PGP NS

Introduction to Computer (Gandhigram Rural University)

UNIT 3

Pretty Good Privacy (PGP)

Definition of PGP Encryption :

- PGP stands for Pretty Good Privacy (PGP) which is invented by Phil Zimmermann.
- Pretty Good Privacy encryption is a data encryption computer program that gives cryptographic privacy and authentication for online communication. It is often used to encrypt and decrypt texts, emails, and files to increase the security of emails.
- PGP encryption uses a mix of data compression, hashing, and public-key cryptography. It also uses symmetric and asymmetric keys to encrypt data that is transferred across networks.
- It combines features of private and public key cryptography.

Basic Idea: PGP is a remarkable phenomenon. Zimmermann has suggested the following steps:

1. Selected the best cryptographic mechanisms (algorithms) as building blocks.
2. Integrated these algorithms into a general purpose application that is independent of operating system and processor
3. Made the package and its source code freely available via the Internet, bulletin boards, and commercial networks such as America On Line (AOL).
4. Entered into an agreement with a company to provide a fully compatible low cost commercial version of PGP.

Benefits of PGP: A number of reasons are sited for growth of PGP

- ✓ PGP is an open source and freely available software package for email security.
- ✓ PGP provides authentication through the use of Digital Signature.
- ✓ It provides confidentiality through the use of symmetric block encryption.
- ✓ It provides compression by using the ZIP algorithm, and EMAIL compatibility using the radix-64 encoding scheme.

Operational Description: PGP consists of the following five services:

1. Authentication
2. Confidentiality
3. Compression
4. E-mail compatibility
5. Segmentation

Table shows a summary of these services and the algorithms used to implement them

Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman or RSA	A message is encrypted using CAST-128 or IDEA or 3DES with a one-time session key generated by the sender. The session key is encrypted using Diffie-Hellman or RSA with the recipient's public key and included with the message.
Compression	ZIP	A message may be compressed, for storage or transmission, using ZIP.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.
Segmentation	—	To accommodate maximum message size limitations, PGP performs segmentation and reassembly.

Table:Summary of PGP services.

Authentication: The digital signature service is provided by PGP. The sequence is as follows:

1. The sender creates a message.
2. SHA-1 is used to generate 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 a message.
4. The receiver uses RSA with the sender's public key to decrypt and recover the hash code.
5. The receiver generates 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 message may be compressed using and algorithm called **ZIP**. This is represented by "Z" in the figure.
- The combination of SHA-1 and RSA provides an effective digital signature scheme. Due to the strength of RSA the recipient is assured that only the possessor of the matching private key can generate the signature. Because of the strength of SHA-1 the recipient is assured that no one else could generate a new message that matches the hash code and hence, the signature of the original message.
- As an alternative, DSS/SHA-1 algorithm can be used.

Confidentiality

- Another basic service provided by PGP is confidentiality which is provided by encrypting messages to be transmitted or to be stored locally as files.
- In both cases, symmetric algorithm CAST-128 may be used. Alternatively IDEA or 3DES may be used. The 64 bit cipher feedback (CFB) mode is used.
- The symmetric key is used only once and is created as a random number with the required number of bits.
- It is transmitted along with the message and is encrypted using the recepients public key.
- The sequence of steps:

1. The sender generates a message and a random number(128 bit) 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.
3. The session key is encrypted with RSA (or another algorithm known as ElGa- mal) using the recipients public key and is prepended to the message.
4. The receiver uses RSA with its private key to decrypt and recover the session key.
5. The session key is used to decrypt the message.

Confidentiality and Authentication

- Here, both services may be used for the same message.
- First, a signature is generated for the plaintext message and prepended to the message. Then the 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 can be done in reverse : Encrypting the message and then generating a signature of the encrypted message.

Compression

- As a default, PGP compresses the message after applying the signature but before encryption. This has the benefit of saving space both for e-mail transmission and for file storage.
- It indicates Z for compression and Z^{-1} for decompression.

