X.509 CERTIFICATE

X.509 is a standard that defines the format of public key certificates. X.509 certificates are used in many Internet protocols, including TLS/SSL, which is the basis for HTTPS, the secure protocol for browsing the web. They're also used in offline applications, like electronic signatures. An X.509 certificate contains a public key and an identity (a hostname, or an organization, or an individual), and is either signed by a certificate authority or self-signed. When a certificate is signed by a certificate authority, or validated by another means, someone holding that certificate can rely on the public key it contains to establish secure communications with another party, or validate documents digitally signed by the corresponding private key.

Besides the format for certificates themselves, X.509 specifies certificate revocation lists as a means to distribute information about certificates that are no longer valid, and a certification path validation algorithm, which allows for certificates to be signed by intermediate CA certificates, which are in turn signed by other certificates, eventually reaching a trust anchor.

X.509 is defined by the International Telecommunications Union's Standardization sector (ITU-T), and is based on ASN.1, another ITU-T standard.

An X.509 certificate is something that can be used in software to both:

1. Verify a person’s identity so you can be sure that the person really is who they say they are.
2. Send the person who owns the certificate encrypted data that only they will be able to decrypt and read.

Digital Certificates and X.509 Authentication Service

The general format for a certificate is:

Version V

Serial number SN

Signature algorithm identifier AI

Issuer Name CA

Period of Validity TA

Subject Name A

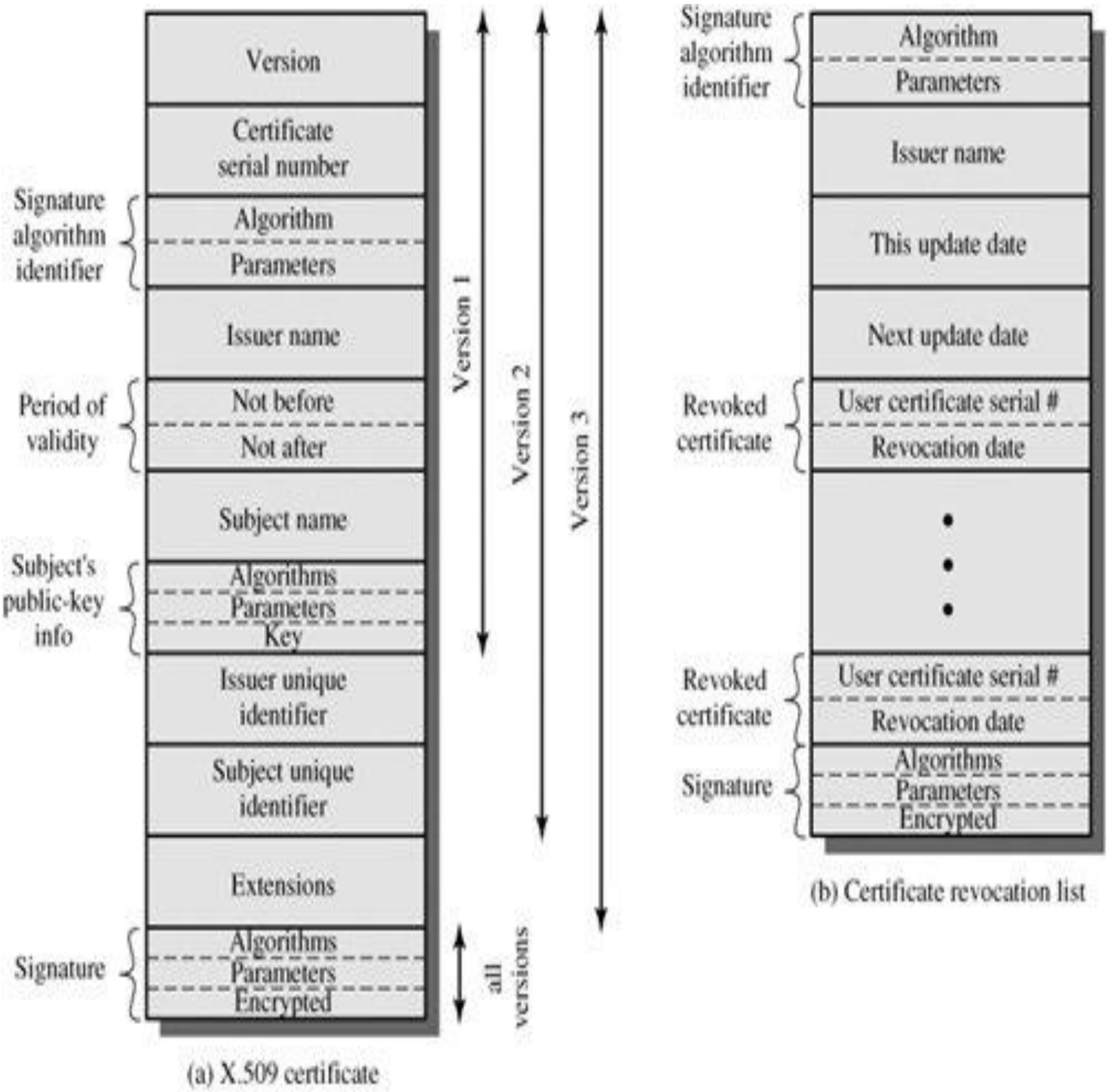
Subject’s Public-key Information Ap

Issuer Unique Identifier (added in Version 2)

Subject Unique Identifier (added in Version 2)

Extensions (added in Version 3)

Signature



(a) X.509 certificate

(b) Certificate revocation list

User certificates generated by a CA use the following standard notation:

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

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

Y {I} = the signing of I by Y consisting of I with an encrypted hash code appended

Obtaining A User Certificate „ User certificates generated by a CA have the following characteristics:

Any user with access to the public key of the CA can recover the user public key that was certified.

No party other than the CA can modify the certificate without being detected.

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

Kerberos

Kerberos was designed to mitigate the following problems in network security:

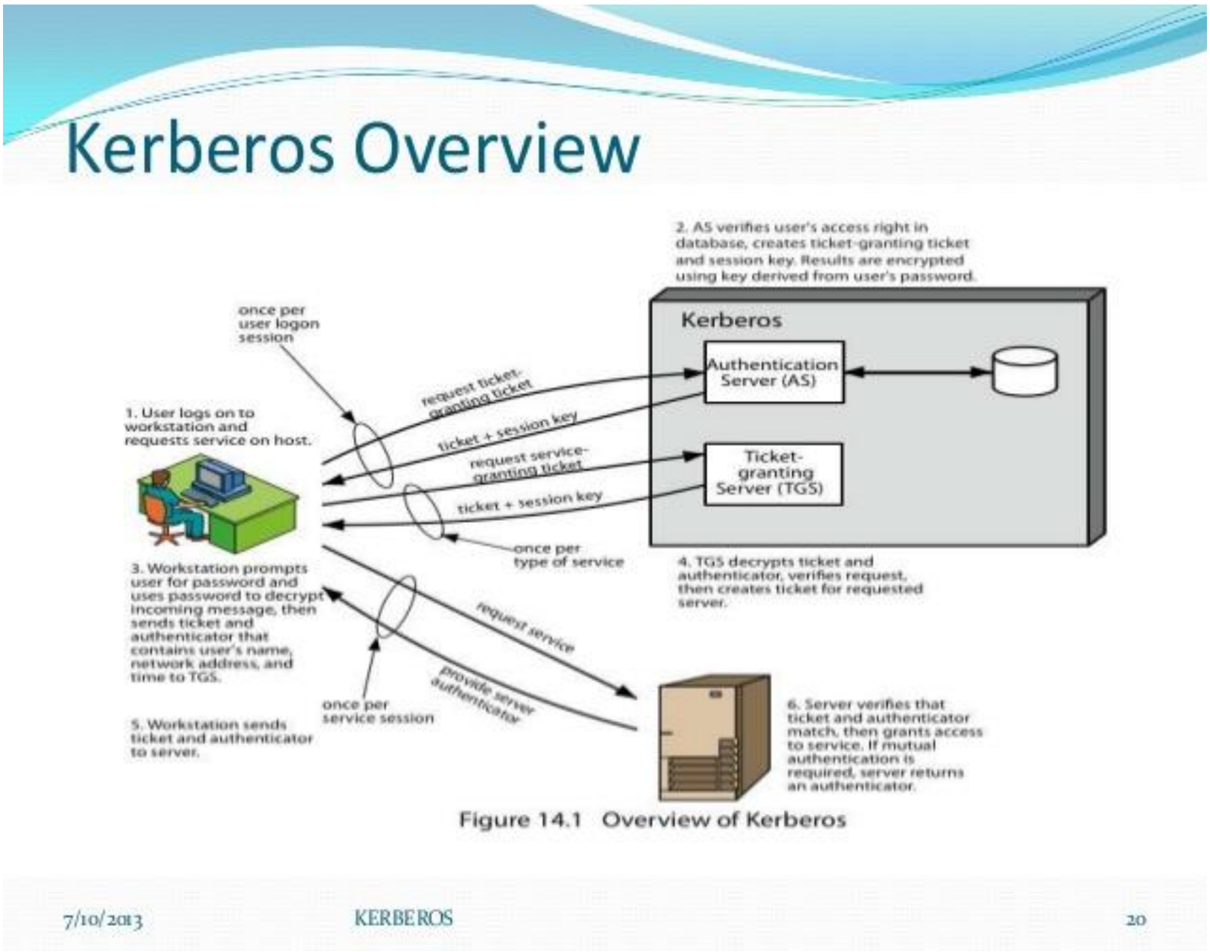
1. Password Sniffing
2. Password database stealing.
3. Password and login credential is centralized in kerberos infrastructure, which prevents clients from storing passwords on their machines.
4. Protocol weaknesses due to unencrypted data transfer on some network services can be reduced with the help of kerberos.

