Pavan Boora
M.Tech in Networking & Internet
Topic : Pretty Good Privacy
Dept of Information Science
Jain University Bangalore

Pretty Good Privacy

- PGP is an open-source, freely available software package for e-mail security.

- It provides Authentication through the use of digital signature,
- The confidentiality through the use of symmetric block encryption,
- Compression using the ZIP algorithm, and
- E-Mail compatibility using the radix-64 encoding scheme.

- PGP has grown very quickly and is now widely used. Here are some reason for this growth.

1. It is freely available worldwide in versions that run on variety of platforms. In addition commercial versions provides vendor support

2. 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 applicabilities, encrypting files and messages to individuals who wish to communicate securely with others worldwide over the Internet.

Operational Description of PGP

- The operation of PGP, consists of four services:

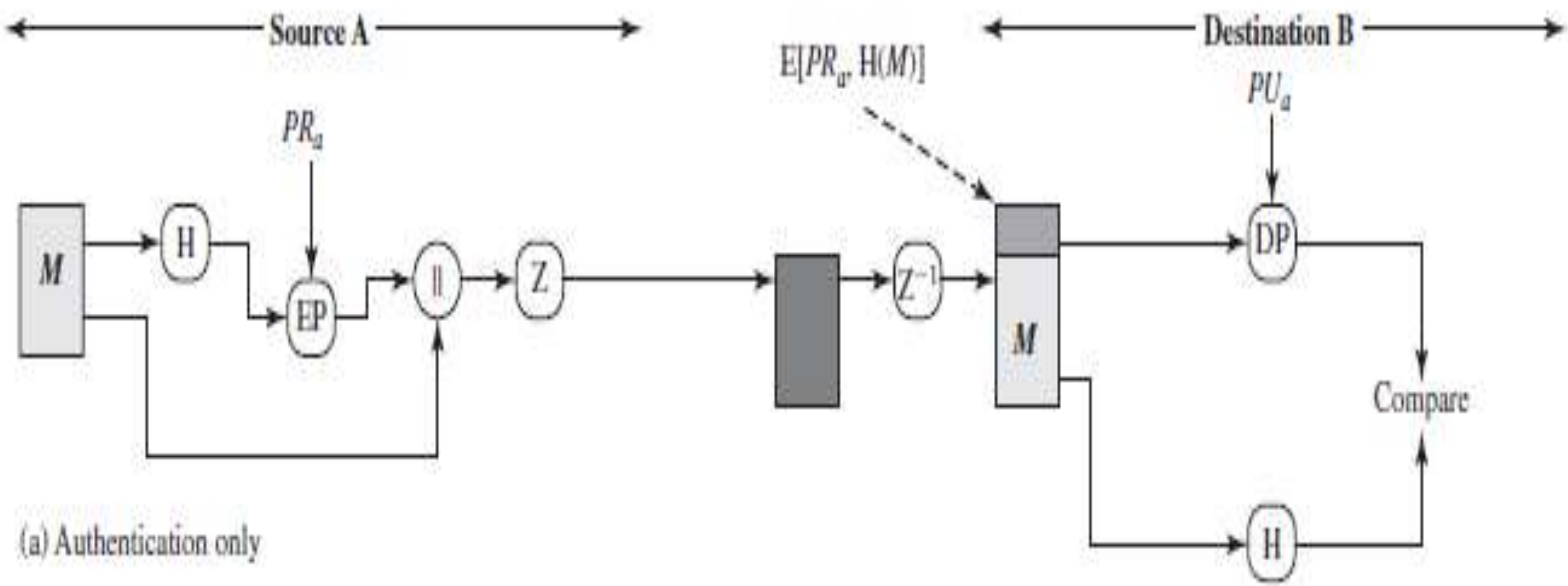
1. Authentication
2. Confidentiality
3. Compression
4. E-mail compatibility &
5. Segmentation

Authentication

- The digital signature service provided by PGP.

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.

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[1]



(a) Authentication only

Confidentiality

- PGP another service is confidentiality, which is encrypting messages for transmitting or to store files locally.
- In both cases, the symmetric encryption algorithm CAST-128 may be used. Alternatively, IDEA or 3DES may be used. And the 64-bit cipher feedback (CFB) mode is used.
- In PGP, each symmetric key is used only once. The session key is bound to the message. To protect the key, it is encrypted with the receiver's public key.

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.

