that only **64-bit OFB** should ever be used (and this is the most efficient use anyway), see

  D Davies, G Parkin, "The Average Cycle Size of the Key Stream in Output Feedback Encipherment" in Advances in Cryptology - Crypto 82, Plenum Press, 1982, pp97-98

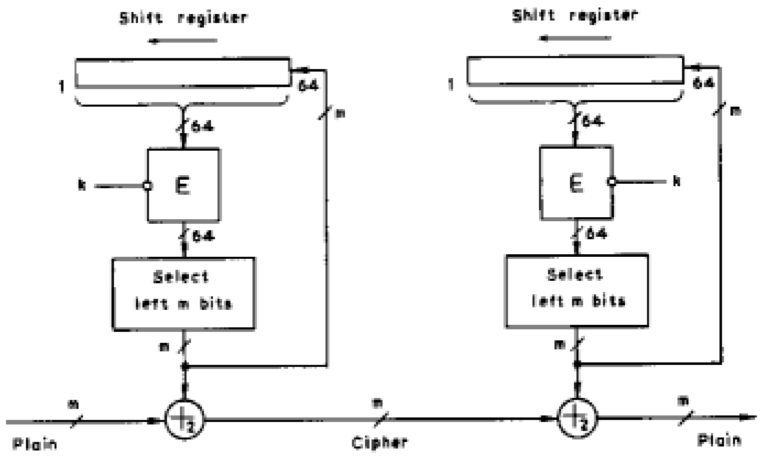


Fig. 4.12 m-bit output feedback

DES Weak Keys

- with many block ciphers there are some keys that should be avoided, because of reduced cipher complexity
- these keys are such that the same sub-key is generated in more than one round, and they include:

Weak Keys

- he same sub-key is generated for every round
- DES has 4 weak keys

Semi-Weak Keys

- only two sub-keys are generated on alternate rounds
- DES has 12 of these (in 6 pairs)

Demi-Semi Weak Keys

- have four sub-keys generated
- none of these cause a problem since they are a tiny fraction of all available keys
- however they MUST be avoided by any key generation program

Triple DES

- DES variant
- standardised in ANSI X9.17 & ISO 8732 and in PEM for key management
- proposed for general EFT standard by ANSI X9
- backwards compatible with many DES schemes
- uses 2 or 3 keys

  C = DES_(K1) Bbc{(DES^(-1)_(K2)Bbc{(DES_(K1)(P)))
- no known practical attacks
  - brute force search impossible
  - meet-in-the-middle attacks need 2^(56) PC pairs per key
- popular current alternative

IDEA (IPES)

- developed by James Massey & Xuejia Lai at ETH originally in Zurich in 1990, then called IPES :
- Name changed to IDEA in 1992
- encrypts 64-bit blocks using a 128-bit key
- based on mixing operations from different (incompatible) algebraic groups (XOR, Addition mod 2^(16) , Multiplication mod 2^(16) +1)
- all operations are on 16-bit sub-blocks, with no permutations used, hence its very efficient in s/w
- IDEA is patented in Europe & US, however non-commercial use is freely permitted
- used in the public domain PGP secure email system (with agreement from the patent holders)
- currently no attack against IDEA is known (it appears secure against differential cryptanalysis), and its key is too long for exhaustive search

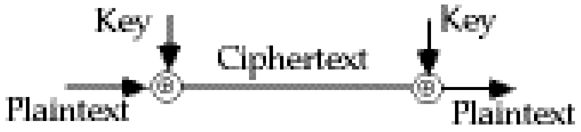
Overview of IDEA

- IDEA encryption works as follows:

- the 64-bit data block is divided by 4 into: X_(1) , X_(2) , X_(3) , X_(4)
- in each of eight the sub-blocks are XORd, added, multiplied with one another and with six 16-bit sub-blocks of key material, and the second and third sub-blocks are swapped
- finally some more key material is combined with the sub-blocks

Stream Ciphers and the Vernam cipher

- Process the message bit by bit (as a stream)
- The most famous of these is the **Vernam cipher** (also known as the **one-time pad**)
- invented by Vernam, working for AT&T, in 1917
- simply add bits of message to random key bits
- need as many key bits as message, difficult in practise (ie distribute on a mag-tape or CDROM)
- is unconditionally secure provided key is truly random



- suggest generating keystream from a smaller (base) key

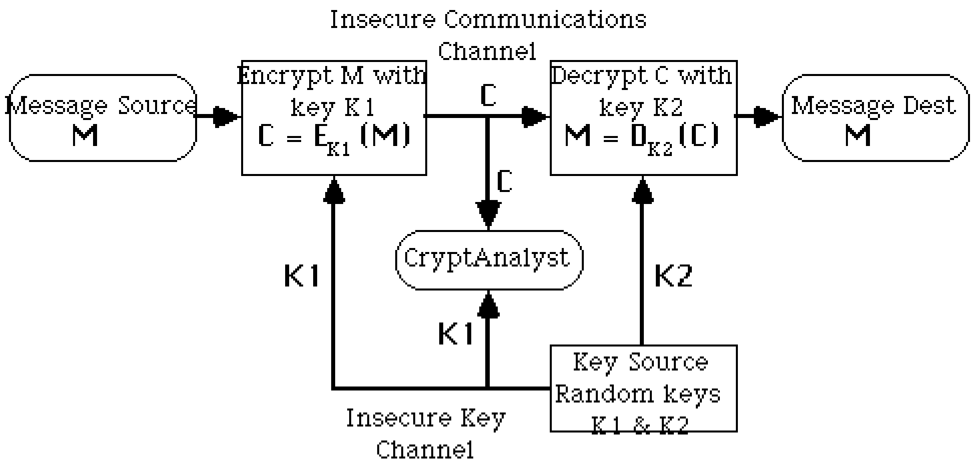
<u>IMPORTANT QUESTIONS:</u>

1. Explain block cipher design principles.
2. Write about DES Algorithm.
3. Explain block cipher modes of operation.
4. What is AES Algorithm?
5. Explain Blowfish algorithm.
6. What are the principles of public key cryptosystems.