Disadvantages of Kerberos:

1. In kerberos infrastructure user login credentials are stored in the central server as mentioned before. So its a tedious job to migrate all login credentials from local machines /etc/passwd and /etc/shadow files to the central server.
2. If some attacker gets access to the central server, the entire infrastructure will be under threat.
3. the applications that can be protected using kerberos must have kerberos functionality inbuilt into them. There are many application programs currently without the support for kerberos.

some of the common terms used in Kerberos infrastructure.

- GSS-API : this is the API that must be present in application programs, to be compatible with Kerberos.
- KDC: Key distribution Centre, this will be the server which we call the middle man server or the central server arbitrator, which issues the keys for the communication.
- REALM: a kerberos network identified by a name, mostly this is the domain name in all caps.
- Principal: this is the name used by the kerberos central server to call users, service name etc.
- TGS: Ticket Granting Server: this is mostly the same central server (KDC server), it grants the tickets for a service.
- TGT: A special ticket which contains the session key for communication between the client machine and the central KDC server.



Kerberos Overview

2. AS verifies user's access right in database, creates ticket-granting ticket and session key. Results are encrypted using key derived from user's password.
Kerberos
Authentication Server (AS)
Ticket-granting Server (TGS)
once per user logon session
request ticket-granting ticket
ticket + session key
request service-granting ticket
ticket + session key
once per type of service
1. User logs on to workstation and requests service on host.
3. Workstation prompts user for password and uses password to decrypt incoming message, then sends ticket and authenticator that contains user's name, network address, and time to TGS.
4. TGS decrypts ticket and authenticator, verifies request, then creates ticket for requested server.
request service
provide server authenticator
once per service session
5. Workstation sends ticket and authenticator to server.
6. Server verifies that ticket and authenticator match, then grants access to service. If mutual authentication is required, server returns an authenticator.

Figure 14.1 Overview of Kerberos

7/10/2013 KERBEROS 20

Kerberos is an authentication service developed as part of Project Athena at MIT. The problem that Kerberos addresses is this: Assume an open distributed environment in which users at workstations wish to access services on servers distributed throughout the network .We would like for servers to be able to restrict access to authorized users and to be able to authenticate requests for service. In this environment, a workstation cannot be trusted to identify its users correctly to network services. In particular, the following three threats exist:

1. A user may gain access to a particular workstation and pretend to be another user operating from that workstation.
2. A user may alter the network address of a workstation so that the requests sent from the altered workstation appear to come from the impersonated workstation.
3. A user may eavesdrop on exchanges and use a replay attack to gain entrance to a server or to disrupt operations.

In any of these cases, an unauthorized user may be able to gain access to services and data that he or she is not authorized to access. Rather than building in elaborate authentication protocols at each server, Kerberos provides a centralized authentication server whose function is to authenticate users to servers and servers to users. Unlike most other authentication schemes, Kerberos relies exclusively on symmetric encryption, making no use of public-key encryption.

Two versions of Kerberos are in common use. Version 4 implementations still exist. Version 5 corrects some of the security deficiencies of version 4 and has been issued as a proposed Internet Standard.

Today‟s environment is a distributed architecture consisting of dedicated user workstations (clients) and distributed or centralized servers. In this environment, three approaches to security can be envisioned.

1. Rely on each individual client workstation to assure the identity of its user or users and rely on each server to enforce a security policy based on user identification (ID).

2. Require that client systems authenticate themselves to servers, but trust the client system concerning the identity of its user.

3. Require the user to prove his or her identity for each service invoked. Also require that servers prove their identity to clients.

In a small, closed environment in which all systems are owned and operated by a single organization, the first or perhaps the second strategy may suffice. But in a more open environment in which network connections to other machines are supported, the third approach is needed to protect user information and resources housed at the server. Kerberos supports this third approach. Kerberos assumes distributed client/server architecture and employs one or more Kerberos servers to provide an authentication service.

Sequence of steps in authentication with Kerberos version is given here in table 1(a) to 1(c)

(a) Authentication Service Exchange to obtain ticket-granting ticket.

$$1. C \rightarrow AS : ID_C \,\|\, ID_{tgs} \,\|\, TS_1$$

$$2.\ AS \rightarrow C : E\,(K_c\,, [\mathrm{K}_{c\,,\,tgs} \,\|\, ID_{tgs} \,\|\, TS_2 \,\|\, Lifetime_2 \,\|\, \mathrm{Ti}\ cket_{tgs}\,])$$

$$Ticket_{tgs} = E\,(K_{tgs}\,, [\mathrm{K}_{c\,,\,tgs} \,\|\, ID_C \,\|\, \mathrm{AD}_C \,\|\, \mathrm{ID}_{tgs} \,\|\, TS_2 \,\|\, Lifetime_2\,])$$

(b) Ticket-Granting Service Exchange to obtain service-granting ticket.

$$3.\ C \rightarrow TGS : ID_V \,\|\, ticket_{tgs} \,\|\, Authenticator_c$$

$$4.\ TGS \rightarrow C : E\,(K_{c\,,\,tgs}\,,\,[\mathrm{K}_{c\,,\mathrm{v}} \,\|\, ID_V \,\|\, TS_4 \,\|\, \mathrm{Ti}\ cket_V\,])$$

$$Ticket_{tgs} = E\,(K_{tgs}\,,\,[\mathrm{K}_{c\,,\,tgs} \,\|\, ID_C \,\|\, \mathrm{AD}_C \,\|\, \mathrm{ID}_{tgs} \,\|\, TS_2 \,\|\, Lifetime_2\,])$$

$$Ticket_v = E\,(K_v\,,\,[\mathrm{K}_{c\,,\mathrm{v}} \,\|\, ID_C \,\|\, \mathrm{AD}_C \,\|\, \mathrm{ID}_v \,\|\, TS_4 \,\|\, Lifetime_4\,])$$

$$Authenticator_c = E\,(K_{c\,,\,tgs}\,,\,[\,ID_C \,\|\, \mathrm{AD}_C \,\|\, TS_3\,])$$

Client/Server Authentication Exchange to obtain service

$$5.\ C \rightarrow V : \mathrm{T}\ icket_v \,\|\, Authenticator_c$$

$$6.\ V \rightarrow C : E\,(K_{c,\mathrm{v}}\,,\,[\,TS_5 + 1]) \text{ (for mutual authentication)}$$

$$Ticket_v = E\,(K_v\,,\,[\mathrm{K}_{c\,,\mathrm{v}} \,\|\, ID_C \,\|\, \mathrm{AD}_C \,\|\, \mathrm{ID}_v \,\|\, TS_4 \,\|\, Lifetime_4\,])$$

$$Authenticator_c = E\,(K_{c\,,\mathrm{v}}\,,\,[\,ID_C \,\|\, \mathrm{AD}_C \,\|\, TS_5\,])$$

Tables 1a to 1c: Summary of Kerberos Version4 Message Exchanges

The notations in the table are briefly explained here. The service TGS, issues tickets to users who have been authenticated to AS.