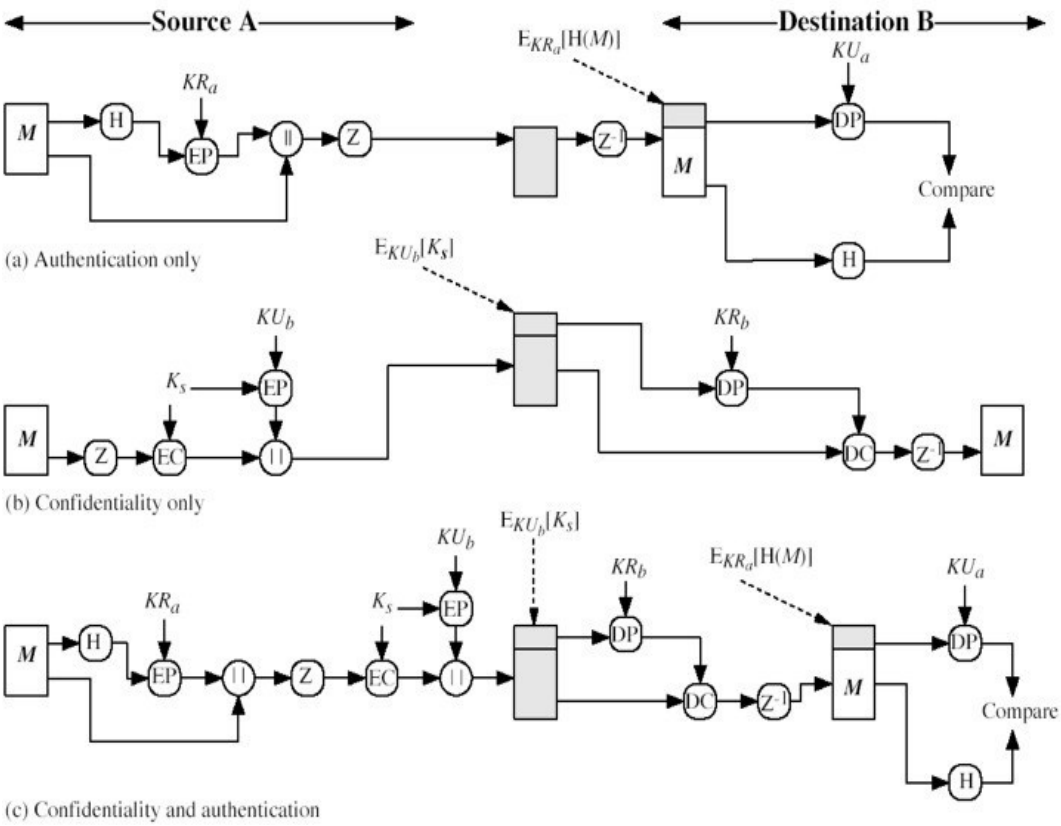


Figure: PGP cryptographic functions.

- The signature is generated before compression for two reasons:
  i. It is preferable to sign an uncompressed message so that it can be used for later verification.
  ii. Different version of PGP produce different compressed forms.
- Message encryption is applied after compression to strengthen cryptographic security. Because the compressed message has less redundancy than the original plaintext, cryptanalysis is more difficult.
- The compression algorithm used is called **ZIP.**

E-mail compatibility :

- ✓ Many electronic mail systems only permit the use of blocks consisting of ASCII text.
- ✓ When PGP is used, at least part of the block to be transmitted is encrypted.
- ✓ To accommodate this restriction PGP uses **radix64** algorithm which maps 6 bits of a binary data into and 8 bit ASCII character.
- ✓ Unfortunately this expands the message by 33% however, with the compression algorithm the overall compression will be about one third (in general).

Segmentation:

- E-mail facilities are often restricted to a maximum message length. For example, many of the facilities accessible throughout the Internet impose a maximum length of 50,000 octets. Any message longer than that must be broken up into smaller segments, each of which is mailed separately.
- To accommodate this restriction, PGP automatically subdivides a large message into segments that are small enough to sent via e-mail.
- The segmentation processing includes the radix-64 conversion. The first segment contains the session key component and signature component that are appear only once at the beginning of.
- At the receiving end, PGP must strip off all e-mail headers and reassemble the entire original block.