UNIT - III

Public-Key Ciphers

- traditional **secret key** cryptography uses a single key shared by both sender and receiver
- if this key is disclosed communications are compromised
- also does not protect sender from receiver forging a message & claiming is sent by sender, parties are equal
- **public-key** (or **two-key**) **cryptography** involves the use of two keys:
  - a **public-key**, which may be known by anybody, and can be used to **encrypt messages**, and **verify signatures**
  - a **private-key**, known only to the recipient, used to **decrypt messages**, and **sign** (create) **signatures**


Insecure Communications Channel
Message Source M
Encrypt M with key K1 $C = E_{K1}(M)$
C
Decrypt C with key K2 $M = D_{K2}(C)$
Message Dest M
C
CryptAnalyst
K1
K2
K1
Key Source Random keys K1 & K2
Insecure Key Channel

Asymmetric (Public-Key) Encryption System

- the public-key is easily computed from the private key and other information about the cipher (a polynomial time (P-time) problem)
- however, knowing the public-key and public description of the cipher, it is still computationally infeasible to compute the private key (an NP-time problem)
- thus the public-key may be distributed to anyone wishing to communicate securly with its owner (although secure distribution of the public-key is a non-trivial problem - the **key distribution** problem)

have three important classes of public-key algorithms

Public-Key Distribution Schemes (PKDS) - where the scheme is used to securely exchange a single piece of information (whose value depends on the two parties, but cannot be set).

- This value is normally used as a session key for a private-key scheme
- **Signature Schemes** - used to create a digital signature only, where the private-key signs (create) signatures, and the public-key verifies signatures
- **Public Key Schemes (PKS)** - used for encryption, where the public-key encrypts

messages, and the private-key decrypts messages.

- Any public-key scheme can be used as a PKDS, just by selecting a message which is the required session key
- Many public-key schemes are also signature schemes (provided encryption& decryption can be done in either order)

RSA Public-Key Cryptosystem

- best known and widely regarded as most practical public-key scheme was proposed by Rivest, Shamir & Adleman in 1977:

R L Rivest, A Shamir, L Adleman, "On Digital Signatures and Public Key Cryptosystems", Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978

- it is a public-key scheme which may be used for encrypting messages, exchanging keys, and creating digital signatures
- is based on exponentiation in a finite (Galois) field over integers modulo a prime
  - nb exponentiation takes $O((\log n)^3)$ operations
- its security relies on the difficulty of calculating factors of large numbers
  - nb factorization takes $O(e^{\log n \log \log n})$ operations
  - (same as for discrete logarithms)
- the algorithm is patented in North America (although algorithms cannot be patented elsewhere in the world)
  - this is a source of legal difficulties in using the scheme

- RSA is a public key encryption algorithm based on exponentiation using modular arithmetic
- to use the scheme, first generate keys:
- Key-Generation by each user consists of:
  - selecting two large primes at random (~100 digit), p, q
  - calculating the system modulus R=p.q p, q primes
  - selecting at random the encryption key e,
  - e < R, gcd(e, F(R)) = 1
  - solving the congruence to find the decryption key d,
  - e.d [[equivalence]] 1 mod [[phi]](R) 0 <= d <= R
  - publishing the public encryption key: K1={e,R}
  - securing the private decryption key: K2={d,p,q}
- Encryption of a message M to obtain ciphertext C is:
- $C = M^e \bmod R$ 0 <= d <= R

- Decryption of a ciphertext C to recover the message M is:
  - $M = C^d = M^{e.d} = M^{1+n.[[phi]](R)} = M \bmod R$
- the RSA system is based on the following result:

if R = pq where p, q are distinct large primes then
X [[phi]](R) = 1 mod R
for all x not divisible by p or q
and [[Phi]](R) = (p-1)(q-1)

RSA Example

- usually the encryption key e is a small number, which must be relatively prime to [[phi]](R) (ie GCD(e, [[phi]](R)) = 1)
- typically e may be the same for all users (provided certain precautions are taken), 3 is suggested
- the decryption key d is found by solving the congruence:

e.d [[equivalence]] 1 mod [[phi]](R), 0 <= d <= R,

- an extended Euclid's GCD or Binary GCD calculation is done to do this.

```
given e=3, R=11*47=517, [[phi]](R)=10*46=460
then d=Inverse(3,460) by Euclid's alg:
 i   y   g   u   v
 0   -  460  1   0
 1   -   3   0   1
 2  153   1   1 -153
 3   3   0 -3  460
ie:        d = -153, or 307 mod 517
```

- a sample RSA encryption/decryption calculation is:

```
M = 26
C = 263 mod 517 = 515
M = 515307 mod 517 = 26
```

AUTHENTICATION REQUIREMENTS

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

Disclosure – releases of message contents to any person or process not possessing the appropriate cryptographic key.

Traffic analysis – discovery of the pattern of traffic between parties.

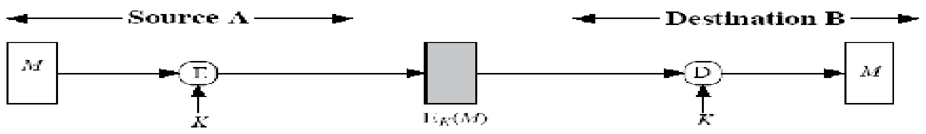
Masquerade – insertion of messages into the network fraudulent source.

Content modification – changes to the content of the message, including insertion deletion, transposition and modification.