Kerberos protocol sequences of operations are described in tables 2 (a) to(c)

Message (1)	Client requests ticket-granting ticket.
ID_C	Tells AS identity of user from this client.
ID_{tgs}	Tells AS that user re quests access to TGS.
TS_1	Allows AS to verify that client‟s clock is synchronized with that of AS.
Message (2)	AS returns ticket-granting ticket.
K_C	Encryption is based on user‟s password, enabling AS and client to verify Password and protecting contents of message (2).

$K_{c,tgs}$	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.
ID_{tgs}	Confirms that this ticket is for the TGS.
TS_2	Informs client of time this ticket was issued.
$Lifetime_2$	Informs client of the lifetime of this ticket.
$Ticket_{tgs}$	Ticket to be used by client to access TGS.

(a) Authentication Service Exchange

Message (3)	Client requests service-granting ticket.
ID_V	Tells TGS that user requests access to server V.
$Ticket_{tgs}$	Assures TGS that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (4)	TGS returns service-granting ticket.
$K_{c,tgs}$	Key shared only by C and TGS protects contents of message (4).
$K_{c,v}$	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.
ID_V	Confirms that this ticket is for server V.
TS_4	Informs client of time this ticket was issued.
$Ticket_V$	Ticket to be used by client to access server V.
$Ticket_{tgs}$	Reusable so that user does not have to reenter password.
K_{tgs}	Ticket is encrypted with key known only to AS and TGS, to prevent tampering.
$K_{c,tgs}$	Copy of session key accessible to TGS used to decrypt authenticator, thereby authenticating ticket.

ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_{tgs}	Assures server that it has decrypted ticket properly.
TS_2	Informs TGS of time this ticket was issued.
$Lifetime_2$	Prevents replay after ticket has expired.
$Authenticator_c$	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.
$K_{c,tgs}$	Authenticator is encrypted with key known only to client and TGS, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.

AD_C	Must match address in ticket to authenticate ticket.
TS_3	Informs TGS of time this authenticator was generated.

(b) Ticket-Granting Service Exchange

Message (5)	Client requests service.
$Ticket_V$	Assures server that this user has been authenticated by AS.
$Authenticator_c$	Generated by client to validate ticket.
Message (6)	Optional authentication of server to client.
$K_{c,v}$	Assures C that this message is from V.
TS_5+1	Assures C that this is not a replay of an old reply.
$Ticket_V$	Reusable so that client does not need to request a new ticket from TGS for each

	access to the same server.
K_V	Ticket is encrypted with key known only to TGS and server, to prevent tampering.
$K_{c,v}$	Copy of session key accessible to client; used to decrypt authenticator, thereby authenticating ticket.
ID_C	Indicates the rightful owner of this ticket.
AD_C	Prevents use of ticket from workstation other than one that initially requested the ticket.
ID_V	Assures server that it has decrypted ticket properly.
TS_4	Informs server of time this ticket was issued.
$Lifetime_4$	Prevents replay after ticket has expired.
$Authenticator_c$	Assures server that the ticket presenter is the same as the client for whom the ticket was issued; has very short lifetime to prevent replay.
$K_{c,v}$	Authenticator is encrypted with key known only to client and server, to prevent tampering.
ID_C	Must match ID in ticket to authenticate ticket.
AD_C	Must match address in ticket to authenticate ticket.
TS_5	Informs server of time this authenticator was generated.

(c) Client/Server Authentication Exchange

Table 2: Rationale for the Elements of the Kerberos Version 4 Protocol

E-MAIL SECURITY

In virtually all distributed environments, electronic mail is the most heavily used network-based application. Users expect to be able to, and do, send e-mail to others who are connected directly or indirectly to the Internet, regardless of host operating system or communications suite. With

the explosively growing reliance on e-mail, there grows a demand for authentication and confidentiality services. Two schemes stand out as approaches that enjoy widespread use: Pretty Good Privacy (PGP) and S/MIME.

Pretty good privacy:

PGP is a remarkable phenomenon. Largely the effort of a single person, Phil Zimmermann, PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications. In essence, Zimmermann has done the following:

1. Selected the best available cryptographic algorithms as building blocks.
2. Integrated 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.
3. Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks such as AOL (America On Line).
4. Entered into an agreement with a company (Viacrypt, now Network Associates) 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.

1. It is available free worldwide in versions that run on a variety of platforms, including Windows, UNIX, Macintosh, and many more. In addition, the commercial version satisfies users who want a product that comes with vendor support.
2. It is based on algorithms that have survived extensive public review and are considered extremely secure. Specifically, the package includes RSA, DSS, and Diffie-Hellman for

public-key encryption; CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.

3. It has a wide range of applicability, from corporations that wish to select and enforce a standardized scheme for encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet and other networks.
4. It was not developed by, nor is it controlled by, any governmental or standards organization. For those with an instinctive distrust of "the establishment," this makes PGP attractive.
5. PGP is now on an Internet standards track. Nevertheless, PGP still has an aura of an antiestablishment endeavor.

Notation:

The following symbols are used.

K_S = session key used in symmetric encryption scheme

PR_a = private key of user A, used in public-key encryption
scheme PU_a = public key of user A, used in public-key
encryption scheme EP=public-key encryption

DP= public-key
decryption
EC=symmetric
encryption DC =
symmetric decryption
H=Hash function
||=concatenation

Z=compression using ZIP algorithm
R64=conversion to radix 64 ASCII
format

The PGP documentation often uses the term *secret key* to refer to a key paired with a public key in a public-key encryption scheme. As was mentioned earlier, this practice risks confusion with a secret key used for symmetric encryption. Hence, we use the term *private key* instead.

The figures below exhibit the three modes of PGP.

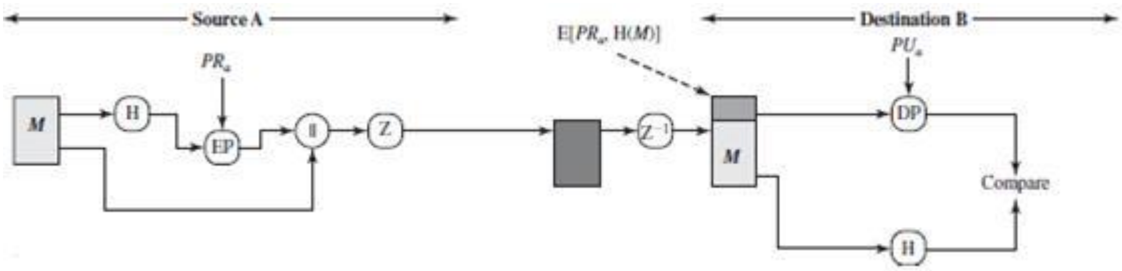


Figure 1: Authentication only

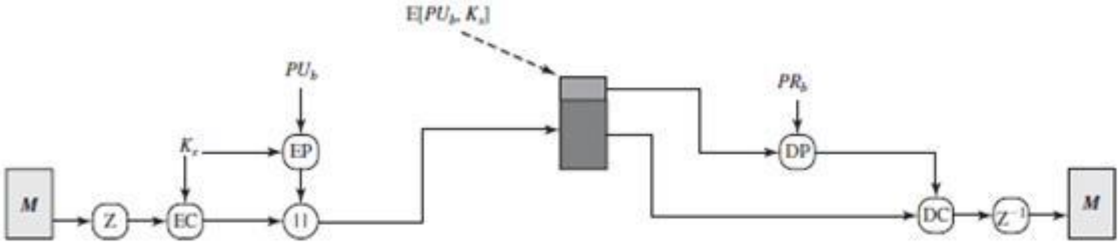


Figure 2: Confidentiality only

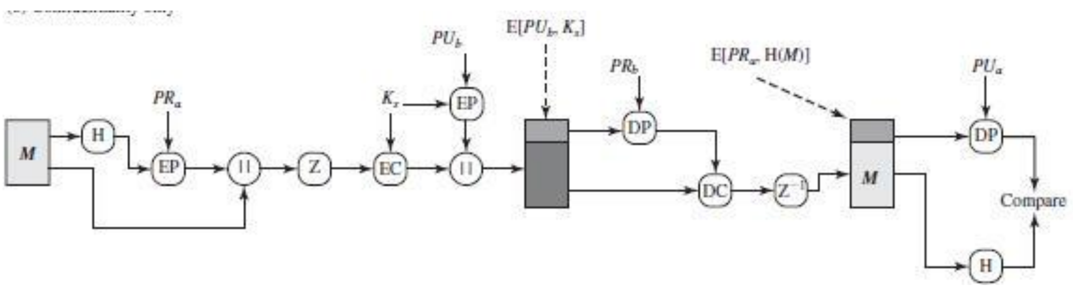


Figure 3: Confidentiality and authentication

Sequences of steps for the PGP service in figure 1 is as follows:

1. The sender creates a message.
2. SHA-1 is used to generate a 160-bit hash code of the message.

3. The hash code is encrypted with RSA using the sender‟s private key, and the result is prepended to the message.

4. The receiver uses RSA with the sender‟s public key to decrypt and recover the hash code.

5. 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.