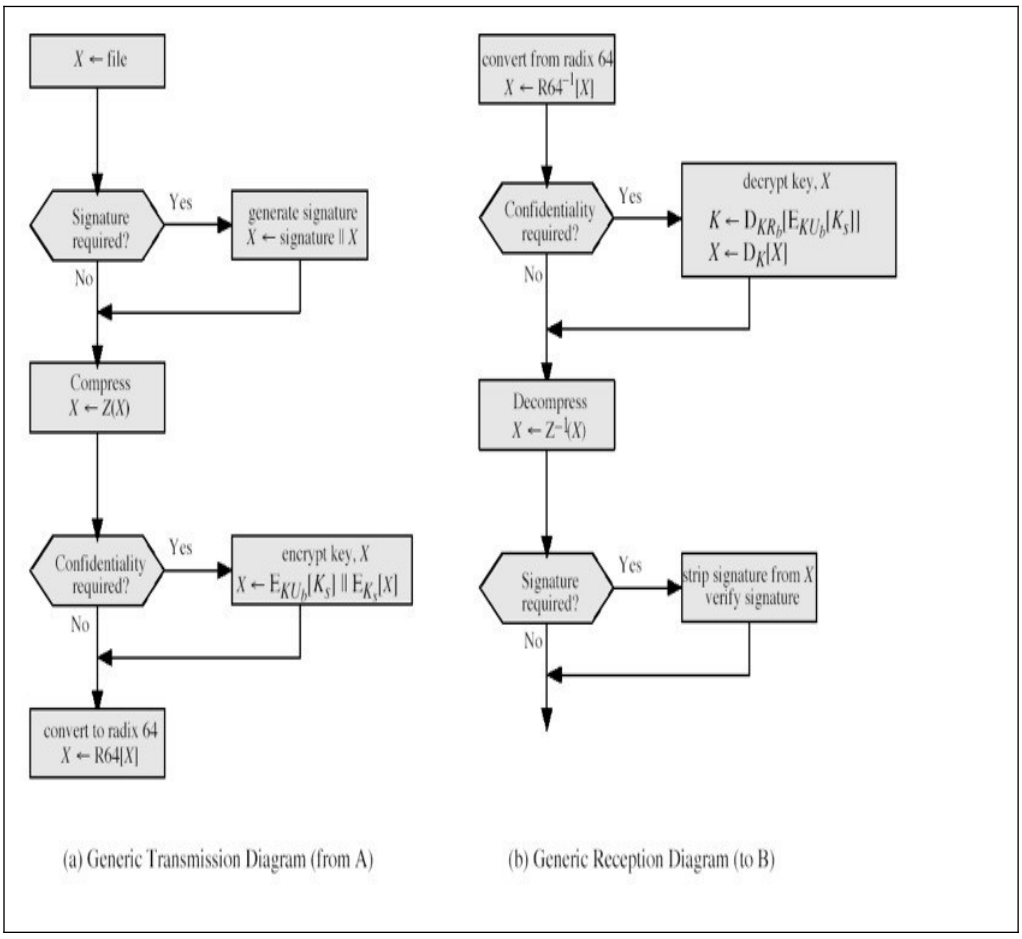


Figure : Transmission and Reception of PGP messages

Cryptographic Keys and Key Rings

PGP makes use of four types of keys:

1. One-time session symmetric keys
2. Public keys
3. Private keys
4. 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. We would like to allow a user to have multiple public-key/private-key pairs. As a result there is not a one-to-one correspondence between users and their public keys. Thus, some means is needed for identifying particular keys.
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.

Session key generation: Each session key is associated with a single message and is used only for the purpose of encryption and decrypting that message. Ex: Assuming it is a 128 bit key that is required, the random 128 bit numbers are generated using CAST-128.

Key Identifiers: It is possible to have more than one public/private key pair per user. Each one therefore needs an ID of some kind. The key ID associated with each public key consists of its least significant 64 bits. That is, the key ID of

public key KU_a is (KU_a mod 2^{64}). This is a sufficient length that the probability of duplicate key IDs is very small. A key ID is also used for the PGP digital signature as the sender may use one of a number of private keys to encrypt the message digest and the recipient must know which one was used.

PGP Message format: A message consists of three components:

1. The message component 2. A signature component,(optional) 3. A session key component (optional).

1. 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.

2. 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. The digest is calculated over the signature timestamp concatenated with the data portion of the message component. The inclusion of the signature timestamp in the digest insures against replay types of attacks.

Leading two octets of message digest: To enables 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.

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. The message component and optional signature component may be compressed using ZIP and may be encrypted using a session key.

