

CHAPTER - 7

ELECTRONIC MAIL SECURITY



7.1 PRETTY GOOD PRIVACY

○ KEY POINTS

- PGP is an open-source, freely available software package for e-mail security.
- It provides authentication through the use of digital signature, confidentiality through the use of symmetric block encryption, compression using the ZIP algorithm, and e-mail compatibility using the radix-64 encoding scheme.
- PGP incorporates tools for developing a public-key trust model and public-key certificate management.
- S/MIME is an Internet standard approach to e-mail security that incorporates the same functionality as PGP.
- DKIM is a specification used by e-mail providers for cryptographically signing e-mail messages on behalf of the source domain.

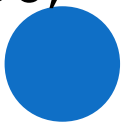


7.1 PRETTY GOOD PRIVACY

- PGP is 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:
 - Selected the best available cryptographic algorithms
 - Integrated these algorithms into a general-purpose application
 - Made the package and its documentation, including the source code, freely available
 - Entered into an agreement with a company to provide a fully compatible, low-cost commercial version of PGP



CONTINUE...

- PGP has grown explosively and is now widely used. A number of reasons can be cited for this growth
 - It is available free worldwide and is used on all platforms. The commercial version comes with a vendor support.
 - It is based on algorithms that are extremely secure and are publicly reviewed. Like RSA, DSS, Diffie-Hellman, etc for Public key encryption and CAST-128, IDEA, and 3DES for symmetric encryption; and SHA-1 for hash coding.
 - It has a wide range of applicability from corporations to individuals wanting to communicate worldwide securely.
 - It was not developed by, nor is it controlled by, any governmental or standards organization
 - PGP is now on an Internet standards track (RFC 3156; *MIME Security with OpenPGP*)
- 

CONTINUE...

○ Notation

- K= session key used in symmetric encryption scheme
- PRa= private key of user A, used in public-key encryption scheme
- PUB= 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



❑ OPERATIONAL DESCRIPTION

- PGP provide following services
 - Authentication
 - Confidentiality
 - Compression
 - e-mail compatibility



□ Authentication

○ The sequence 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.



□ Authentication

Table 7.1 Summary of PGP Services

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.
E-mail compatibility	Radix-64 conversion	To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.

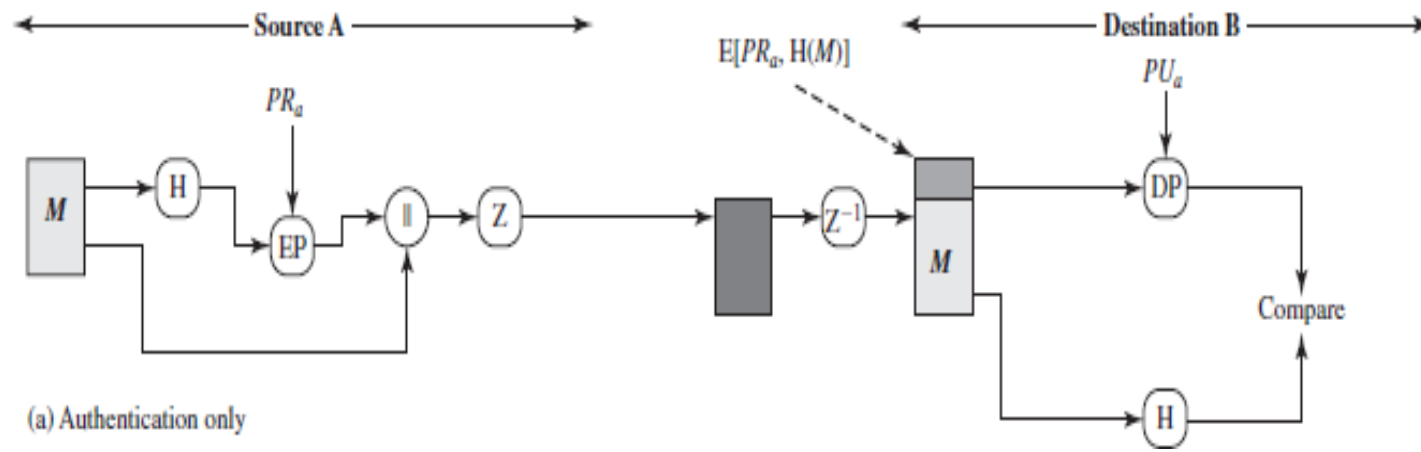


□ Authentication

- The combination of SHA-1 and RSA provides an effective digital signature scheme.
- Because of the strength of RSA, the recipient is assured
 - that only the possessor of the matching private key can generate the signature.
 - And that, 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, signatures can be generated using DSS/SHA-1.
- Signature are appended with the messages in most of the cases but sometimes. E.g. when more than one parties need to sign a contract, a detached signature is made and trasnmitted separately.



❑ *AUTHENTICATION*



□ Confidentiality

- Confidentiality is provided by encrypting messages to be transmitted or to be stored locally as files.
- In both cases, the symmetric encryption algorithm CAST-128 may be used.
- Alternatively, IDEA or 3DES may be used. The 64-bit cipher feedback (CFB) mode is used.
- **Key Distribution**
- A new symmetric key is generated as a random 128-bit number for each message, also called session key or one-time key.
- Session key is bounded with the message and transmitted. Its secured by receiver's public key.