The PGP service in figure 2 has the following steps:

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 (or IDEA or 3DES) with the session key-Message confidentiality.

3. The session key is encrypted with RSA using the recipient‟s public key and is prepended to the message- session key confidentiality.

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.

Finally the PGP service in figure 3 offers both confidentiality and authentication.

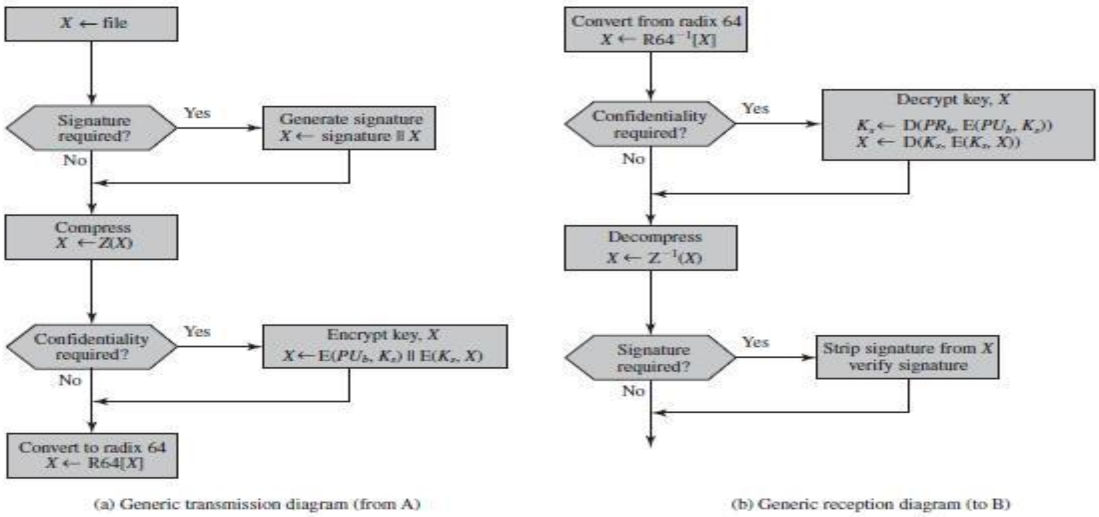
First, a signature is generated for the plaintext message and prepended to the message. Then the plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal). This sequence is preferable to the opposite: encrypting the message and then generating a signature for the encrypted message. It is generally more convenient to store a signature with a plaintext version of a message. Furthermore, for purposes of third-party verification, if the signature is performed first, a third party need not be concerned with the symmetric key when verifying the signature.

In summary, when both services are used, the sender first signs the message with its own private key, then encrypts the message with a session key, and finally encrypts the session key with the recipient‟s public key.

E-mail compatibility: When PGP is used, at least part of the block to be transmitted is encrypted. If only the signature service is used, then the message digest is encrypted (with the sender‟s private key). If the confidentiality service is used, the message plus signature (if present) are encrypted (with a one-time symmetric key). Thus, part or the entire resulting block consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permit the use of blocks consisting of ASCII text. 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. This format also appends a CRC to detect transmission errors.

Figure 4 shows the relationship among the four services so far discussed. On transmission (if it is required), a signature is generated using a hash code of the uncompressed plaintext. Then the plaintext (plus signature if present) is compressed. Next, if confidentiality is required, the block (compressed plaintext or compressed signature plus plaintext) is encrypted and prepended with the public-key encrypted symmetric encryption key. Finally, the entire block is converted to radix-64 format.

On reception, the incoming block is first converted back from radix-64 format to binary. Then, if the message is encrypted, the recipient recovers the session key and decrypts the message. The resulting block is then decompressed. If the message is signed, the recipient recovers the transmitted hash code and compares it to its own calculation of the hash code.



X ← file
Signature required?
Yes
Generate signature X ← signature ll X
No
Compress X ← Z(X)
Confidentiality required?
Yes
Encrypt key, X X ← E(PU_b, K_s) ll E(K_s, X)
No
Convert to radix 64 X ← R64[X]
(a) Generic transmission diagram (from A)
Convert from radix 64 X ← R64^-1[X]
Confidentiality required?
Yes
Decrypt key, X K_s ← D(PR_b, E(PU_b, K_s)) X ← D(K_s, E(K_s, X))
No
Decompress X ← Z^-1(X)
Signature required?
Yes
Strip signature from X verify signature
No
(b) Generic reception diagram (to B)

(a) Generic transmission diagram (from A)   b) Generic reception diagram (to B)

Figure 4: Transmission and Reception of PGP Messages

Cryptographic Keys and Key Rings

PGP makes use of four types of keys: one-time session symmetric keys, public keys, private keys, and passphrase-based symmetric keys. Three separate requirements can be identified with respect to these keys:

1. A means of generating unpredictable session keys is needed.

2. A user is allowed to have multiple public-key/private-key pairs.

3. Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

PGP Session Keys

Each session key is associated with a single message and is used only for the purpose of encrypting and decrypting that message. Random numbers are generated using the algorithm specified in ANSI X12.17, with inputs based on keystroke input from the user, where both the keystroke timing and the actual keys struck are used to generate a randomized stream of numbers.

Key Identifiers

In PGP, any given user may have multiple public/private key pairs. That means, a user may have many public/private key pairs at his disposal. He wishes to encrypt or sign a message using one of his keys. But, the problem of informing the other party, which key he has used arises. Attaching the whole public key every time is inefficient. Rather PGP uses a key identifier based on the least significant 64-bits of the key, which will very likely be unique. That is, the key ID of public PUa is

(PUa mod 2 64). Then only the much shorter key ID would need to be transmitted with any message. A key ID is also required for the PGP digital signature.

PGP Message Format

A message consists of three components: the message component, a signature (optional), and a session key component (optional).

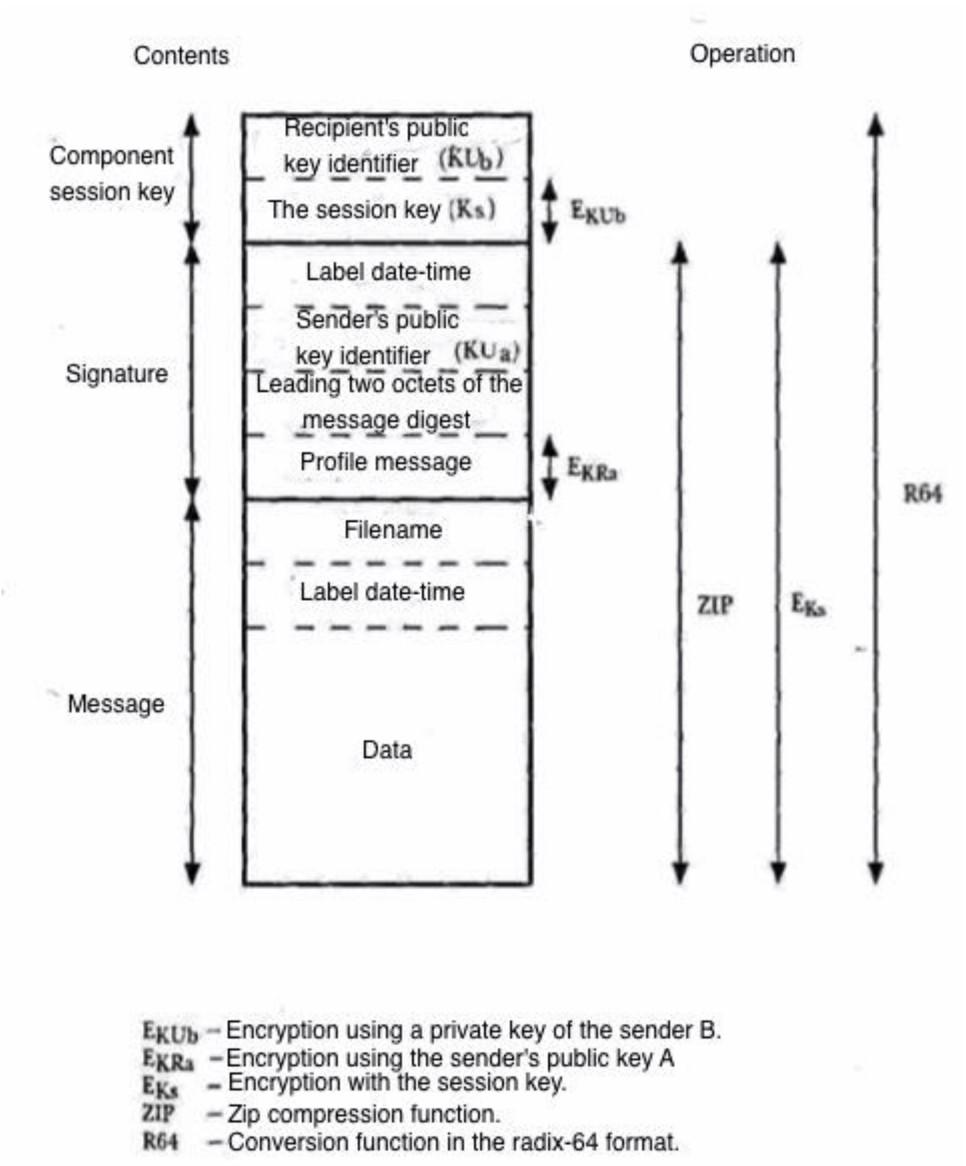
The message component includes the actual data to be stored or transmitted, as well as a filename and a timestamp that specifies the time of creation. The signature component includes the following:

Timestamp: The time at which the signature was made.

Message digest: The 160-bit SHA-1 digest, encrypted with the sender's private signature key.

Leading two octets of message digest: To enable the recipient to determine if the correct public key was used to decrypt the message digest for authentication, by comparing this plaintext copy of the first two octets with the first two octets of the decrypted digest. These octets also serve as a 16-bit frame check sequence for the message

Key ID of sender's public key: Identifies the public key that should be used to decrypt the message digest and, hence, identifies the private key that was used to encrypt the message digest


Contents
Operation
Component session key
Recipient's public key identifier (KUb)
The session key (Ks)
EKUb
Signature
Label date-time
Sender's public key identifier (KUa)
Leading two octets of the message digest
Profile message
EKRa
Message
Filename
Label date-time
Data
ZIP
EKs
R64
EKUb – Encryption using a private key of the sender B.
EKRa – Encryption using the sender's public key A
EKs – Encryption with the session key.
ZIP – Zip compression function.
R64 – Conversion function in the radix-64 format.

The session key component includes the session key and the identifier of the recipient's public key that was used by the sender to encrypt the session key. The entire block is usually encoded with radix-64 encoding.

PGP Key Rings

Keys & key IDs are critical to the operation of PGP. These keys need to be stored and organized in a systematic way for efficient and effective use by all parties. PGP uses a pair of data structures, one to store the user’s public/private key pairs - their private-key ring; and one to store the public keys of other known users, their public-key ring.

General Structure of Private- and Public-Key Rings

a) **Private-Key Ring**

The Private-Key ring can be viewed as a table, in which each row represents one of the public/private key pairs owned by this user. Each row contains the following entries:

- **Timestamp**: The date/time when this key pair was generated.
- Key ID: The least significant 64 bits of the public key for this entry.
- Public key: The public-key portion of the pair.
- Private key: The private-key portion of the pair; this field is encrypted.
- User ID: Typically, this will be the user's e-mail address (e.g., stallings@acm.org).

The private-key ring is intended to be stored only on the machine of the user that created and owns the key pairs, and that it be accessible only to that user, it makes sense to make the value of the private key as secure as possible. Accordingly, the private key itself is not stored in the key ring. Rather, this key is encrypted using CAST-128 (or IDEA or 3DES). The procedure is as follows:

1. The user selects a passphrase to be used for encrypting private keys.

2. When the system generates a new public/private key pair using RSA, it asks the user for the passphrase. Using SHA-1, a 160-bit hash code is generated from the passphrase, and the passphrase is discarded.

3. The system encrypts the private key using CAST-128 with the 128 bits of the hash code as the key. The hash code is then discarded, and the encrypted private key is stored in the private-key ring.

Subsequently, when a user accesses the private-key ring to retrieve a private key, he or she must supply the passphrase. PGP will retrieve the encrypted private key, generate the hash code of the passphrase, and decrypt the encrypted private key using CAST-128 with the hash code. . As in any system based on passwords, the security of this system depends on the security of the password, which should be not easily guessed but easily remembered.

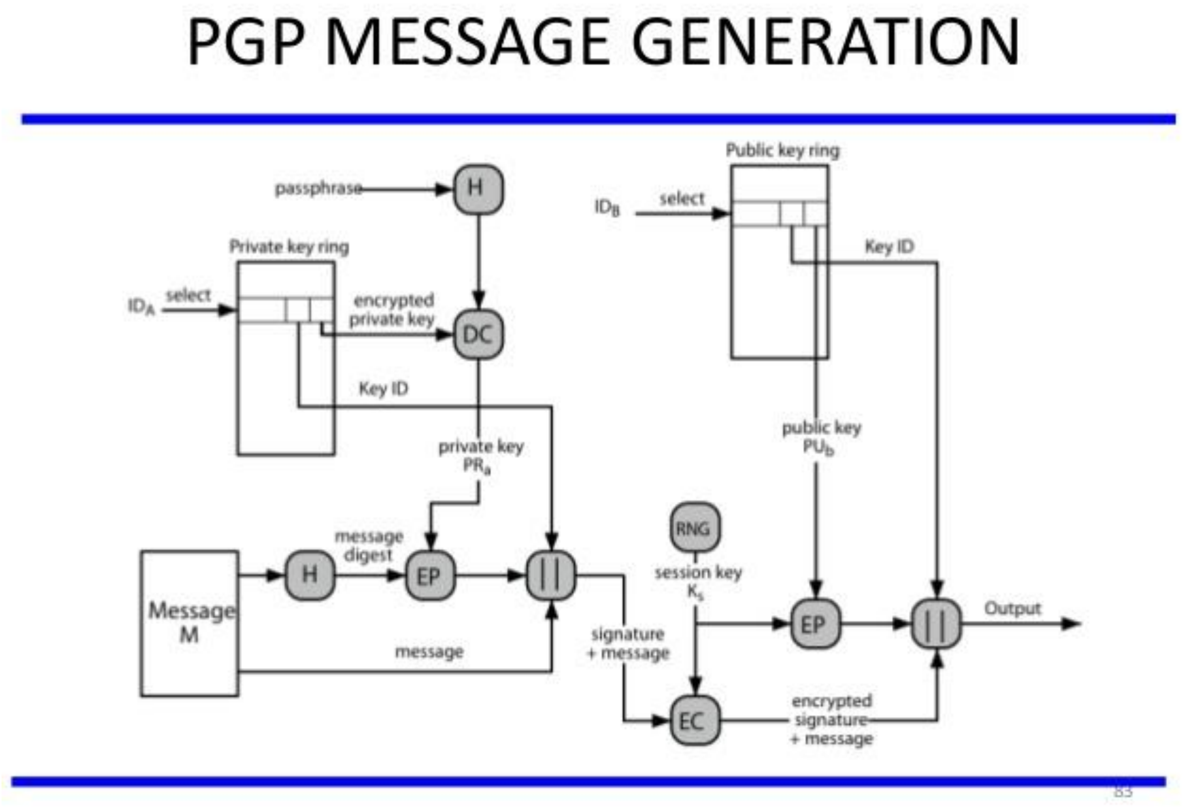
b) Public-key Ring This data structure is used to store public keys of other users that are known to this user.

Timestamp: The date/time when this entry was generated.