3.The session key component includes 1. The session key and 2. The identifier of the recipient's public key

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

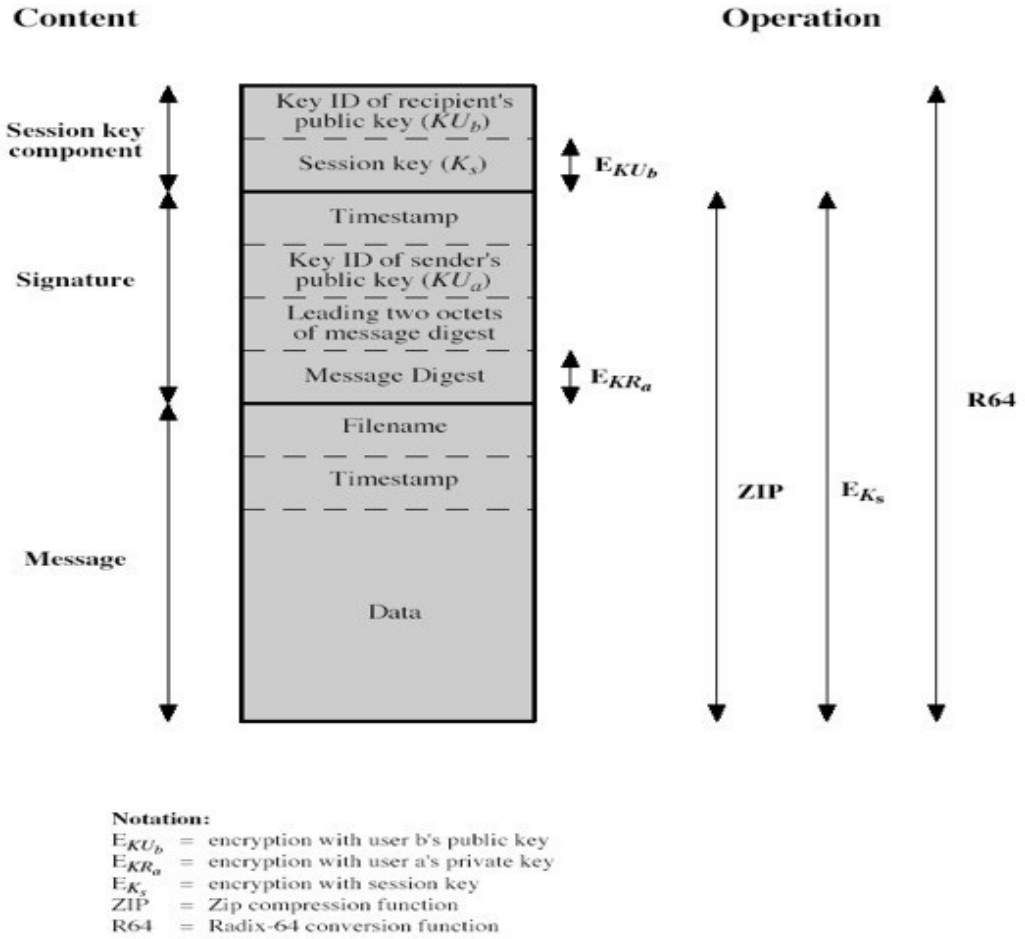


Figure :General format of PGP message (from A to B).

Key Rings :

- ✓ Key IDs are critical to the operation of PGP. From figure it can be seen that two key IDs are included in any PGP message that provides both confidentiality and authentication. These keys need to be stored and organised in a systematic way for efficient and effective use by all parties.
- ✓ The scheme used in PGP is to provide a pair of data structures at each node, one to store the public/private key pairs owned by that node and one to store the public keys of other users known at this node. These data structures are referred to, respectively as the private-key ring and the public key ring.
- ✓ We can view the ring as a table where each row represents one of the public/private key pairs owned by this user. Each row contains the following:

- **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.
- **User ID:** Typically a user's e-mail address.

- ✓ The private key ring can be indexed by either User ID or key ID or both.
- ✓ However for security the value of the key is not stored in the key ring but and encrypted version of it which requires a pass phrase to decrypt.
- ✓ As with all passphrase schemes, the security of this method depends on the strength of the passphrase.

Figure: General structure of private and public-key rings.

Private Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	KU_i	$E_{H(P_i)}[KR_i]$	User i
• • •	• • •	• • •	• • •	• • •