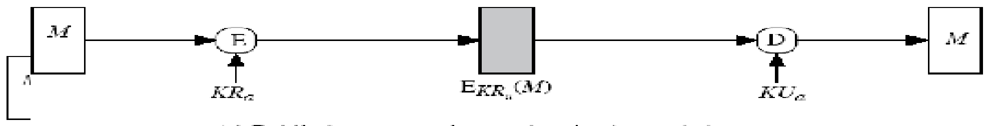
Sequence modification – any modification to a sequence of messages between parties, including insertion, deletion and reordering.

Timing modification – delay or replay of messages.

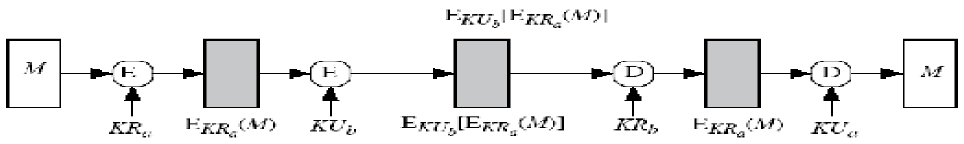
Source repudiation – denial of transmission of message by source.



(a) Symmetric encryption: confidentiality and authentication



(c) Public-key encryption: authentication and signature



(d) Public-key encryption: confidentiality, authentication, and signature

Destination repudiation – denial of transmission of message by destination.

easures to deal with first two attacks are in the realm of message confidentiality. Measures to deal with 3 through 6 are regarded as message authentication. Item 7 comes under digital signature and dealing with item 8 may require a combination of digital signature and a protocol to counter this attack.

AUTHENTICATION FUNCTIONS

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

The different types of functions that may be used to produce an **authenticator** are as follows:

Message encryption – the cipher text of the entire message serves as its authenticator.

Message authentication code (MAC) – a public function of the message and a secret key that produces a fixed length value serves as the authenticator.

Hash function – a public function that maps a message of any length into a fixed length hash value, which serves as the authenticator.

Message encryption

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

MESSAGE AUTHENTICATION CODE (MAC)

An alternative authentication technique involves the use of secret key to generate a small fixed size block of data, known as cryptographic checksum or MAC that is appended to the message. This technique assumes that two communication parties say A and B, share a common secret key ‘k’. When A has to send a message to B, it calculates the MAC as a function of the message and the key.

MAC = $C_K(M)$ Where M – input message

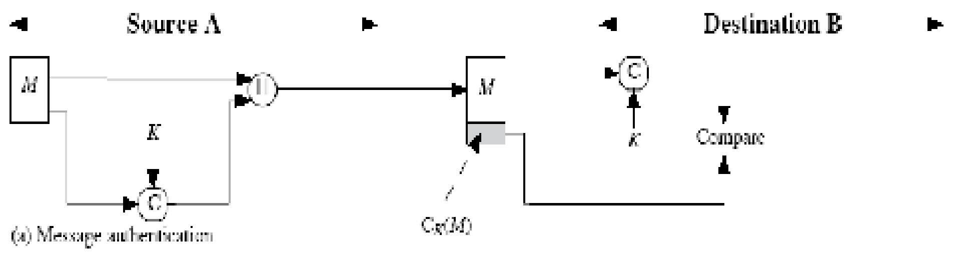
C – MAC function

K – Shared secret key

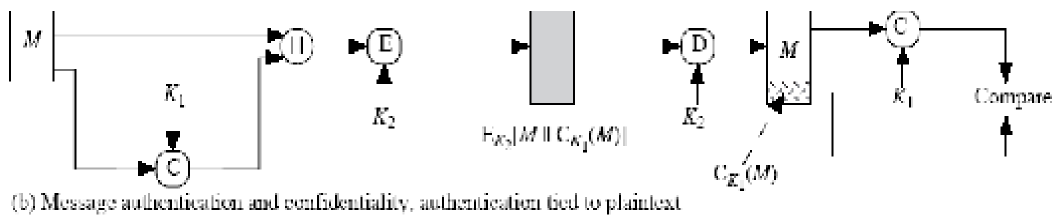
+MAC - Message Authentication Code

The message plus MAC are transmitted to the intended recipient. The recipient performs the same calculation on the received message, using the shared secret key, to generate a new MAC. The received MAC is compared to the calculated MAC. If it is equal, then the message is considered authentic.

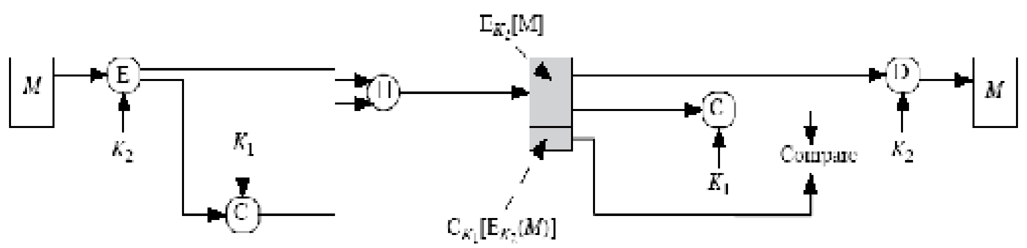
A MAC function is similar to encryption. One difference is that MAC algorithm need not be reversible, as it must for decryption. In general, the MAC function is a many- to-one function.



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

Requirements for MAC:

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 the case of a MAC, the considerations are entirely different. 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 M_1 and MAC_1, with $MAC_1 = CK (M_1)$, the cryptanalyst can perform $MAC_i = CK_i (M_1)$ for all possible key values K_i.

At least one key is guaranteed to produce a match of $MAC_i = MAC_1$.