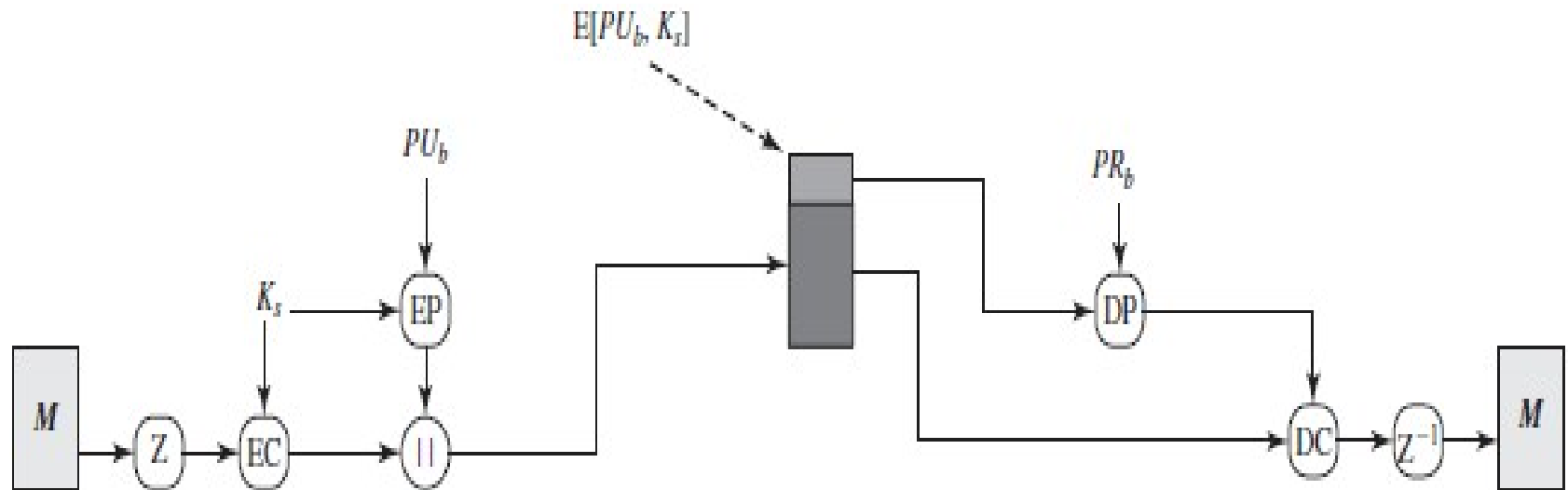


□ Confidentiality

- Following figure illustrates the sequence, which can be described as follows.
- 1. The sender generates a message and a random 128-bit number to be used as a session key for this message only.
- 2. The message is encrypted using CAST-128 (or IDEA or 3DES) with the session key.
- 3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.
- 4. The receiver uses RSA with its private key to decrypt and recover the session key.
- 5. The session key is used to decrypt the message.
- As an alternative to the use of RSA for key encryption, PGP provides an option referred to as Diffie-Hellman.



❑ **CONFIDENTIALITY**



(b) Confidentiality only

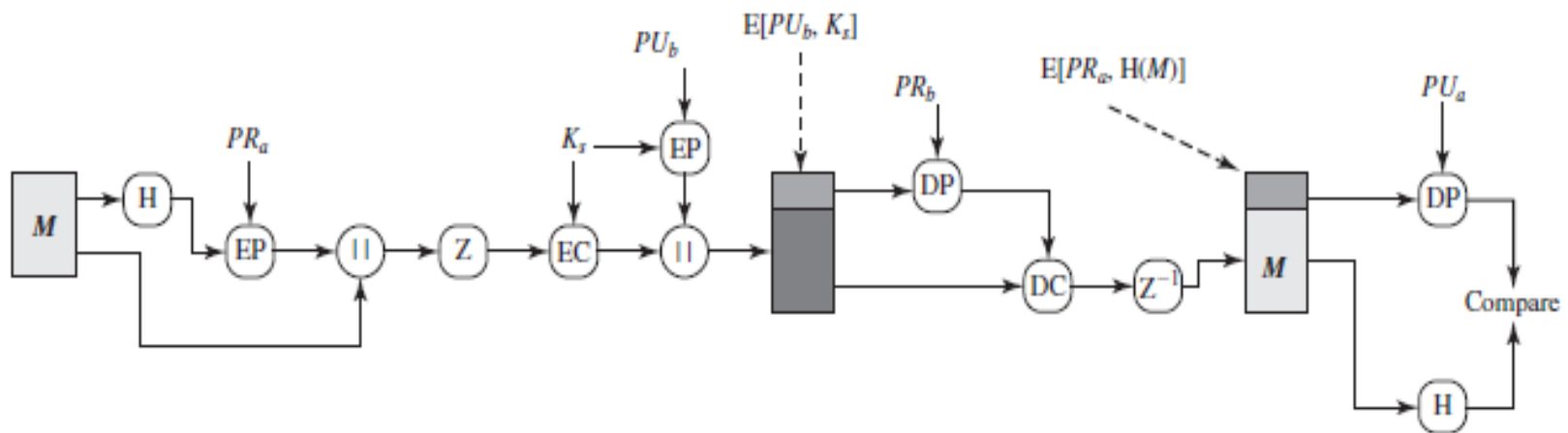


□ Confidentiality and Authentication

- 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 plain-text message plus signature is encrypted using CAST-128 (or IDEA or 3DES), and the session key is encrypted using RSA (or ElGamal).
- Alternatively, the message can be encrypted first and then signature can be generated but note preferable if third party verification is to be done.



CONFIDENTIALITY AND AUTHENTICATION



(c) Confidentiality and authentication



□ COMPRESSION

- PGP compresses the message after applying the signature but before encryption which saves space both for e-mail transmission and for file storage.
- The placement of the compression algorithm, indicated by Z for compression and Z^{-1} for decompression is critical as shown in previous figures.
- There are two ways for applying compression as follows.



❑ COMPRESSION

○ 1. Generating signature **before compression**:

○ Reason 1

- It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.
- If one signed a compressed document, then it would be necessary either to store a compressed version of the message for later verification or to recompress the message when verification is required.



❑ COMPRESSION

- Reason 2
- PGP's compression algorithm presents difficulty in generating re-compressed message for verification dynamically.
- The algorithm is not deterministic; various implementations of the algorithm achieve different tradeoffs in running speed versus compression ratio and, as a result, produce different compressed forms.
- Compression algorithms are interoperable, any version of the algorithm can correctly decompress the output of any other version.
- Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm.



❑ COMPRESSION

2. Message encryption is applied **after compression** to strengthen cryptographic security.

Because the compressed message has less redundancy than the original plain-text, crypt-analysis is more difficult.

The compression algorithm used is ZIP



□ E-MAIL COMPATIBILITY

- Different services require different type of encryption mechanisms with resulting block consists of a stream of arbitrary 8-bit octets.
- But many electronic mail systems only permit the use of blocks consisting of ASCII text.
- For that, PGP provides the service of converting the raw 8-bit binary stream to a stream of printable ASCII characters, called 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.



❑ E-MAIL COMPATIBILITY

- The use of radix-64 expands a message by 33%.
- The session key and signature portions of the message are relatively compact, and the plaintext message has been compressed compensating radix-64 expansion for .
- In fact, the compression should be more than enough to compensate for the radix-64 expansion.
- Radix-64 blindly converts the input stream to radix-64 format regardless of content even if the input happens to be ASCII text.