Public Key Ring

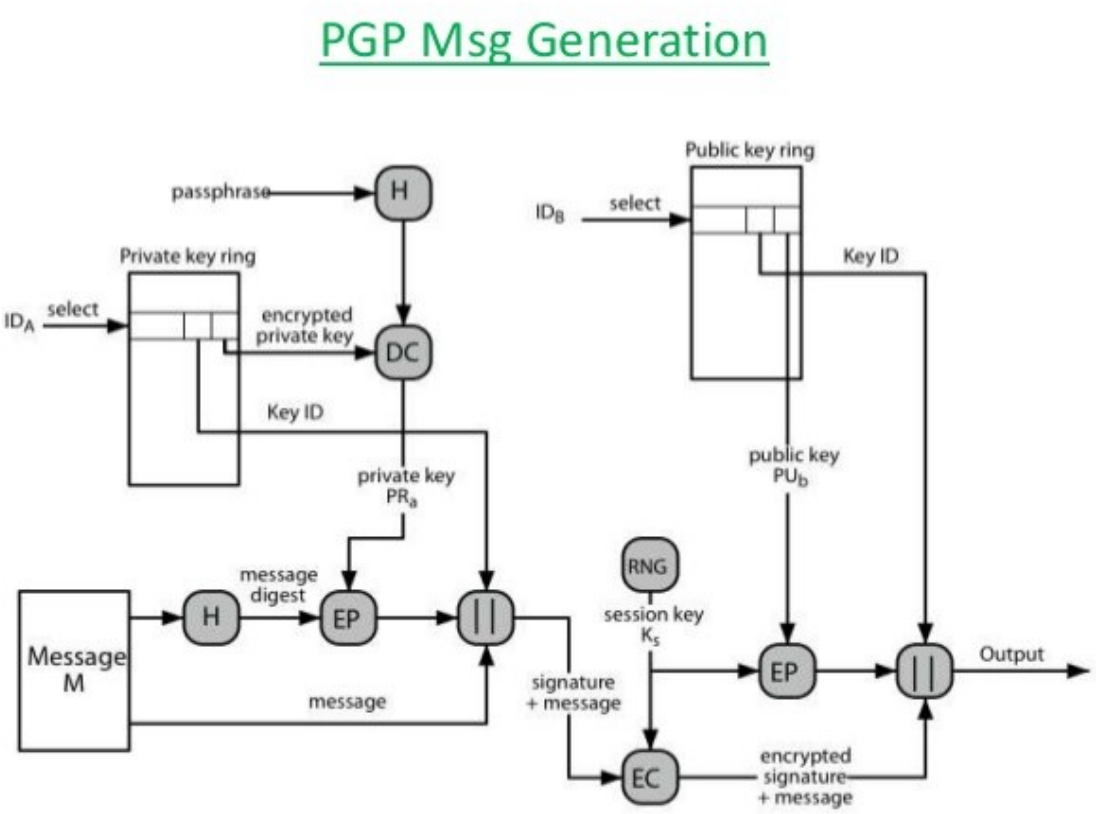
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	$KU_i \bmod 2^{64}$	Ku_i	$trust\_flag_i$	User i	$trust\_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

PGP Message Generation and PGP Message Reception

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

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.

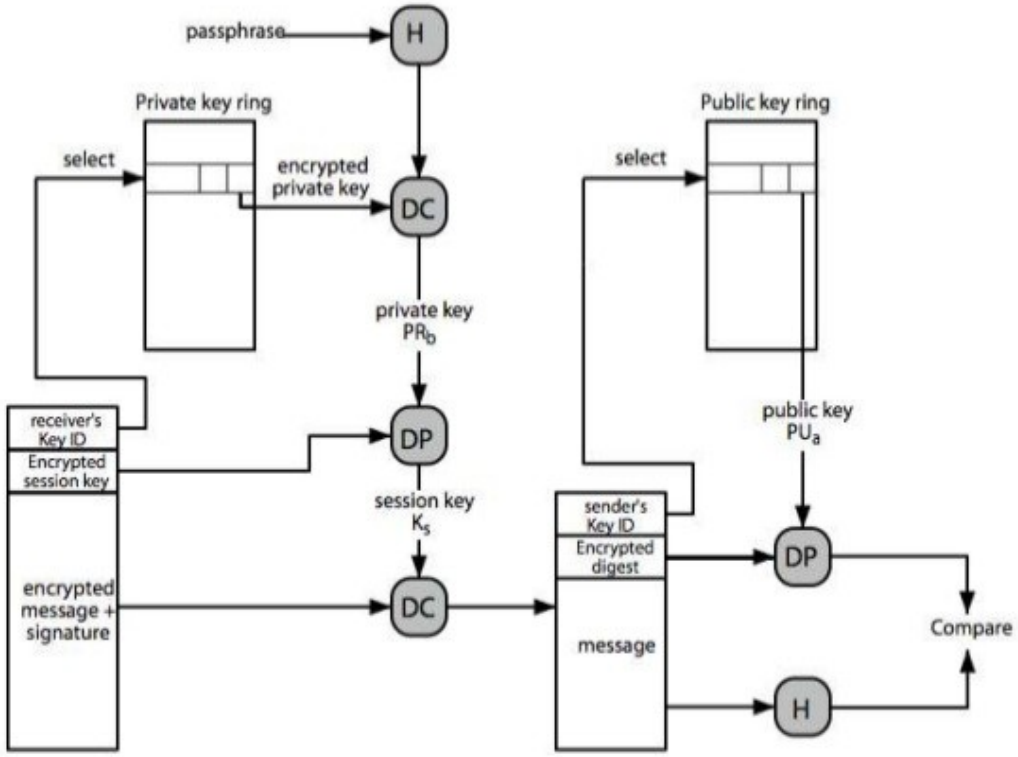
2. Authenticating the message:

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.