Note that a total of 2^k MACs will be produced, but there are only $2^n < 2^k$ 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 $2^k/2^n = 2^{(k-n)}$ keys will produce a match. Thus, the opponent must iterate the attack:

Round 1

Given: M_1, $MAC_1 = CK(M_1)$

Compute $MAC_i = CK_i (M_1)$ for all 2^k keys

Number of matches $\approx 2^{(k-n)}$

Round 2

Given: M_2, $MAC_2 = CK(M_2)$

Compute $MAC_i = CK_i (M_2)$ for the $2^{(k-n)}$ keys resulting from Round 1

Number of matches $\approx 2^{(k-2xn)}$

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 2^{48} possible keys. The second round will narrow the possible keys to about 2^{16} 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.

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 = ($X_1||X_2||...||X_m$) be a message that is treated as a concatenation of 64-bit blocks X_i. Then define

$\Delta(M) = X1 \oplus X2 \oplus \dots \oplus Xm$

$Ck(M)=Ek(\Delta(M)$

where $\oplus$ 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 2^{56} encryptions. But the opponent can attack the system by replacing X_1 through

X_{m-1} with any desired values Y_1 through Y_{m-1} and replacing X_m with Y_m where Y_m is calculated as follows:

$Y_m = Y_1 \oplus Y_2 \oplus \dots \oplus Y_{m1} \oplus \Delta(M)$

The opponent can now concatenate the new message, which consists of Y_1 through Y_m, 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.

Then the MAC function should satisfy the following requirements: The MAC function should have

the following properties:

If an opponent observes M and $C_K(M)$, it should be computationally infeasible for

the opponent to construct a message M' such that $C_K(M') = C_K(M)$

$C_K(M)$ 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 2^{-n} where n is the number of bits in the MAC.

Let M' be equal to some known transformation on M. i.e., M' = f(M).

MAC based on DES

One of the most widely used MACs, referred to as Data Authentication Algorithm

(DAA) is based on DES.

The algorithm can be defined as using cipher block chaining (CBC) mode of operation of DES with an initialization vector of zero. The data to be authenticated are grouped into contiguous 64-bit blocks: D_1, D_2 … D_n. if necessary, the final block is padded on the right with zeros to form a full 64-bit block. Using the DES encryption algorithm and a secret key, a data authentication code

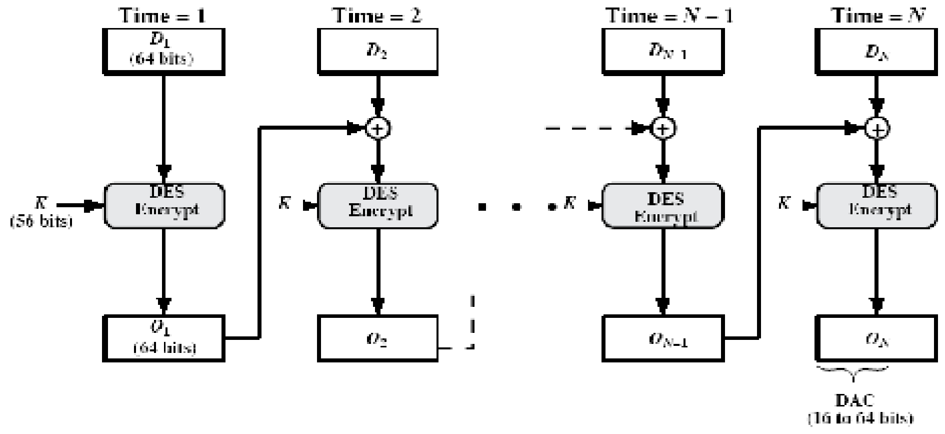
(DAC) is calculated as follows:

$O_1 = E_K(D_1)$

$O_2 = E_K(D_2 \oplus O_1)$

$O_3 = E_K(D_3 \oplus O_2)$ …

$O_N = E_K(D_N \oplus O_{N-1})$

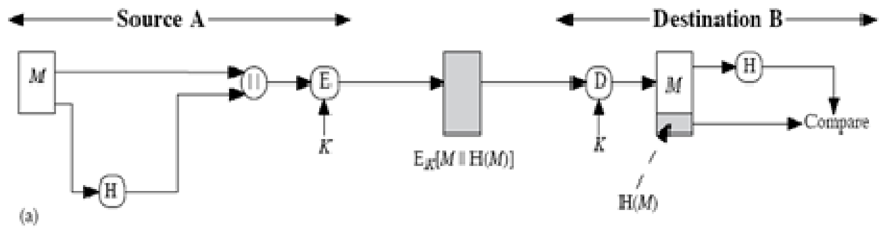


HASH FUNCTIONS

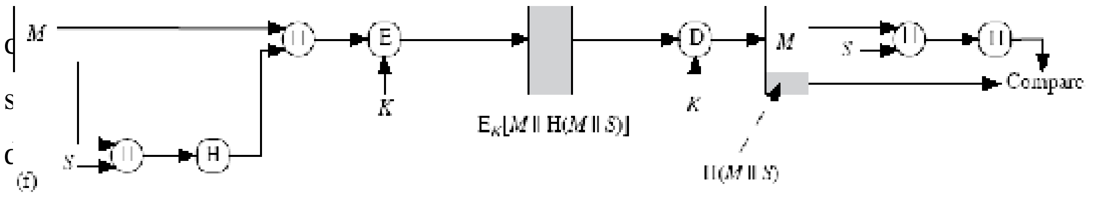
A variation on the message authentication code is the one way hash function. As with MAC, a hash function accepts a variable size message M as input and produces a fixed-size output, referred to as hash code H(M). Unlike a MAC, a hash code does not use a key but is a function only of the input message. The hash code is also referred to as a message digest or hash value.