Key ID: The least significant 64 bits of the public key for this entry**.**

Public Key: The public key for this entry

User ID: Identifies the owner of this key. Multiple user IDs may be associated with a single public key



PGP MESSAGE GENERATION

The sending PGP entity performs the following steps:

1. Signing the message

a. PGP retrieves the sender's private key from the private-key ring using your_userid as an index. If your_userid was not provided in the command, the first private key on the ring is retrieved.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

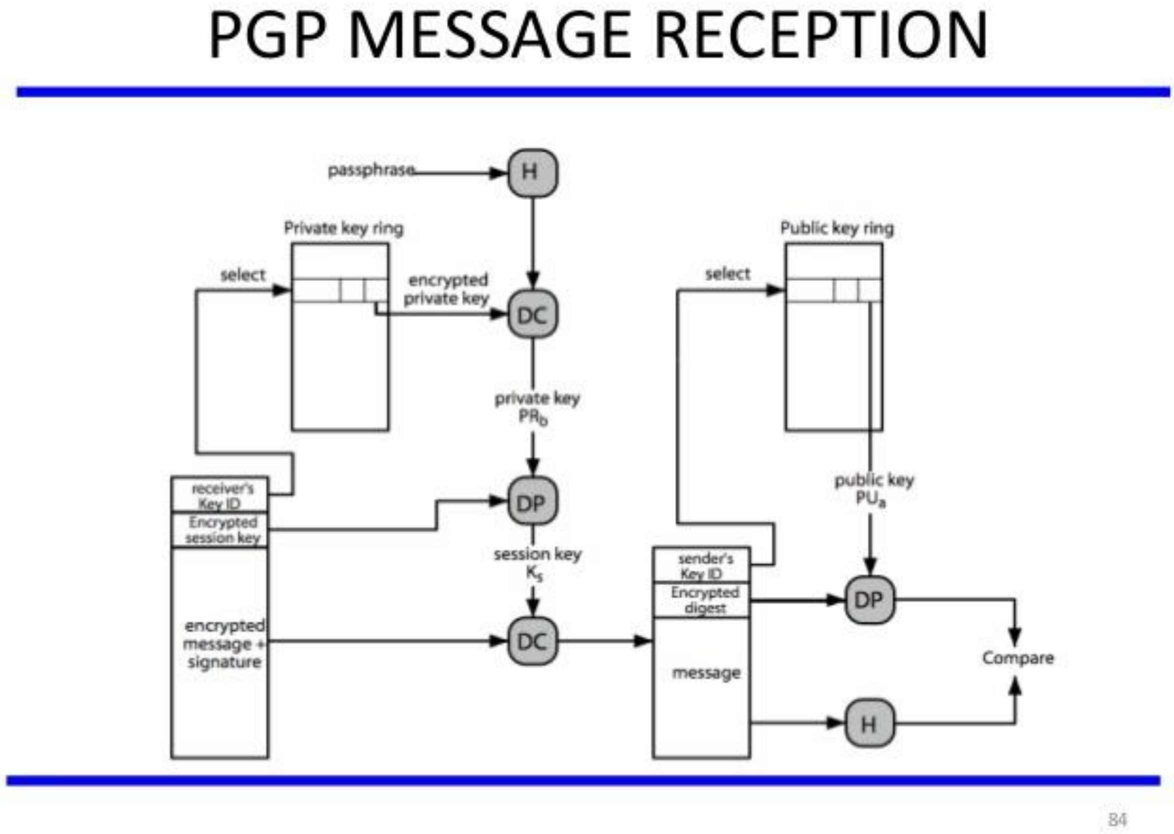
c. The signature component of the message is constructed.

2. Encrypting the message

a. PGP generates a session key and encrypts the message.

b. PGP retrieves the recipient's public key from the public-key ring using her_userid as an index.

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



PGP MESSAGE RECEPTION

The receiving PGP entity performs the following steps:

1. Decrypting the message

a. 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.

b. PGP prompts the user for the passphrase to recover the unencrypted private key.

c. PGP then recovers the session key and decrypts the message**.**

2. Authenticating the message

a. 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.

b. PGP recovers the transmitted message digest.

c. PGP computes the message digest for the received message and compares it to the transmitted message digest to authenticate.

S/MIME (Secure/Multipurpose Internet Mail Extension)

S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, which in turn provided support for varying content types and multi-part messages over the text only support in the original Internet RFC822 email standard. MIME allows encoding of binary data to textual form for transport over traditional RFC822 email systems. S/MIME is defined in a number of documents, most importantly RFCs 3369, 3370, 3850 and 3851 and S/MIME support is now included in many modern mail agents.

RFC 822 RFC 822 defines a format for text messages that are sent using electronic mail and it has been the standard for Internet-based text mail message. The overall structure of a message that conforms to RFC 822 is very simple. A message consists of some number of header lines (the header) followed by unrestricted text (the body). The header is separated from the body by a blank line. A header line usually consists of a keyword, followed by a colon, followed by the keyword's arguments; the format allows a long line to be broken up into several lines. The most frequently used keywords are From, To, Subject, and Date.

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.

Problems with RFC 822 and SMTP

• Executable files or other binary objects must be converted into ASCII. Various schemes exist (e.g., Unix UUencode), but a standard is needed

• Text data that includes special characters (e.g., Hungarian text) cannot be transmitted as SMTP is limited to 7-bit ASCII

• Some servers reject mail messages over a certain size

• Some common problems exist with the SMTP implementations which do not adhere completely to the SMTP standards defined in RFC 821. They are:

- delete, add, or reorder CR and LF characters
- truncate or wrap lines longer than 76 characters remove trailing white space (tabs and spaces)
- pad lines in a message to the same length
- convert tab characters into multiple spaces

MIME is intended to resolve these problems in a manner that is compatible with existing RFC 822 implementations and the specification is provided in RFC’s 2045 through 2049.

The MIME specification includes the following elements:

1. Five new message header fields are defined, which provide information about the body of the message.

2. A number of content formats are defined, thus standardizing representations that support multimedia electronic mail.

3. Transfer encodings are defined that protect the content from alteration by the mail system.

MIME - New header fields

The five header fields defined in MIME are as follows:

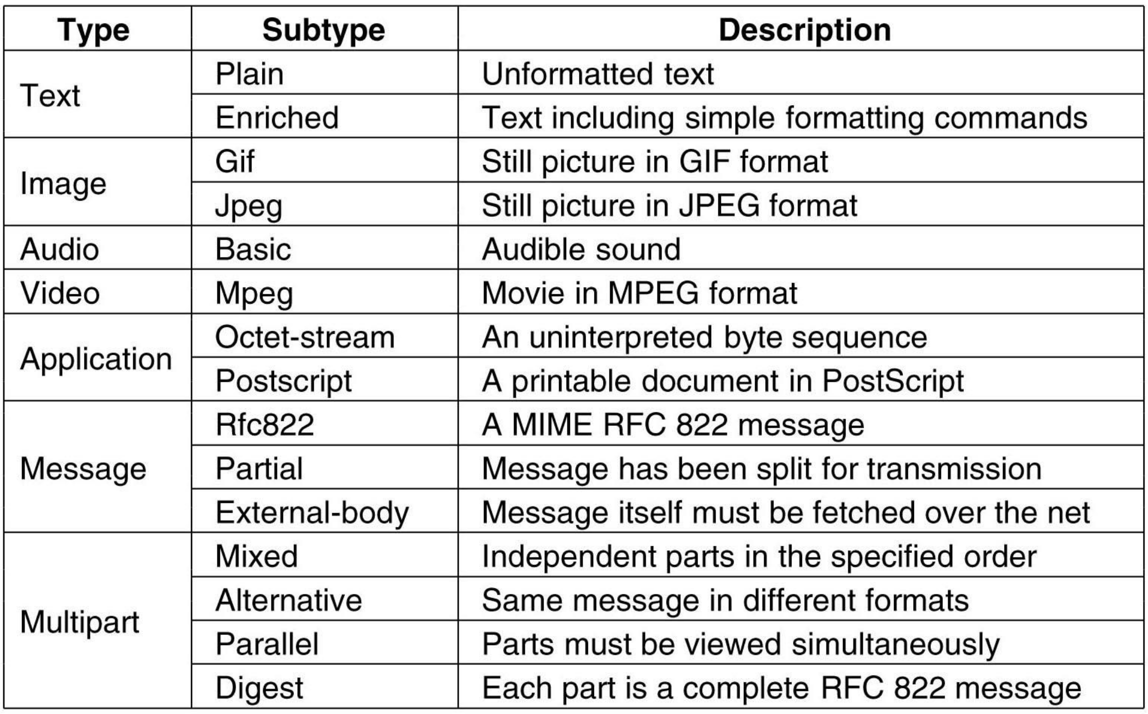
- **MIME-Version**: Must have the parameter value 1.0. This field indicates that the message conforms to RFCs 2045 and 2046.
- **Content-Type**: Describes the data contained in the body with sufficient detail that the receiving user agent can pick an appropriate agent or mechanism to represent the data to the user or otherwise deal with the data in an appropriate manner.
- **Content-Transfer-Encoding:** Indicates the type of transformation that has been used 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 useful when 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. There are seven different major types of content and a total of 15 subtypes. In general, a content type declares the general type of data, and the subtype specifies a particular format for that type of data.