PGP Msg Reception



RFC 822

- The Internet RFC 822 specification defines an electronic message format consisting of header fields and an optional message body.
- The header fields contain information about the message, such as the sender, the recipient, and the subject.
- The message body contains actual message . It is separated from the header fields by an empty line (\r\n).

- Example :

```
From: someone@example.com
To: someone_else@example.com
Subject: An RFC 822 formatted message

This is the plain text body of the message.
```

MIME(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.

\

SMTP/RFC822 scheme limitations:

1. SMTP cannot transmit executable files or other binary files.
2. SMTP cannot transmit text data that includes national language characters because these are represented by 8-bit codes with values of 128 decimal or higher, and SMTP is limited to 7-bit ASCII.
3. SMTP servers may reject mail message over a certain size.
4. SMTP gateways that translate between ASCII to EBCDIC suffer translation problems.
5. SMTP gateways to X.400 electronic mail networks cannot handle nontextual data included in X.400 messages.
6. Some SMTP implementations do not adhere completely to the SMTP standard defined in RFC 822.

Common problems include:

- Deletion, addition, or reordering of carriage return and linefeed
- Truncating or wrapping lines longer than 76 characters
- Removal of trailing white space (tab and space characters)
- Padding of lines in a message to the same length
- Conversion of tab characters into multiple space characters

Solution : MIME is intended to resolve these problems

Overview of MIME:

- MIME specification includes the following elements:

1. Five new message header fields. These fields provide information about the body of the message.
2. A number of content formats are defined, thus standardizing representations that supports multimedia e-mail.
3. Transfer encodings are defined that enable that protect any content format to be altered by the mail system.

Five header files defined in MIME are :

1. MIME-Version :
   - Must have the parameter value 1.0
   - This field indicates that the message conforms to RFCs 2045 and 2046
   - There are three versions of S/MIME:
     - S/MIME version **1 (1995)**- was specified and officially published in 1995 by RSA Security, Inc.
     - S/MIME version **2 (1998)**- was specified in a pair of informational RFC documents - RFC 2311 and RFC 2312 - in March1998.
     - The work was continued in the IETF S/MIME Mail Security (SMIME) WG and resulted in S/MIME version **3 (1999)** specified in RFCs 2630 to 2634 in June 1999.
2. 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
3. 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

4. Content-ID :
   - Used to identify MIME entities uniquely in multiple contexts
5. Content-Description :
   - A text description of the object with the body; this is useful when the object is not readable

MIME Content types : (10 marks)

For the **text type** of body, no special software is required to get the full meaning of the text aside from support ofthe indicated character set. The primary subtype is *plain text*, which is simply a string of ASCII characters orISO 8859 characters. The *enriched* subtype allows greater formatting flexibility.

The **multiparttype** indicatesthat the body contains multiple, independent parts. The Content-Type headerfield includes a parameter (called a boundary) that defines the delimiter between body parts. Each boundary starts on a new line and consists of two hyphens followed by the boundary value.

There are four subtypes of the multipart type

The**multipart/mixed subtype** is used when there are multiple independent body parts that need to be bundled. For the **multipart/parallel subtype,** the order of the parts is not significant. If therecipient's system is appropriate, the multiple parts can be presented in parallel. For example, a picture or text part could be accompanied by a voice commentary that is played while the picture or text is displayed.

The **multipart/alternative subtype,** the various parts are different representtations of the same information.

The **multipart/digest subtype** is used when each of the body parts is inter preted as an RFC 5322 messagewith headers

The **message type** provides a number of important capabilities in MIME.

The **message/rfc822subtype** indicates that the body is an entire message, including header and body.

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/externalbody subtype** indicates that the actual data to be conveyed in this message are notcontained 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 uninter- preted binary data or information to be processed by a mail-based application.