There are varieties of ways in which a hash code can be used to provide message authentication, as follows:

a) The message plus the hash code is encrypted using symmetric encryption. This is identical to that of internal error control strategy. Because encryption is applied to the entire message plus the hash code, confidentiality is also provided.



b) Only the hash code is encrypted, using symmetric encryption. This reduces the processing burden for those applications that do not require confidentiality.



c
s
c the public key
e

Figure 11.5 Basic Uses of Hash Function (page 2 of 2)

e) This technique uses a hash function, but no encryption for message authentication. This technique assumes that the two communicating parties share a common secret value ‘S’. The source computes the hash value over the concatenation of M and S and appends the resulting hash value to M.

f) Confidentiality can be added to the previous approach by encrypting the entire message plus the hash code.

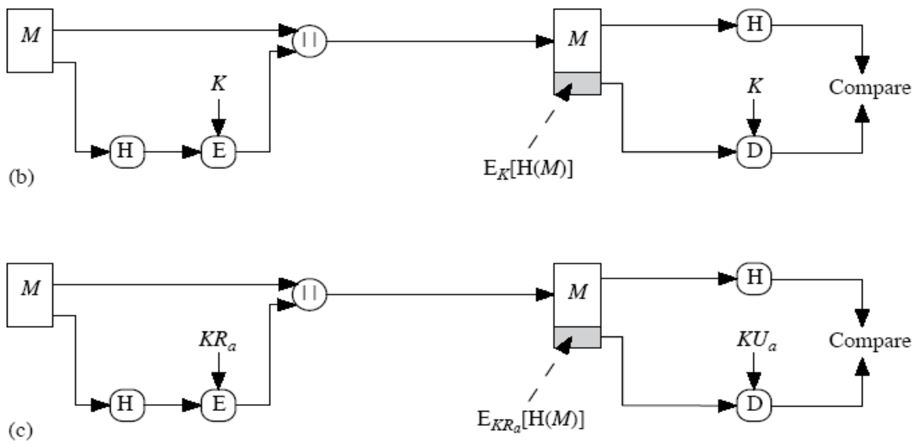


Figure 11.5 Basic Uses of Hash Function (page 1 of 2)

A hash value h is generated by a function H of the form h = H(M)

Where M is a variable-length message and H(M) is the fixed-length hash value. The hash value is appended to the message at the source at a time when the message is assumed or known to be correct. The receiver authenticates that message by re-computing the hash value.

Cryptanalysis

As with encryption algorithms, cryptanalytic attacks on hash functions and MAC algorithms seek to exploit some property of the algorithm to perform some attack other than an exhaustive search.

Hash Functions

In recent years, there has been considerable effort, and some successes, in developing cryptanalytic attacks on hash functions. To understand these, we need to look at the overall structure of a typical secure hash function, and is the structure of most hash functions in use today, including SHA and Whirlpool.

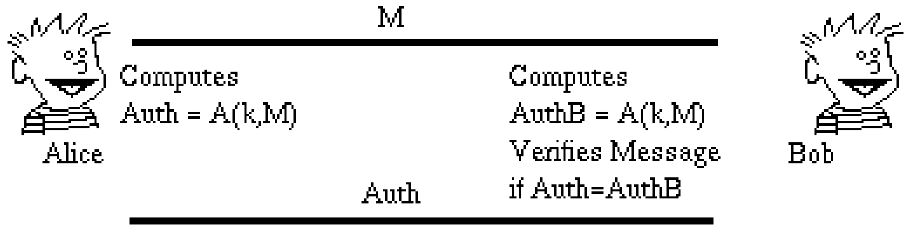
The hash function takes an input message and partitions it into L fixed-sized blocks of b bits each. If necessary, the final block is padded to b bits.

The final block also includes the value of the total length of the input to the hash function.The inclusion of the length makes the job of the opponent more difficult.

Either the opponent must find two messages of equal length that hash to the same value or two messages of differing lengths that, together with their length values, hash to the same value.

Message Authentication.

- Message authentication is concerned with:
  - protecting the integrity of a message
  - validating identity of originator
  - non-repudiation of origin (dispute resolution)
- electronic equivalent of a signature on a message
- An **authenticator**, **signature**, or **message authentication code (MAC)** is sent along with the message
- The MAC is generated via some algorithm which depends on both the message and some (public or private) key known only to the sender and receiver
- The message may be of any length
- the MAC may be of any length, but more often is some fixed size, requiring the use of some **hash function** to condense the message to the required size if this is not acheived by the authentication scheme
- need to consider replay problems with message and MAC
  - require a message sequence number, timestamp or negotiated random values



Authentication using Private-key Ciphers

- if a message is being encrypted using a session key known only to the sender and receiver, then the message may also be authenticated
    - since only sender or receiver could have created it
    - any interference will corrupt the message (provided it includes sufficient redundancy to detect change)
    - but this does not provide non-repudiation since it is impossible to prove who created the message
- message authentication may also be done using the standard modes of use of a block cipher
    - sometimes do not want to send encrypted messages
    - can use either CBC or CFB modes and send final block, since this will depend on all previous bits of the message
    - no hash function is required, since this method accepts arbitrary length input and produces a fixed output
    - usually use a fixed known IV
    - this is the approached used in Australian EFT standards AS8205
    - major disadvantage is small size of resulting MAC since 64-bits is probably too small

Hashing Functions

- hashing functions are used to condense an arbitrary length message to a fixed size, usually for subsequent signature by a digital signature algorithm
- good cryptographic hash function h should have the following properties:
    - h should destroy all holomorphic structures in the underlying public key cryptosystem (be unable to compute hash value of 2 messages combined given their individual hash values)
    - h should be computed on the entire message
    - h should be a one-way function so that messages are not disclosed by their signatures
    - it should be computationally infeasible given a message and its hash value to compute another message with the same hash value

- should resist **birthday attacks** (finding any 2 messages with the same hash value, perhaps by iterating through minor permutations of 2 messages)