For the text type of body, the primary subtype is plain text, 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. This boundary should not appear in any parts of the message. Each boundary starts on a new line and consists of two hyphens followed by the boundary value. The final boundary, which indicates the end of the last part, also has a suffix of two hyphens. Within each part, there may be an optional ordinary MIME header. There are four subtypes of the multipart type, all of which have the same overall syntax.

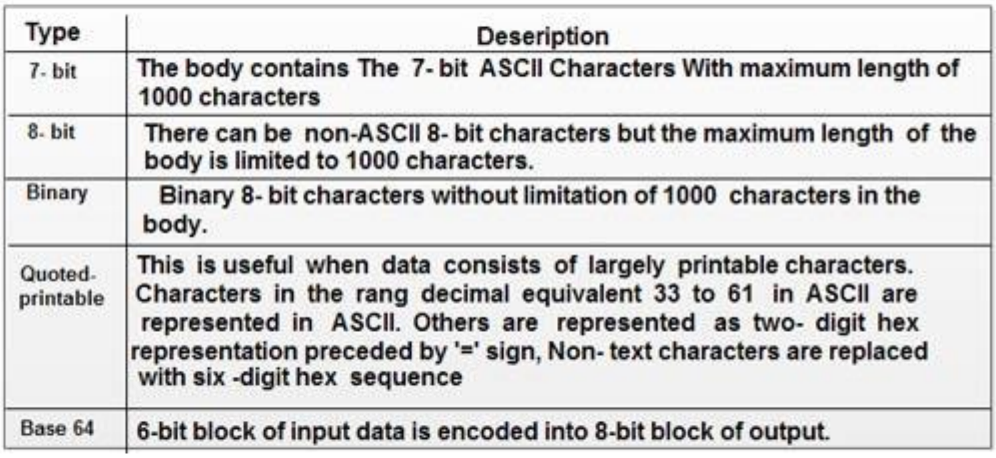


Type	Subtype	Description
Text	Plain	Unformatted text
	Enriched	Text including simple formatting commands
Image	Gif	Still picture in GIF format
	Jpeg	Still picture in JPEG format
Audio	Basic	Audible sound
Video	Mpeg	Movie in MPEG format
Application	Octet-stream	An uninterpreted byte sequence
	Postscript	A printable document in PostScript
Message	Rfc822	A MIME RFC 822 message
	Partial	Message has been split for transmission
	External-body	Message itself must be fetched over the net
Multipart	Mixed	Independent parts in the specified order
	Alternative	Same message in different formats
	Parallel	Parts must be viewed simultaneously
	Digest	Each part is a complete RFC 822 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. 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.



Type	Deseription
7- bit	The body contains The 7- bit ASCII Characters With maximum length of 1000 characters
8- bit	There can be non-ASCII 8- bit characters but the maximum length of the body is limited to 1000 characters.
Binary	Binary 8- bit characters without limitation of 1000 characters in the body.
Quoted-printable	This is useful when data consists of largely printable characters. Characters in the rang decimal equivalent 33 to 61 in ASCII are represented in ASCII. Others are represented as two- digit hex representation preceded by '=' sign, Non- text characters are replaced with six -digit hex sequence
Base 64	6-bit block of input data is encoded into 8-bit block of output.

The MIME standard defines two methods of encoding data. The Content-Transfer-Encoding field can actually take on six values. Three of these values (7bit, 8bit, and binary) indicate that no encoding has been done but provide some information about the nature of the data. 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. Two schemes are defined to provide a choice between a transfer technique that is essentially human readable and one that is safe for all types of data in a way that is reasonably compact.

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

S/MIME has a very similar functionality to PGP. Both offer the ability to sign and/or encrypt messages.

Functions

S/MIME provides the following functions:

- Enveloped data: This consists of encrypted content of any type and encryptedcontent encryption keys for one or more recipients.
- Signed data: A digital signature is formed by taking the message digest of the content 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 is formed. 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 be nested, so that encrypted data may be signed and signed data or clear-signed data may be encrypted.

IP security

To provide security, the IAB (Internet Architecture Board) included authentication and encryption as necessary security features in the next-generation IP, which has been issued as IPv6. Fortunately, these security capabilities were designed to be usable both with the current IPv4 and the future IPv6. This means that vendors can begin offering these features now, and many vendors now do have some IPsec capability in their products. The IPsec specification now exists as a set of Internet standards.

Applications of IP security

IPsec provides the capability to secure communications across a LAN, across private and public WANs, and across the Internet. Examples of its use include:

- **Secure branch office connectivity over the Internet:** A company can build a secure virtual private network over the Internet or over a public WAN. This enables a business to rely heavily on the Internet and reduce its need for private networks, saving costs and network management overhead.

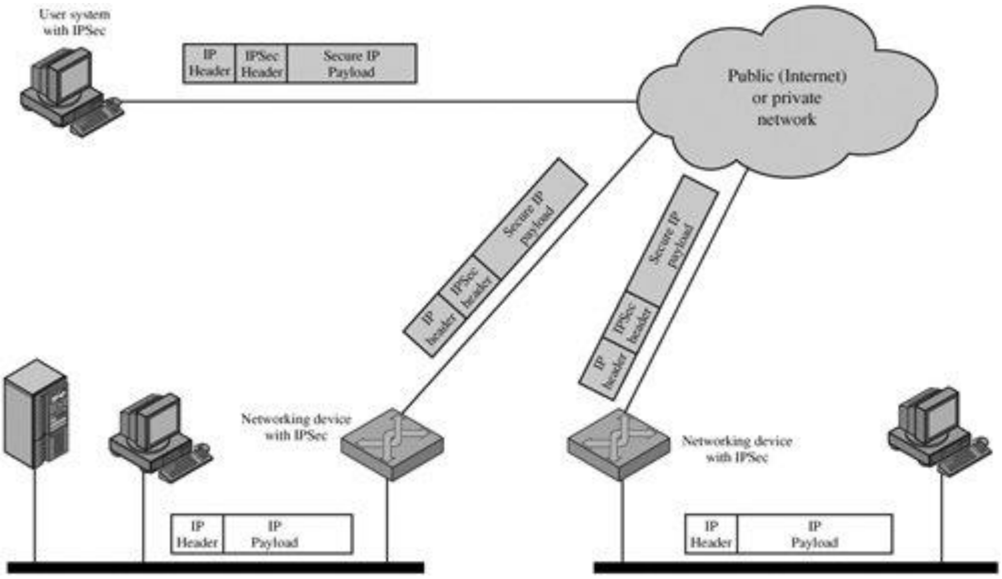
- **Secure remote access over the Internet:** An end user whose system is equipped with IP security protocols can make a local call to an Internet Service Provider (ISP) and gain secure access to a company network. This reduces the cost of toll charges for traveling employees and telecommuters.

- **Establishing extranet and intranet connectivity with partners:** IPsec can be used to secure communication with other organizations, ensuring authentication and confidentiality and providing a key exchange mechanism.

- **Enhancing electronic commerce security:** Even though some Web and electronic commerce applications have built-in security protocols, the use of IPsec enhances that security. IPsec guarantees that all traffic designated by the network administrator is both encrypted and authenticated, adding an additional layer of security to whatever is provided at the application layer.

The principal feature of IPsec that enables it to support these varied applications is that it can encrypt and/or authenticate *all* traffic at the IP level. Thus, all distributed applications (including remote logon, client/server, e-mail, file transfer, Web access, and so on) can be secured.

Figure is a typical scenario of IPsec usage. An organization maintains LANs at dispersed locations. Non-secure IP traffic is conducted on each LAN. For traffic offsite, through some sort of private or public WAN, IPsec protocols are used. These protocols operate in networking devices, such as a router or a firewall, that connect each LAN to the outside world. The IPsec networking device will typically encrypt and compress all traffic going into the WAN and decrypt and decompress traffic coming from the WAN; these operations are transparent to workstations and servers on the LAN. Secure transmission is also possible with individual users who dial into the WAN. Such user workstations must implement the IPsec protocols to provide security.