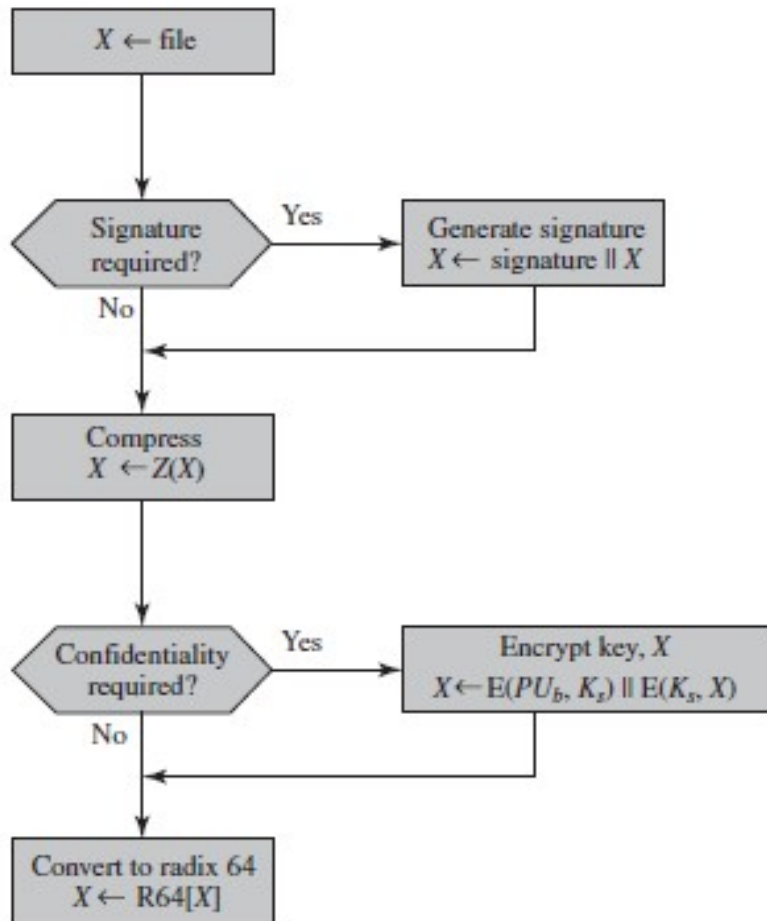


□ E-MAIL COMPATIBILITY

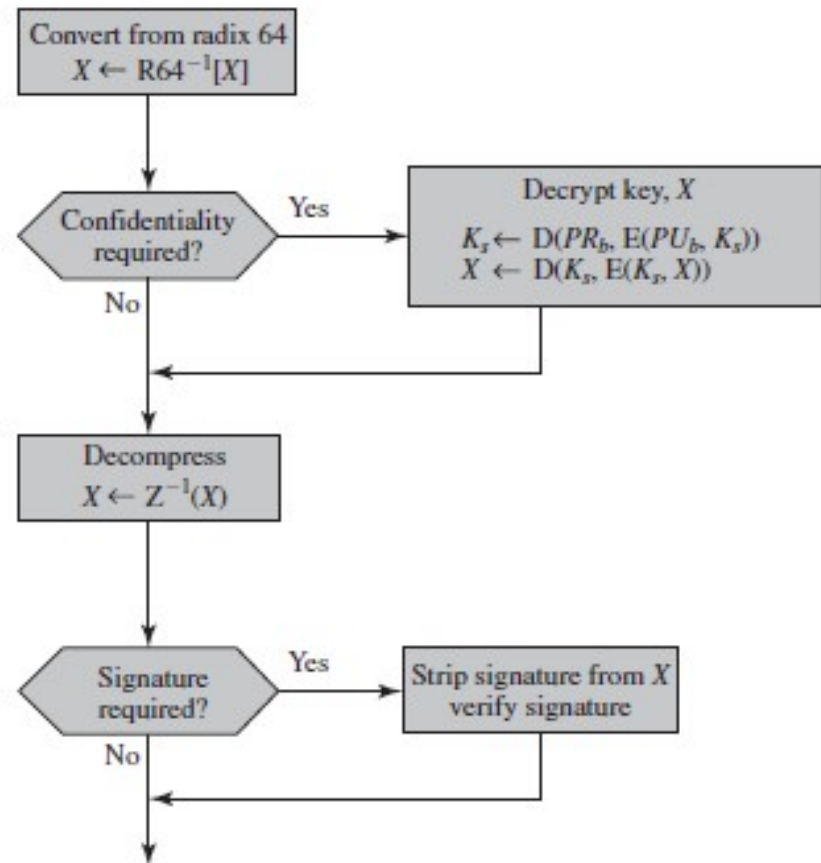
- A signed message which is not encrypted if converted to radix-64, the o/p will be unreadable. (But it gives certain level of confidentiality)
- Thus, PGP can be configured to convert only the signature portion of the message.
- So human recipient can read it without using PGP, as its used for signature only.
- Following Figure 7.2 shows the relationship among the four services so far discussed.



□ E-MAIL COMPATIBILITY



(a) Generic transmission diagram (from A)



(b) Generic reception diagram (to B)

Figure 7.2 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.
 - Generating unpredictable session keys
 - Allow a user to have multiple public-key/private-key pairs, so that it can be changed time to time.
 - Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents



1. SESSION KEY GENERATION

- For each message a unique session key is used for encryption and decryption with symmetric key algorithms like CAST-128 and IDEA (128-bits) and 3DES (164-bits)



1. SESSION KEY GENERATION

- The input to the random number generator consists of a 128-bit key and two 64-bit blocks that are treated as plaintext to be encrypted. Using cipher feedback mode, the CAST-128 encrypter produces two 64-bit cipher text blocks, which are concatenated to form the 128-bit session key.
- The plain text input here is derived from a stream of 128-bit randomized numbers based on keystroke i/p from the user.
- The random i/p is also combined with previous session key o/p to form the key i/p to the generator giving effectively unpredictable session keys.



2. KEY IDENTIFIERS

- We know that an encrypted message is prepended by a session key which is encrypted by recipients public key.
- If each user employed a single public/private key pair, he/she will automatically know which key to use to decrypt the session key.
- But as per our requirement stated, any given user may have multiple public/private keys.
- So how the user know which key has been applied? So we can also transmit public key along with the message. But it may waste space during transmission.
- E.g. RSA public key may be hundreds of digits in length.