- it is usually assumed that the hash function is public and not keyed
- traditional CRCs do not satisfy the above requirements
- length should be large enough to resist birthday attacks (64-bits is now regarded as too small, 128-512 proposed)

MD2, MD4 and MD5

- family of one-way hash functions by Ronald Rivest
- MD2 is the oldest, produces a 128-bit hash value, and is regarded as slower and less secure than MD4 and MD5
- MD4 produces a 128-bit hash of the message, using bit operations on 32-bit operands for fast implementation

R L Rivest, "The MD4 Message Digest Algorithm", Advances in Cryptology - Crypto'90, Lecture Notes in Computer Science No 537, Springer-Verlag 1991, pp303-311

- MD4 overview
  - pad message so its length is 448 mod 512
  - append a 64-bit message length value to message
  - initialise the 4-word (128-bit) buffer (A,B,C,D)
  - process the message in 16-word (512-bit) chunks, using 3 rounds of 16 bit operations each on the chunk & buffer
  - output hash value is the final buffer value
- some progress at cryptanalysing MD4 has been made, with a small number of collisions having been found
- MD5 was designed as a strengthened version, using four rounds, a little more complex than in MD4
- a little progress at cryptanalysing MD5 has been made with a small number of collisions having been found
- both MD4 and MD5 are still in use and considered secure in most practical applications

- both are specified as Internet standards (MD4 in RFC1320, MD5 in RFC1321)

3.3.1 SHA (Secure Hash Algorithm)

- SHA was designed by NIST & NSA and is the US federal standard for use with the DSA signature scheme (nb the algorithm is SHA, the standard is SHS)
- it produces 160-bit hash values
- SHA overview
  - pad message so its length is a multiple of 512 bits
  - initialise the 5-word (160-bit) buffer (A,B,C,D,E) to
  - (67452301,efcdab89,98badcfe,10325476,c3d2e1f0)
  - process the message in 16-word (512-bit) chunks, using 4 rounds of 20 bit operations each on the chunk & buffer
  - output hash value is the final buffer value
- SHA is a close relative of MD5, sharing much common design, but each having differences
- SHA has very recently been subject to modification following NIST identification of some concerns, the exact nature of which is not public
- current version is regarded as secure

Digital Signature Schemes

- public key signature schemes
- the private-key signs (creates) signatures, and the public-key verifies signatures
- only the owner (of the private-key) can create the digital signature, hence it can be used to verify who created a message
- anyone knowing the public key can verify the signature (provided they are confident of the identity of the owner of the public key - the key distribution problem)
- usually don't sign the whole message (doubling the size of information exchanged), but just a **hash** of the message

- digital signatures can provide non-repudiation of message origin, since an asymmetric algorithm is used in their creation, provided suitable timestamps and redundancies are incorporated in the signature

RSA

- RSA encryption and decryption are commutative, hence it may be used directly as a digital signature scheme
  - given an RSA scheme {(e,R), (d,p,q)}
- to **sign** a message, compute:
  - $S = M^{d}(\text{mod } R)$
- to **verify** a signature, compute:
  - $M = S^{e}(\text{mod } R) = M^{e.d}(\text{mod } R) = M(\text{mod } R)$
- thus know the message was signed by the owner of the public-key
- would seem obvious that a message may be encrypted, then signed using RSA without increasing it size
  - but have blocking problem, since it is encrypted using the receivers modulus, but signed using the senders modulus (which may be smaller)
  - several approaches possible to overcome this
- more commonly use a hash function to create a separate MDC which is then signed

El Gamal Signature Scheme

- whilst the ElGamal encryption algorithm is not commutative, a closely related signature scheme exists
- El Gamal Signature scheme
- given prime p, public random number g, private (key) random number x, compute
  - $y = g^{x}(\text{mod } p)$
- public key is (y,g,p)

- nb (g,p) may be shared by many users
- p must be large enough so discrete log is hard

- private key is (x)
- to **sign** a message M
    - choose a random number k, GCD(k,p-1)=1
    - compute a = g^k(mod p)
    - use extended Euclidean (inverse) algorithm to solve
    - M = x.a + k.b (mod p-1)
    - the signature is (a,b), k must be kept secret
    - (like ElGamal encryption is double the message size)
- to **verify** a signature (a,b) confirm:
    - $y^a.a^b$(mod p) = g^M(mod p)

Example of ElGamal Signature Scheme

- given p=11, g=2
- choose private key x=8
- compute
    - y = g^x(mod p) = 2^8(mod 11) = 3
- public key is y=3,g=2,p=11)
- to sign a message M=5
    - choose random k=9
    - confirm gcd(10,9)=1
    - compute

← a = g^k(mod p) = 2^9(mod 11) = 6

- solve
    - M = x.a+k.b(mod p-1)
    - 5 = 8.6+9.b(mod 10)
    - giving b = 3
- signature is (a=6,b=3)

- to verify the signature, confirm the following are correct:
    - $y^{a} \cdot a^{b}$(mod p) = g^{M}(mod p)
    - $3^{6} \cdot 6^{3}$(mod 11) = 2^{5}(mod 11)

IMPORTANT QUESTIONS:

1.What is secure hash algorithm?

2.Discuss about SHA-512.

3. Define message authentication code.

4.Discuss HMAC and CMAC.

5.What is digital signature?

UNIT - IV

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:

- **Secure:** A network eavesdropper should not be able to obtain the necessary information to impersonate a user. More generally, Kerberos should be strong enough that a potential opponent does not find it to be the weak link.
- **Reliable:** For all services that rely on Kerberos for access control, lack of availability 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.
- **Transparent:** Ideally, the user should not be aware that authentication is taking place, beyond the requirement to enter a password.
- **Scalable:** The system should be capable of supporting large numbers of clients and 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.