Table 18.3 MIME Content Types

Type	Subtype	Description
Text	Plain	Unformatted text; may be ASCII or ISO 8859.
	Enriched	Provides greater format flexibility.
Multipart	Mixed	The different parts are independent but are to be transmitted together. They should be presented to the receiver in the order that they appear in the mail message.
	Parallel	Differs from Mixed only in that no order is defined for delivering the parts to the receiver.
	Alternative	The different parts are alternative versions of the same information. They are ordered in increasing faithfulness to the original, and the recipient's mail system should display the "best" version to the user.
	Digest	Similar to Mixed, but the default type/subtype of each part is message/rfc822.
Message	rfc822	The body is itself an encapsulated message that conforms to RFC 822.
	Partial	Used to allow fragmentation of large mail items, in a way that is transparent to the recipient.
	External-body	Contains a pointer to an object that exists elsewhere.
Image	jpeg	The image is in JPEG format, JFIF encoding.
	gif	The image is in GIF format.
Video	mpeg	MPEG format.
Audio	Basic	Single-channel 8-bit ISDN mu-law encoding at a sample rate of 8 kHz.
Application	PostScript	Adobe Postscript format.
	octet-stream	General binary data consisting of 8-bit bytes.

Example : Content-Type: multipart/signed;
protocol="application/pkcs7-signature";
micalg=sha1; boundary=boundary42
--boundary42
Content-Type: text/plain
This is a clear-signed message.
--boundary42

MIME Transfer Encodings (five marks)

The MIME standard defines two methods of encoding data.

1) The Content- Transfer-Encoding field can actually take on six values, as listed in Table. It indicate that no encoding has been done but provide some information about the nature of the data.

2) Another Content-Transfer-Encoding value is x-token,

7bit	The data are all represented by short lines of ASCII characters.
8bit	The lines are short, but there may be non-ASCII characters (octets with the high-order bit set).
binary	Not only may non-ASCII characters be present but the lines are not necessarily short enough for SMTP transport.
quoted-printable	Encodes the data in such a way that if the data being encoded are mostly ASCII text, the encoded form of the data remains largely recognizable by humans.
base64	Encodes data by mapping 6-bit blocks of input to 8-bit blocks of output, all of which are printable ASCII characters.
x-token	A named nonstandard encoding.

Native Vs Canonical Form (two mark)

1. Canonical form is a format, appropriate to the content type, that is standardized for use between systems. This is in contrast to native form.
2. Native form is a format that may be peculiar to a particular syste006D

Native Form	The body to be transmitted is created in the system's native format. The native character set is used and, where appropriate, local end-of-line conventions are used as well. The body may be a UNIX-style text file, or a Sun raster image, or a VMS indexed file, or audio data in a system-dependent format stored only in memory, or anything else that corresponds to the local model for the representation of some form of information. Fundamentally, the data is created in the "native" form that corresponds to the type specified by the media type.
Canonical Form	The entire body, including "out-of-band" information such as record lengths and possibly file attribute information, is converted to a universal canonical form. The specific media type of the body as well as its associated attributes dictate the nature of the canonical form that is used. Conversion to the proper canonical form may involve character set conversion, transformation of audio data, compression, or various other operations specific to the various media types. If character set conversion is involved, however, care must be taken to understand the semantics of the media type, which may have strong implications for any character set conversion (e.g. with regard to syntactically meaningful characters in a text subtype other than "plain").

S/MIME Functionality(5 mark)

In terms of general functionality, S/MIME is very similar 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 encrypted-content 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.

2. Cryptographic Algorithms :S/MIME uses the following terminology, taken from RFC 2119 to specify the requirement level:

- **Must:** The definition is an absolute requirement of the specification. An implementation must include this feature or function to be in conformance with the specification.

- **Should:** There may exist valid reasons in particular circumstances to ignore this feature or function, but it is recommended that an implementation include the feature or function.

Cryptographic Algorithms Used in S/MIME	
Function	**Requirement**
Create a message digest to be used in forming a digital signature. Encrypt message digest to form digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility. Sending and receiving agents MUST support DSS. Sending agents SHOULD support RSA encryption. Receiving agents SHOULD support verification of RSA signatures with key sizes 512 bits to 1024 bits.
Encrypt session key for transmission with message.	Sending and receiving agents SHOULD support Diffie-Hellman. Sending and receiving agents MUST support RSA encryption with key sizes 512 bits to 1024 bits.
Encrypt message for transmission with one-time session key.	Sending and receiving agents MUST support encryption with triple DES Sending agents SHOULD support encryption with AES. Sending agents SHOULD support encryption with RC2/40.
Create a message authentication code	Receiving agents MUST support HMAC with SHA-1. Receiving agents SHOULD support HMAC with SHA-1.