2. KEY IDENTIFIERS

- Another solution is to have identifier for each public key, and then a much shorter ID only would need to be transmitted.
- This solution will generate an overhead of maintaining key IDs on both sender and receiver sides.
- The solution adopted by PGP is to assign a key ID to each public key that is, with very high probability, unique within a user ID.
- The key ID associated with each public key consists of its least significant 64 bits. That is, the key ID of public key PUa is $(PUa \bmod 2^{64})$



2. KEY IDENTIFIERS

- Now that the concept of key ID has been introduced, we can take a more detailed look at the format of a transmitted message, which is shown in Figure 7.3.
- 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.



2. KEY IDENTIFIERS

- 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 timestamp insures against the replay types of attacks



2. KEY IDENTIFIERS

- **Leading two octets of message digest:** 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.

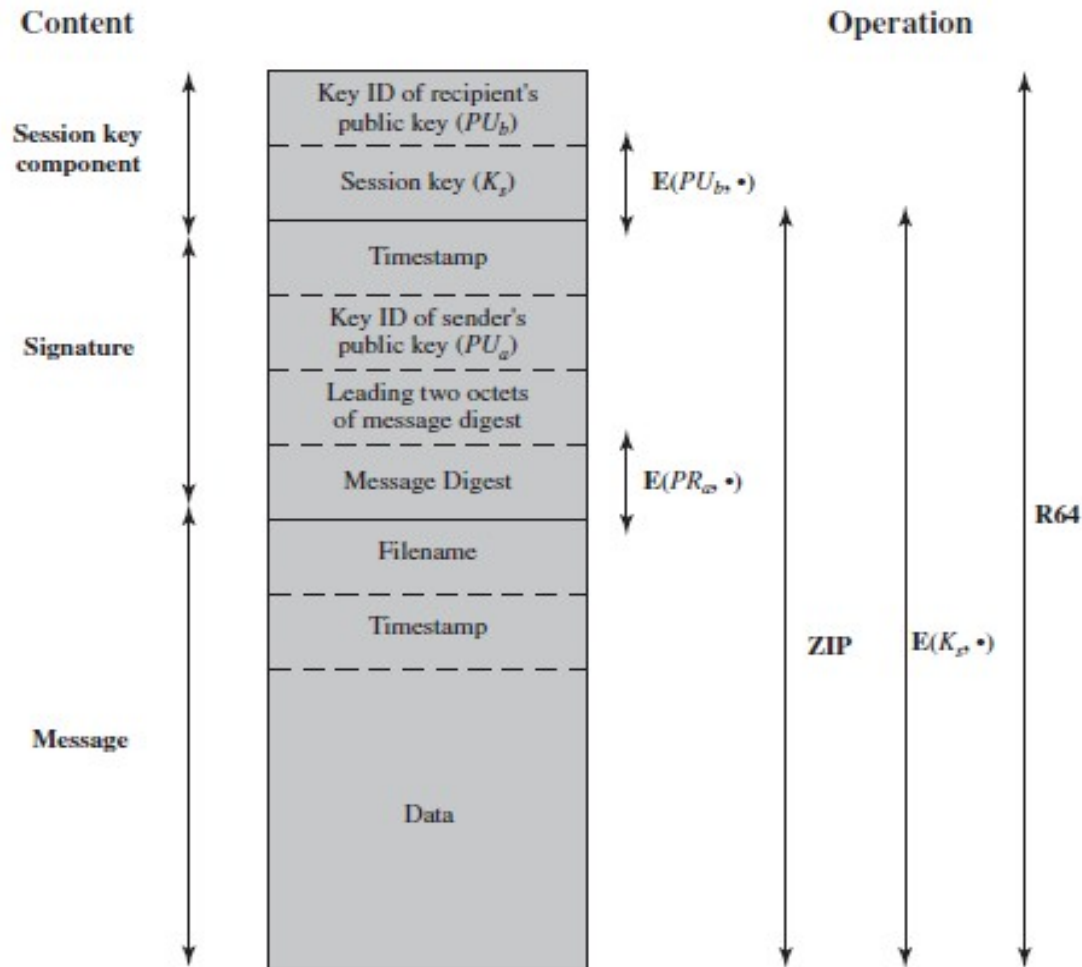


2. KEY IDENTIFIERS

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



□ GENERAL FORMAT PGP MESSAGE



Notation:

$E(PU_b, \bullet)$ = encryption with user b's public key

$E(PR_a, \bullet)$ = encryption with user a's private key

$E(K_s, \bullet)$ = encryption with session key

ZIP = Zip compression function

R64 = Radix-64 conversion function

Figure 7.3 General Format PGP Message (from A to B)

3. KEY RINGS

- Key IDs are critical to the operation of PGP are included in any PGP message that provides both confidentiality and authentication.
- These keys need to be stored and organized 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**



3. KEY RINGS

- Figure 7.4 shows the general structure of a private-key ring. We can view the ring as a table in which each row represents one of the public/private key pairs owned by this user. Each row contains the 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:** This will be the user's e-mail address. The user may also choose to associate a different name with each pair or to reuse the same User ID more than once.



CONTINUE...

Private-Key Ring

Timestamp	Key ID*	Public Key	Encrypted Private Key	User ID*
• • •	• • •	• • •	• • •	• • •
T_i	$PU_i \bmod 2^{64}$	PU_i	$E(H(P_i), PR_i)$	User i
• • •	• • •	• • •	• • •	• • •

- The private-key ring can be indexed by either User ID or Key ID.
- It is intended that the private-key ring be stored only on the machine of the user or the one who created it.

CONTINUE...

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



CONTINUE...

- 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.



CONTINUE...

- Following Figure also shows the general structure of a public-key ring 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.
- The public-key ring can be indexed by either User ID or Key ID or sometimes both.



CONTINUE...

Public-Key Ring

Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
T_i	$PU_i \bmod 2^{64}$	PU_i	trust_flag_i	$\text{User } i$	trust_flag_i		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•



CONTINUE...

- **Key Legitimacy Field** indicates that the extent to which PGP will trust that this is a valid public key for this user
- **Signature Trust Field** that indicates the degree to which this PGP user trusts the signer to certify public keys
- **Owner Trust Field** is included that indicates the degree to which this public key is trusted to sign other public-key certificates



How these key rings are used in message transmission and reception?