Benefits of IP security

- When IPsec is implemented in a firewall or router, it provides strong security that can be applied to all traffic crossing the perimeter. Traffic within a company or workgroup does not incur the overhead of security-related processing.
- IPsec in a firewall is resistant to bypass if all traffic from the outside must use IP and the firewall is the only means of entrance from the Internet into the organization.
- IPsec is below the transport layer (TCP, UDP) and so is transparent to applications. There is no need to change software on a user or server system when IPsec is implemented in the firewall or router. Even if IPsec is implemented in end systems, upper-layer software, including applications, is not affected.
- IPsec can be transparent to end users. There is no need to train users on security mechanisms, issue keying material on a per-user basis, or revoke keying material when users leave the organization. IPsec can provide security for individual users if needed. This is useful for offsite workers and for setting up a secure virtual subnetwork within an organization for sensitive applications.

IP security services

IPsec provides security services at the IP layer by enabling a system to select required security protocols, determine the algorithm(s) to use for the service(s), and put in place any cryptographic keys required to provide the requested services. Two protocols are used to provide security: an authentication protocol designated by the header of the protocol, Authentication Header (AH); and a combined encryption/ authentication protocol designated by the format of the packet for that protocol, Encapsulating Security Payload (ESP). RFC 4301 lists the following services:

1. Access control
2. Connectionless integrity
3. Data origin authentication
4. Rejection of replayed packets (a form of partial sequence integrity)
5. Confidentiality (encryption)

Limited traffic flow confidentiality

Modes of operation

IPsec can be implemented in a host-to-host transport mode, as well as in a network tunneling mode.

Transport mode

In transport mode, only the payload of the IP packet is usually encrypted or authenticated. The routing is intact, since the IP header is neither modified nor encrypted; however, when the authentication header is used, the IP addresses cannot be modified by network address translation, as this always invalidates the hash value. The transport and application layers are always secured by a hash, so they cannot be modified in any way, for example by translating the port numbers.

A means to encapsulate IPsec messages for NAT traversal has been defined by RFC documents describing the NAT-T mechanism.

Tunnel mode

In tunnel mode, the entire IP packet is encrypted and authenticated. It is then encapsulated into a new IP packet with a new IP header. Tunnel mode is used to create virtual private networks for network-to-network communications (e.g. between routers to link sites), host-to-network communications (e.g. remote user access) and host-to-host communications (e.g. private chat).

Tunnel mode supports NAT traversal.

INTRUSION DETECTION

Unauthorized intrusion into a computer system or network is one of the most serious threats to computer security. User trespass can take the form of unauthorized logon to a machine or, in the case of an authorized user, acquisition of privileges or performance of actions beyond those that have been authorized. Software trespass can take the form of a virus, worm, or Trojan horse.

All these attacks relate to network security because system entry can be achieved by means of a network. However, these attacks are not confined to network-based attacks. A user with access to a local terminal may attempt trespass without using an intermediate network. A virus or Trojan horse may be introduced into a system by means of a diskette. Only the worm is a uniquely network phenomenon. Thus, system trespass is an area in which the concerns of network security and computer security overlap.

At first, we examine the nature of attacks to systems. One of the common threats to security of a system is intruder, also referred to as hacker or cracker. Intruders are classified into three classes.

i. Masquerader: Unauthorized individual who penetrates a system‟s access control to exploit an authorized user‟s account.

ii. Misfeasor: Genuine user accesses data for which he is not authorized or he is authorized for access but misuses his or her privileges.

iii. Clandestine user: A person who seizes supervisory control of the system and uses this control to evade auditing and access controls or to suppress audit collection.

Masquerader is mostly an outsider whereas misfeasor is generally an insider. Clandestine user can be outsider or insider.

Intruder attacks range from benign to the serious. Sometimes people explore internets and see what is out there. This is benign type. There are cases where individuals attempt to read/modify data in the system, though unauthorized. This is serious attack. These attacks are still not under total control. We point out some techniques for intrusion.

Intrusion techniques

The objective of the intruder is to gain access to a system or to increase the range of privileges accessible on a system. Generally this requires the intruder to acquire information that is protected. Often this information is in the form of a user password.

With knowledge of some other user's password, an intruder can log in to a system and exercise all the privileges accorded to the legitimate user. Typically, a system must maintain a file that associates a password with each authorized user. If such a file is stored with no protection, then it is an easy matter to gain access to it and learn passwords. The password file can be protected in one of two ways:

- **One-way function:** The system stores only the value of a function based on the user's password. When the user presents a password, the system transforms that password and compares it with the stored value. In practice, the system usually performs a one-way transformation (not reversible) in which the password is used to generate a key for the one-way function and in which a fixed-length output is produced.

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

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

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

[illegible] particular system at leisure until a valid password that provided greater privileges was discovered.

Guessing attacks are feasible, and indeed highly effective, when a large number of guesses can be attempted automatically and each guess verified, without the guessing process being detectable.

The seventh method of attack, the Trojan horse, can be particularly difficult to counter. An example of a program that bypasses access controls was cited in the literature. A low-privilege user produced a game program and invited the system operator to use it in his or her spare time. The program did indeed play a game, but in the background it also contained code to copy the password file, which was unencrypted but access protected, into the user's file. Because the game was running under the operator's high-privilege mode, it was able to gain access to the password file.

The eighth attack listed namely line tapping, is a matter of physical security. It can be countered with link encryption techniques. Other intrusion techniques do not require learning a password. Intruders can get access to a system by exploiting attacks such as buffer overflows on a program that runs with certain privileges. Privilege escalation can be done this way as well.

We turn now to a discussion of the two principal countermeasures: detection and prevention. Detection is concerned with learning of an attack, either before or after its success. Prevention is a challenging security goal and an uphill battle at all times. The difficulty stems from the fact that the defender must attempt to thwart all possible attacks, whereas the attacker is free to try to find the weakest link in the defense chain and attack at that point

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:

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

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

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

Approaches for intrusion detection are

1. Statistical anomaly detection: Involves the collection of data relating to the behavior of legitimate users over a period of time. Then statistical tests are applied to observed behavior to determine with a high level of confidence whether that behavior is not legitimate user behavior.

> **a.** Threshold detection: This approach involves defining thresholds, independent of user, for the frequency of occurrence of various events.
>
> **b.** Profile based: A profile of the activity of each user is developed and used to detect changes in the behavior of individual accounts.

2. Rule-based detection: Involves an attempt to define a set of rules that can be used to decide that a given behavior is that of an intruder.

> **a.** Anomaly detection: Rules are developed to detect deviation from previous usage patterns.
>
> **b.** Penetration identification: An expert system approach that searches for suspicious behavior.

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

In terms of the types of attackers listed earlier, statistical anomaly detection is effective against masqueraders, who are unlikely to mimic the behavior patterns of the accounts they appropriate. 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.

A good example of detection-specific audit records reported in literature 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. All activity arises through commands issued by subjects. Subjects may be grouped into different access classes, and these classes may overlap.

- **Action:** Operation performed by the subject on or with an object; for example, login, read, perform I/O, execute.

- **Object:** Receptors of actions. Examples include files, programs, messages, records, terminals, printers, and user- or program-created structures. When a subject is the recipient of an action, such as electronic mail, then that subject is considered an object. Objects may be grouped by type. Object granularity may vary by object type and by environment. For example, database actions may be audited for the database as a whole or at the record level.
- **Exception-Condition:** Denotes which, if any, exception condition is raised on return.

- **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.

The audit records provide input to the intrusion detection using statistical anomaly detection in two ways. First, the designer must decide on a number of quantitative metrics that can be used to measure user behavior. An analysis of audit records over a period of time can be used to determine the activity profile of the average user. Thus, the audit records serve to define typical behavior. Second, current audit records are the input used to detect intrusion.

That is, the intrusion detection model analyzes incoming audit records to determine deviation from average behavior.

As far as rule based intrusion detection, audit records are examined as they are generated, and they are matched against the rule base. If a match is found, then the user's *suspicion rating* is increased. If enough rules are matched, then the rating will pass a threshold that results in the reporting of an anomaly.

Distributed Intrusion Detection

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

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

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