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:

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:

a. C >> AS: IDc||IDtgs

b. AS >> C: Ek_{c} ($Ticket_{tgs}$)

Once per type of service:

c. C >> TGS: IDc||IDv||$Ticket_{tgs}$

d. TGS >> C: $ticket_{v}$

Once per service session:

e. C >> V: IDc||$ticket_{v}$

$Ticket_{tgs}$= Ek_{tgs}(IDc||ADc||IDtgs||TS1||Lifetime1) $Ticket_{v}$= Ek_{v}(IDc||ADc||IDv||TS2||Lifetime2)

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 ($Ticket_{tgs}$) 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:

3. 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.

4. 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 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 (K_v) 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:

5. 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 nd 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.

X.509 Certificates

- **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**

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.

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

<u>IMPORTANT QUESTIONS:</u>

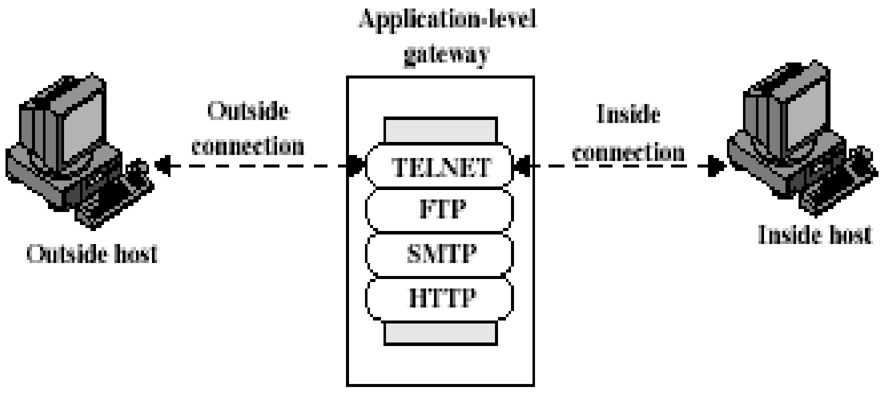
1. Explain web security.
2. What is secure socket layer?
3. What is SSL?
4. List the features of SSH?
5. Explain services defined by IEEE 802.11.

UNIT-V

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.

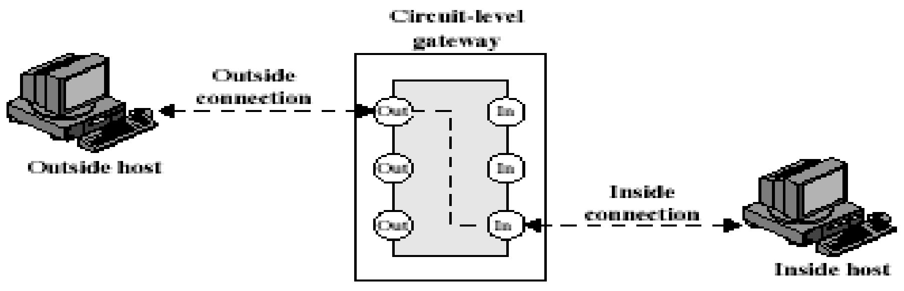


(b) Application-level gateway

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.



(c) Circuit-level gateway

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 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.

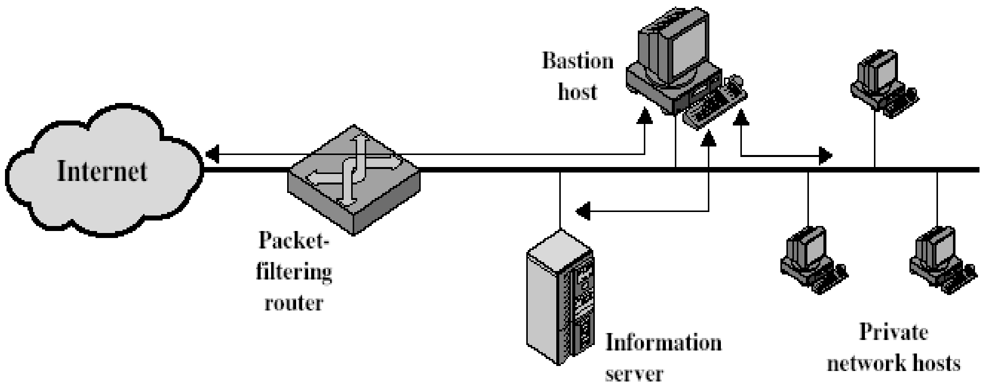
1. Screened host firewall, single-homed basiton configuration

In this configuration, the firewall consists of two systems: a packet filtering router and a bastion host. Typically, the router is configured so that

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.

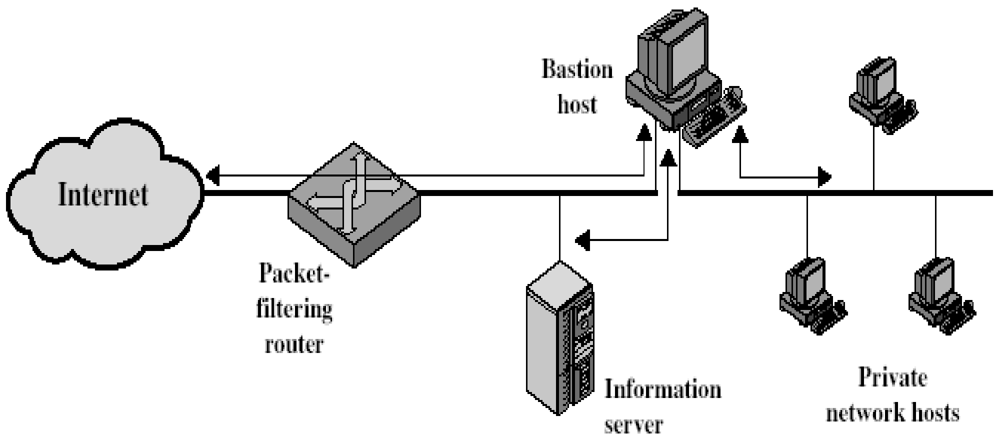
An intruder must generally penetrate two separate systems before the security of the internal network is compromised.