- Consider following figure. For simplicity we ignore compression and radix-64 conversion.
- 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.



How these key rings are used in message transmission and reception?

○ **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.



How these key rings are used in message transmission and reception?

○ **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 MESSAGE GENERATION

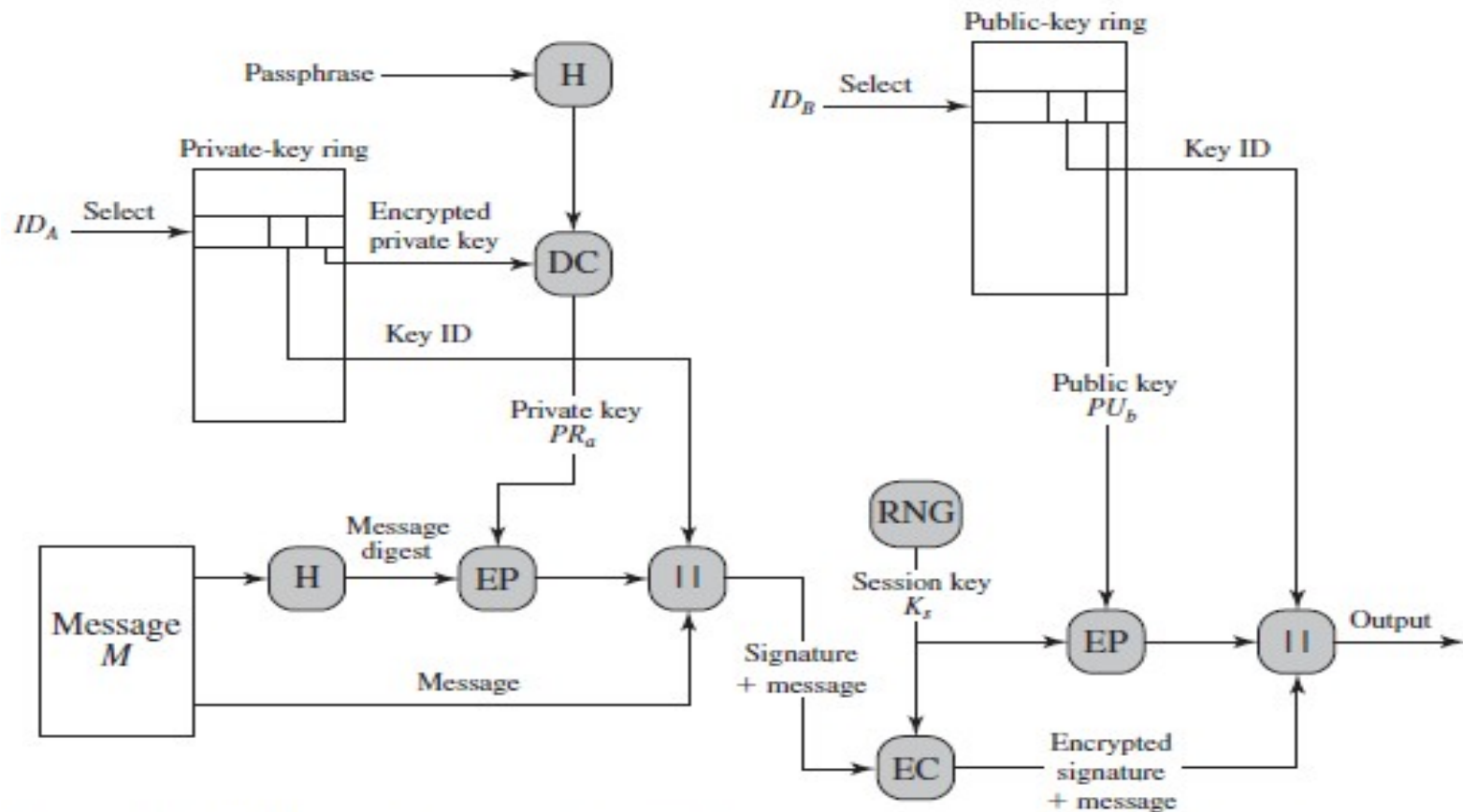


Figure 7.5 PGP Message Generation (from User A to User B: no compression or radix-64 conversion)