S/MIME Messages

S/MIME makes use of a number of new MIME content types. All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories.

Type	Subtype	S/MIME parameter	Description

Multipart	Signed		A clear message in tow parts: One is the message and the other is the signature.
Application	pkcs7-mime pkcs7-mime Pkcs7-signature pkcs10-mime	signedData envelopedData -- --	A signed S/MIE entity. An encrypted S/MIME entity. multipart/signed message. A certificate registration request message.

Securing a MIME Entity:

- S/MIME secures a MIME entity with a signature, encryption, or both
- The MIME entity is prepared according to the normal rules for MIME message preparation
    - The MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object
    - A PKCS object is then treated as message content and wrapped in MIME
- In all cases the message to be sent is converted to canonical form

Enveloped Data :

The steps for preparing an envelopedData MIME entity are

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES).
2. For each recipient, encrypt the session key with the recipient's public RSA key.
3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate, an identifier of the algorithm used to encrypt the session key, and the encrypted session key.
4. Encrypt the message content with the session key.

Working ;The RecipientInfo blocks followed by the encrypted content constitute the envelopedData . This information is then encoded into base64.

To recover the encrypted message, the recipient first strips off the base64 encoding.Then the recipient's private key is used to recover the session key. Finally,the message content is decrypted with the session key.

SignedData :

The signedData smime-type can be used with one or more signers.

The steps for preparing a signedData MIME entity are

1. Select a message digest algorithm (SHA or MD5).
2. Compute the message digest (hash function) of the content to be signed.
3. Encrypt the message digest with the signer's private key.
4. Prepare a block known as SignerInfo that contains the signer's public-key

Registration Request : Typically, an application or user will apply to a certification authority for a public-key certificate. The application/pkcs10 S/MIME entity is used to transfer a certification request. The certification request includes certification RequestInfo block, followed by an identifier of the public-key encryption algorithm. The certificationRequestInfo block includes a name of the certificate subject (the entity whose public key is to be certified) and a bit-string representation of the user's public key.

Certificates-Only Message

A message containing only certificates or a certificaterevocation list (CRL) canbe sent in response to a registration reques t.

CERTIFICATE PROCESSING

Certificate: An identifier of the message digest algorithm, an identifier of the algorithm used to encrypt the message digest, and the encrypted message digest.

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

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

S/MIME Certificate Processing : S/MIME uses public-key certificates that conform to version 3 of X.509.

USER AGENT ROLE An S/MIME user has several key-management functions to perform.

- **Key generation:** The user of some related administrative utility . Each key pair MUST be generated from a good source of non- deterministic random input and be protected in a secure fashion. A user agent SHOULD generate RSA keypairs with a length in the range of 768 to 1024 bits
- **Registration:** A user's public key must be registeredwith a certification authority in order to receive an X.509 public-key certificate.
- **Certificate storage and retrieval:** A user requires access to a local list of certificates in order to veri fyincoming signatures and to encrypt outgoing messages. Such a list could be maintained by the user or by some local administrative entity on behalf of a number of users.

Example ;

- ✓ Verisign one of most widely used
- ✓ Verisign issues several types of Digital IDs

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

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

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

- Address
- E-mail address

- Basic registration information (country, zip code, age, and gender)

Classes used :

- For Class 1 Digital IDs, VeriSign confirms the user's e-mail address by sending a PIN and Digital ID pick-up information to the e-mail address provided in the application.
- For Class 2 Digital IDs, VeriSign verifies the information in the application through an automated comparison with a consumer database in addition to

Enhanced Security Services (two marks) : The three services are

a. **Signed receipts:** A signed receipt may be requested in a SignedData object.The recipient signs t he entire
original message+the original (sender's) signature and appends the new signature to form a new S/MIME message.

b. **Security labels:** A security label may be included in the authenticated attributes of a SignedData object. A security label is a set of security informationregarding the sensitivity of the content that is protected by S/MIME encapsulation. It include priority (secret, confidential, restricted, and so on)

c. **Secure mailing lists:** When a user sends a message to multiple recipients, a certain amount of per-recipient processing is required. The user can be relieved of this work by employing the services of an S/MIME Mail List Agent (MLA). An MLA can take a single incoming message, perform the recipient-specific encryption for each recipient, and forward the message. The originator of a message need only send the message to the MLA with encryption performed using the MLA's public key.