(a) Screened host firewall system (single-homed bastion host)

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.



(b) Screened host firewall system (dual-homed bastion host)

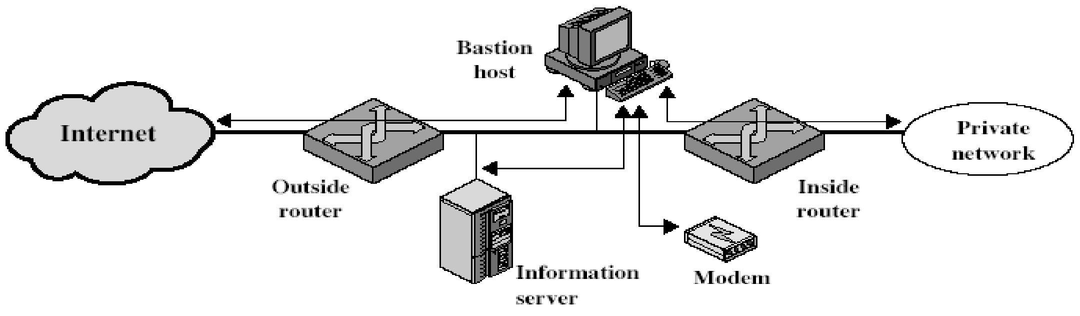
3. Screened subnet firewall configuration

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.



(c) Screened-subnet firewall system

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.

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

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.

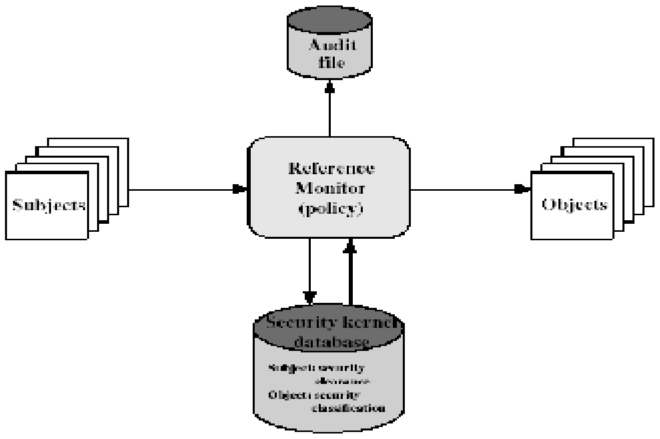

Audit file
Subjects
Reference Monitor (policy)
Objects
Security kernel database
Subject: security clearance
Object: security classification

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

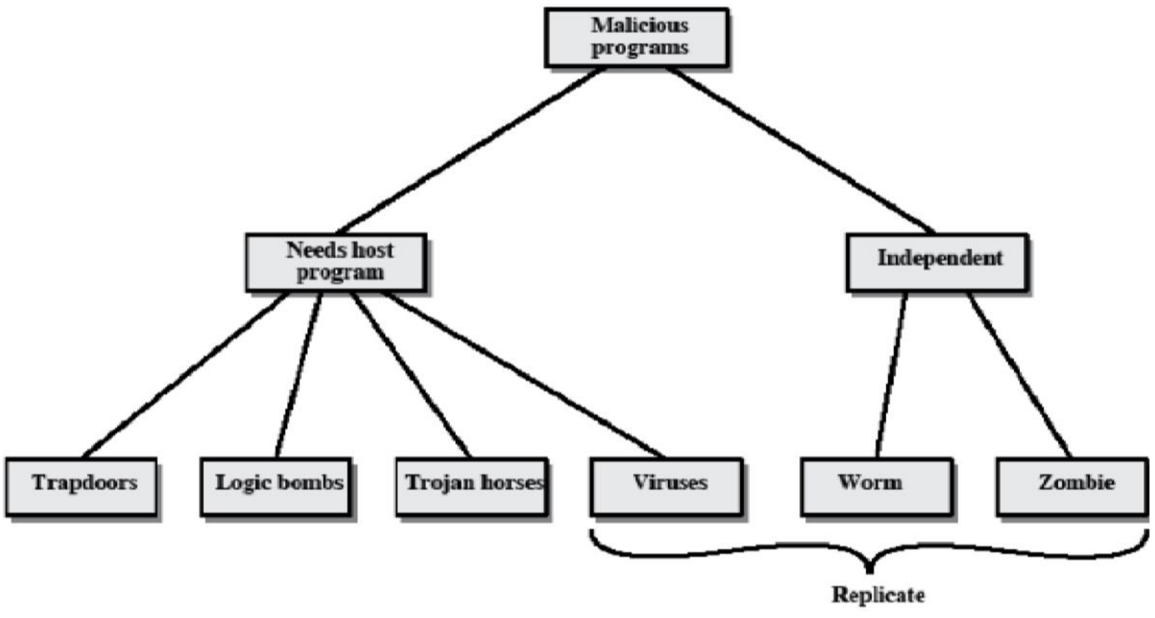


Figure 19.1 Taxonomy of Malicious Programs

Name	**Description**
Virus	Attaches itself to a program and propagates copies of itself to othe 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	Program modification that allows unauthorized access t 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 (viru 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 o 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 compute system and gained root-level access
Zombie	Program activated on an infected machine that is activated to launc 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.

of virus. Macro viruses are particularly threatening for a number of reasons:

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.

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.

IMPORTANT QUESTIONS:

1. Explain key management of IPSec.
2. Explain ISAKMP.
3. Explain the difference between IPSec and SSL.
4. Write short notes on virtual elections.
5. Write short notes on Internet key exchange.