PGP MESSAGE RECEPTION

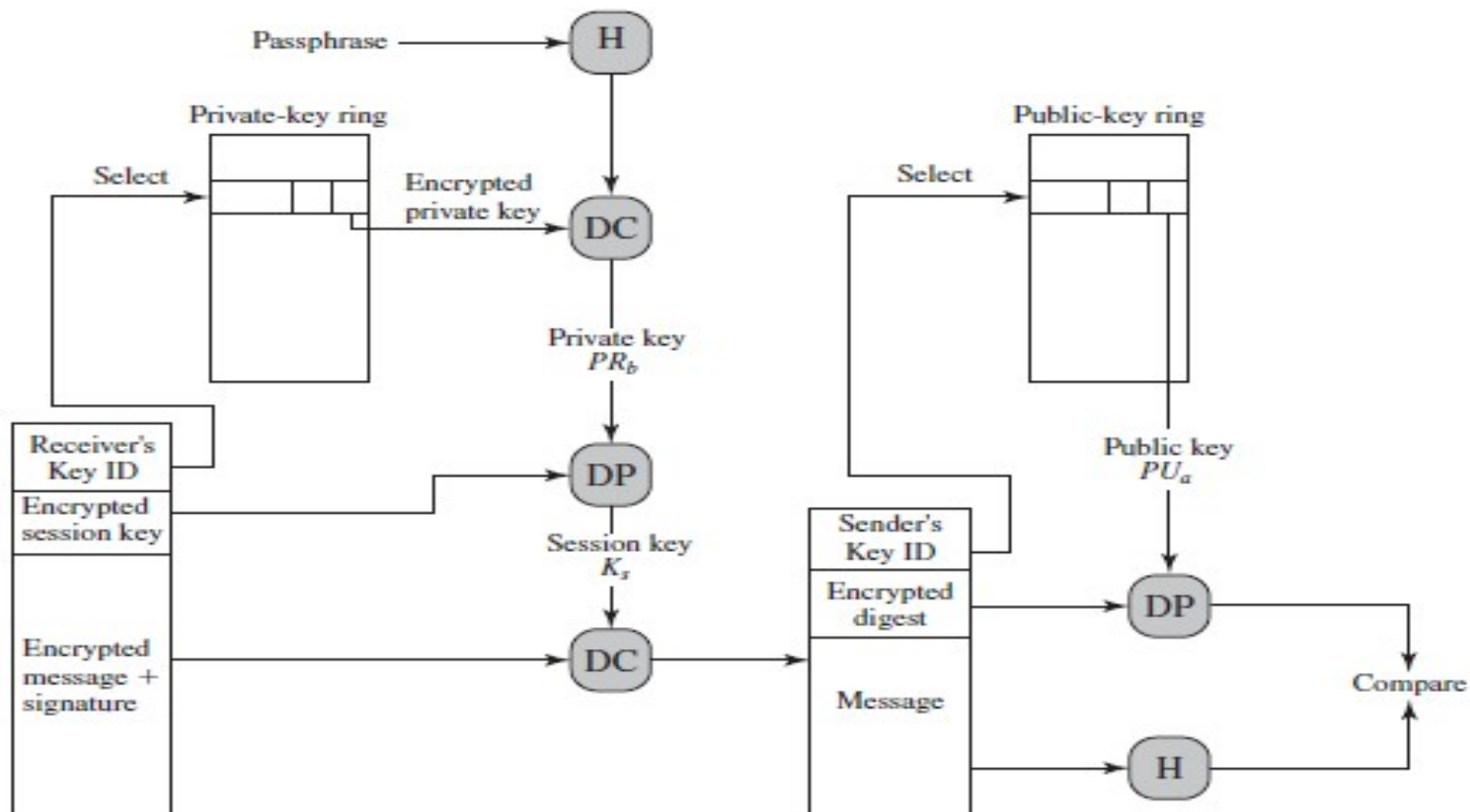


Figure 7.6 PGP Message Reception (from User A to User B; no compression or radix-64 conversion)

Public-Key Management

- PGP provides provide an effective confidentiality and authentication service.
- One final area needs to be addressed, is of public-key management.
- Business of protecting public keys from tampering is the single most difficult problem in practical public key applications.
- Why Key Management is needed?



Public-Key Management

- Suppose that A's key ring contains a public key attributed to B, but in fact the key is owned by C.
- This could happen, for example, if A got the key from a bulletin board system (BBS) that was used by B to post the public key but that has been compromised by C.
- The result is that two threats now exist.
 - First, C can send messages to A and forge B's signature so that A will accept the message as coming from B.
 - Second, any encrypted message from A to B can be read by C.
- A number of approaches are there to make sure that a user's public-key ring does not contain false public keys. Suppose that A wishes to obtain a reliable public key.



Public-Key Management

1. Physically get the key from B. B could store her public key on a floppy disk and hand it to A.
2. This method has though some practical limitations.
3. Verify a key by telephone. If A can recognize B on the phone, A could call B and ask her to dictate the key, in radix-64 format, over the phone.
4. Obtain B's public key from a mutual trusted individual D.
5. Obtain B's public key from a trusted certifying authority



THE USE OF TRUST

- PGP provide a convenient means of using trust, associating trust with public keys, and exploiting trust information.
- Each entry in the public-key ring is a public-key certificate
- For each entry there is a **key legitimacy field** that indicates the extent to which PGP will trust that its a valid public key for this user; the higher the level of trust, the stronger is the binding of this user ID to this key.
- This field is computed by PGP.
- Each signature has associated with it a **signature trust field** that indicates the degree to which this PGP user trusts the signer to certify public keys



THE USE OF TRUST

- Figure 7.7 provides an example of the way in which signature trust and key legitimacy are related along with the structure of a public-key ring.
- The user has acquired a number of public keys—some directly from their owners and some from a third party such as a key server.
- The node labeled “You” refers to the entry in the public-key ring corresponding to this user.
- This user has specified that it always trusts the following users to sign other keys: D, E, F, L. This user partially trusts users A and B to sign other keys.



THE USE OF TRUST

- So the shading, or lack thereof, indicates the level of trust assigned by this user.
- The tree structure indicates which keys have been signed by which other users.
- If a key is signed by a user whose key is also in this key ring, the arrow joins the signed key to the signatory.
- If the key is signed by a user whose key is not present in this key ring, the arrow joins the signed key to a question mark, indicating that the signatory is unknown to this user.



THE USE OF TRUST

- 1. Note that all keys whose owners are fully or partially trusted by this user have been signed by this user, with the exception of node L.
- 2. We assume that two partially trusted signatures are sufficient to certify a key. Hence, the key for user H is deemed legitimate by PGP because it is signed by A and B, both of whom are partially trusted.



THE USE OF TRUST

- 3. A key may be determined to be legitimate because it is signed by one fully trusted or two partially trusted signatories, but its user may not be trusted to sign other keys.
- If you wish to send a private message to some individual, it is not necessary that you trust that individual in any respect. It is only necessary that you are sure that you have the correct public key for that individual.
- 4. Figure 7.7 also shows an example of a detached “orphan” node S, with two unknown signatures. Such a key may have been acquired from a key server.



PGP TRUST MODEL EXAMPLE

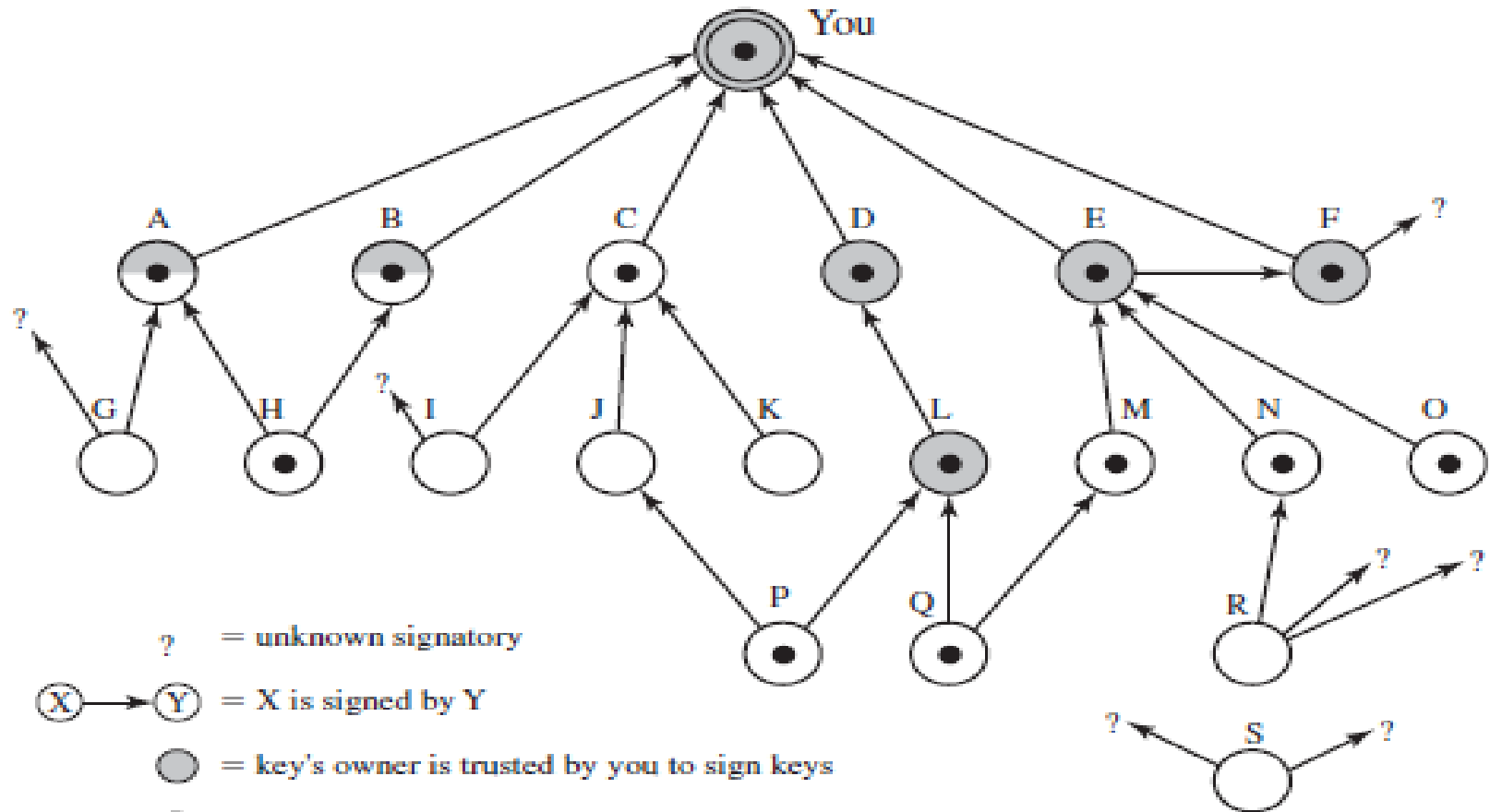


Figure 7.7 PGP Trust Model Example

REVOKING PUBLIC KEYS

- A user may wish to revoke his or her current public key either because compromise is suspected or simply to avoid the use of the same key for an extended period.
- A compromise means an opponent somehow had obtained a copy of your unencrypted private key or both the private key from your private-key ring and your passphrase.
- Owner needs to issue a key revocation certificate, signed by the owner with the same form as a normal signature certificate but includes an indicator that the purpose of this certificate is to revoke the use of this public key.
- The owner disseminates this certificate as widely and as quickly as possible to enable potential correspondents to update their public-key rings.



7.2 S/MIME

- Secure/Multipurpose Internet Mail Extension (S/MIME) is a security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security
- It appears likely that S/MIME will emerge as the industry standard for commercial and organizational use, while PGP will remain the choice for personal e-mail security for many users.



7.2 S/MIME

- S/MIME is defined in a number of documents—most importantly RFCs 3370, 3850, 3851, and 3852..
- To understand S/MIME, we need first to have a general understanding of the underlying e-mail format that it uses, namely MIME.
- We need to go back to the traditional e-mail format standard, RFC 822. The most recent version of this format specification is RFC 5322 (Internet Message Format).



□ RFC 5322

- RFC 5322 defines a format for text messages that are sent using electronic mail
- In the RFC 5322 context, messages are viewed as having an envelope and contents.
- The envelope contains whatever information is needed to accomplish transmission and delivery.
- The contents compose the object to be delivered to the recipient
- The RFC 5322 standard applies only to the contents that includes header field to create envelope and the standard is intended to facilitate the acquisition of such information by programs.



□ RFC 5322

- 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.
- The most frequently used keywords in the header are From, To, Subject, and Date.
- Another field that is commonly found in RFC 5322 headers is Message-ID, a unique identifier associated with this message.



□ **EXAMPLE MESSAGE**

Date: October 8, 2009 2:15:49 PM EDT

From: " William Stallings " <ws@shore.net>

Subject: The Syntax in RFC 5322

To: Smith@Other-host.com

Cc: Jones@Yet-Another-Host.com

Hello. This section begins the actual message body, which is delimited from the message heading by a blank line.



❑ **MULTIPURPOSE INTERNET MAIL EXTENSIONS**

- Multipurpose Internet Mail Extension (MIME) is an extension to the RFC 5322 framework intended to address some of the problems and limitations of the Simple Mail Transfer Protocol (SMTP)



❏ MULTIPURPOSE INTERNET MAIL EXTENSIONS

○ Following list the limitations of the SMTP/5322 scheme

- SMTP cannot transmit executable files/binary objects and 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.)
- Translation problems
- SMTP servers may reject mail message over a certain size
- SMTP gateways cannot handle non-textual data
- 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
 - Padding of lines in a message to the same length
 - Conversion of tab characters into multiple space characters



❏ MULTIPURPOSE INTERNET MAIL EXTENSIONS

- MIME is intended to resolve these problems in a manner that is compatible with existing RFC 5322 implementations.
- The specification is provided in RFCs 2045 through 2049.
- **OVERVIEW** The MIME specification includes the following elements.
 - 1. Five new message header fields are defined, which may be included in an RFC 5322 header. These fields 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 enable the conversion of any content format into a form that is protected from alteration by the mail system.



1. HEADER FIELDS

The five header fields defined in MIME are:

- MIME-Version: must have parameter value 1.0. It also indicates that message conforms to RFC 2045 to 2046.
- Content-Type: Describes the data contained in the body with sufficient detail that the receiving user agent can deal with the data in an appropriate manner.
- Content-Transfer-Encoding: type of transformation used to represent the body of the message acceptable for mail transport.
- Content-ID: 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)



HEADER FIELDS

- RFC 5322 compliant implementation must support the MIME-Version, Content-Type, and Content-Transfer-Encoding fields; the Content-ID and Content-Description fields are optional.



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.

Table 7.3 MIME Content Types

3. MIME TRANSFER ENCODINGS

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.



❑ S/MIME FUNCTIONALITY

- In terms of general functionality, S/MIME is very similar to PGP. Both offer the ability to sign and/or encrypt messages.
- Enveloped data: encrypted content of any type and encrypted content encryption keys for one or more recipients
- Signed data: digital signature formed taking the message digest of the content to be signed and then encrypting it 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.



❑ S/MIME FUNCTIONALITY

- Clear-signed data: a digital signature of the content is formed and is encoded using base64. 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.



❏ ***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***

- S/MIME incorporates three public-key algorithms.
- The **Digital Signature Standard (DSS)** for digital signature.
- S/MIME lists Diffie-Hellman algorithm for encrypting session keys; in fact, a **variant of Diffie-Hellman** that does provide encryption/decryption, known as ElGamal.
 - As an alternative, RSA can be used for both signatures and session key encryption.
- For the hash function used to create the digital signature, the specification requires the **160-bit SHA-1** but recommends receiver support for the **128-bit MD5** for backward compatibility with older versions of S/MIME
- For message encryption, **three-key triple DES (3DES)** is recommended.



Table 7.6 Cryptographic Algorithms Used in S/MIME

Function	Requirement
Create a message digest to be used in forming a digital signature.	MUST support SHA-1. Receiver SHOULD support MD5 for backward compatibility.
Encrypt message digest to form a digital signature.	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 a 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 a one-time session key.	Sending and receiving agents MUST support encryption with tripleDES. 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. Sending agents SHOULD support HMAC with SHA-1.

❏ ***CRYPTOGRAPHIC ALGORITHMS***

- S/MIME specification includes a discussion of the procedure for deciding which content encryption algorithm to use.
- A sending agent has two decisions to make:
 - if the receiving agent is capable of decrypting using a given encryption algorithm
 - if the receiving agent is only capable of accepting weakly encrypted content, then is it acceptable?



❏ ***CRYPTOGRAPHIC ALGORITHMS***

- The following rules, in the following order, should be followed by a sending agent
 - If the sending agent has a list of preferred decrypting capabilities from an intended recipient, it **SHOULD** choose the first capability on the list that it is capable of using
 - If the sending agent has no such list of capabilities from an intended recipient but has received one or more messages from the recipient, then the outgoing message **SHOULD** use the same encryption algorithm as was used on the last signed and encrypted message received from that intended recipient



CONTINUE...

- If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is willing to risk that the recipient may not be able to decrypt the message, then the sending agent SHOULD use triple DES
- If the sending agent has no knowledge about the decryption capabilities of the intended recipient and is not willing to risk that the recipient may not be able to decrypt the message, then the sending agent MUST use RC2/40



□ S/MIME MESSAGES

- S/MIME makes use of a number of new MIME content types as given in following table.
- All of the new application types use the designation PKCS. This refers to a set of public-key cryptography specifications issued by RSA Laboratories.
- ***SECURING A MIME ENTITY***
 - S/MIME secures a MIME entity with a signature, encryption, or both.
 - A MIME entity may be an entire message, or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message.



CONTINUE...

Table 7.7 S/MIME Content Types

Type	Subtype	smime Parameter	Description
Multipart	Signed		A clear-signed message in two parts: one is the message and the other is the signature.
Application	pkcs7-mime	signedData	A signed S/MIME entity.
	pkcs7-mime	envelopedData	An encrypted S/MIME entity.
	pkcs7-mime	degenerate signedData	An entity containing only public-key certificates.
	pkcs7-mime	CompressedData	A compressed S/MIME entity.
	pkcs7- signature	signedData	The content type of the signature subpart of a multipart/signed message.



CONTINUE...

○ **ENVELOPED DATA**

- The steps for preparing an enveloped Data MIME entity are
 - Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or triple DES)
 - For each recipient, encrypt the session key with the recipient's public RSA key
 - 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
 - Encrypt the message content with the session key.



CONTINUE...

○ **SIGNED DATA**

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



CONTINUE...

○ ***CLEAR SIGNING***

- Clear signing is achieved using the multipart content type with a signed subtype
- This signing process does not involve transforming the message to be signed, so that the message is sent “in the clear.”
- Thus, recipients with MIME capability but not S/MIME capability are able to read the incoming message.



CONTINUE...

○ **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, followed by the signature of the certification RequestInfo block made using the sender's private key.



CONTINUE...

○ ***CERTIFICATES-ONLY MESSAGE***

- A message containing only certificates or a certificate revocation list (CRL) can be sent in response to a registration request



❑ S/MIME CERTIFICATE PROCESSING

○ **USER AGENT ROLE**

- An S/MIME user has several key-management functions to perform.
 - Key generation
 - MUST be capable of generating separate Diffie-Hellman and DSS key pairs and SHOULD be capable of generating RSA key pairs.
 - Each key pair MUST be generated from a good source of nondeterministic random input and be protected in a secure fashion.
 - SHOULD generate RSA key pairs with a length in the range of 768 to 1024 bits and MUST NOT generate a length of less than 512 bits.



CONTINUE...

- Registration
 - A user's public key must be registered with a certification authority in order to receive an X.509 public-key certificate.
- Certificate storage and retrieval
 - Certificate must be stored in a PKI.



CONTINUE...

○ **VERISIGN CERTIFICATES**

- There are several companies that provide certification authority (CA) services.
- There are a number of Internet-based CAs, including VeriSign, GTE, and the U.S. Postal Service.
- VeriSign provides a CA service that is intended to be compatible with S/MIME and a variety of other applications.
- VeriSign issues X.509 certificates with the product name VeriSign Digital ID



CONTINUE...

- 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



CONTINUE...

- VeriSign provides three levels, or classes, of security for public-key certificates
 - For Class 1 Digital IDs, VeriSign confirms the user's e-mail address by sending a PIN
 - For Class 2 Digital IDs, VeriSign verifies the information in the application
 - For Class 3 Digital IDs, VeriSign requires a higher level of identity assurance



❑ ENHANCED SECURITY SERVICES

- **Signed receipts:** Returning a signed receipt provides proof of delivery to the originator of a message and allows the originator to demonstrate to a third party that the recipient received the message.
- **Security labels:** a set of security information regarding the sensitivity of the content that is protected by S/MIME encapsulation. E.g. access control, priorities based on different factors
- **Secure mailing lists:** 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.