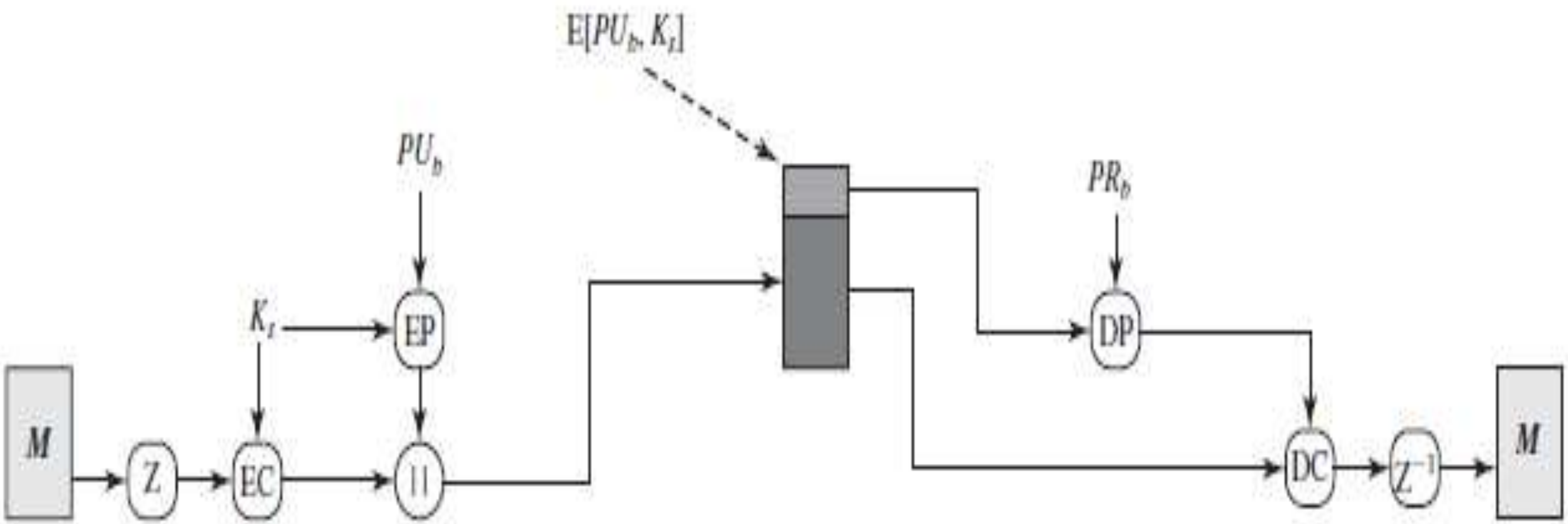
3. The session key is encrypted with RSA using the recipient's public key and is prepended to the message.

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.

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[1]



(b) Confidentiality only

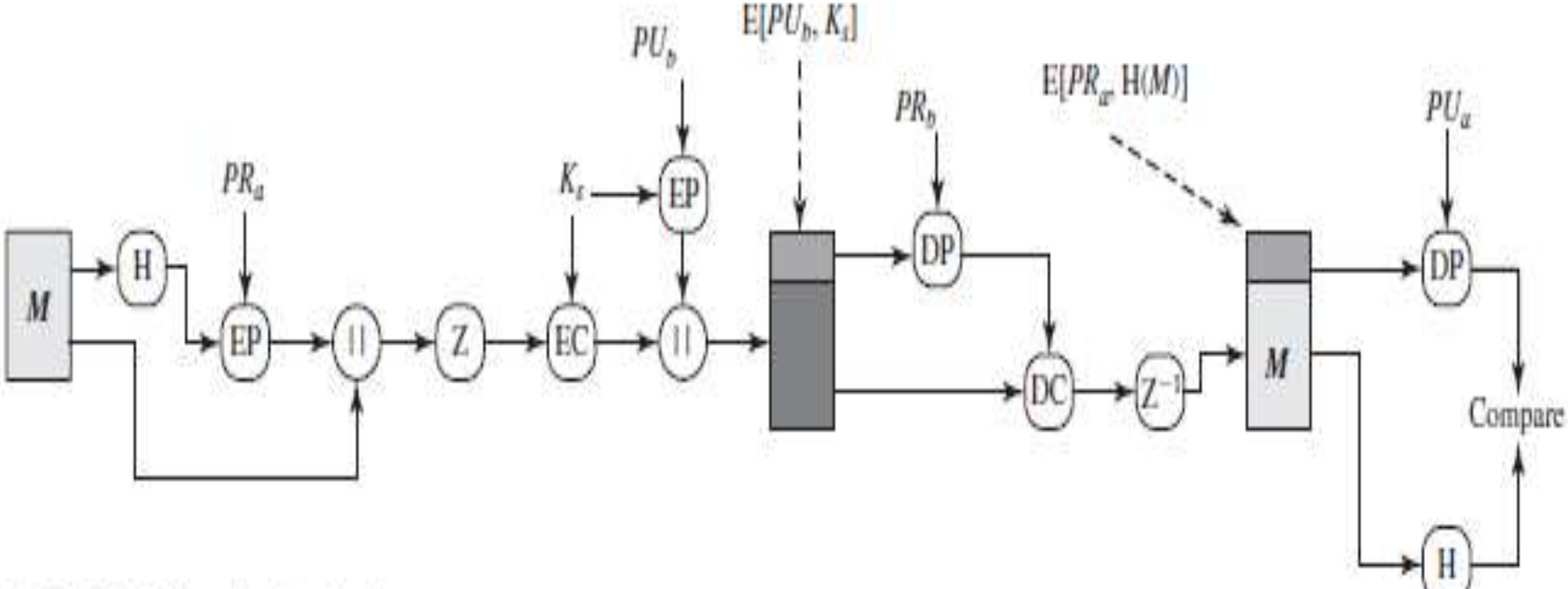
Confidentiality & Authentication

- 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).

- 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.

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



(c) Confidentiality and authentication

Compression

- 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.
- Z for compression and Z–1 for decompression

The signature is generated before compression for two reasons:

- It is preferable to sign an uncompressed message so that one can store only the uncompressed message together with the signature for future verification.

- If you generate signature after compression then there is a need recompression for message verification, PGP's compression algorithm presents a difficulty.

- Message encryption is applied after compression to strengthen cryptographic security. Therefore cryptanalysis is more difficult.
- The compression algorithm used here is ZIP Algorithm

E-Mail-Compatibility

- The resulting message 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.

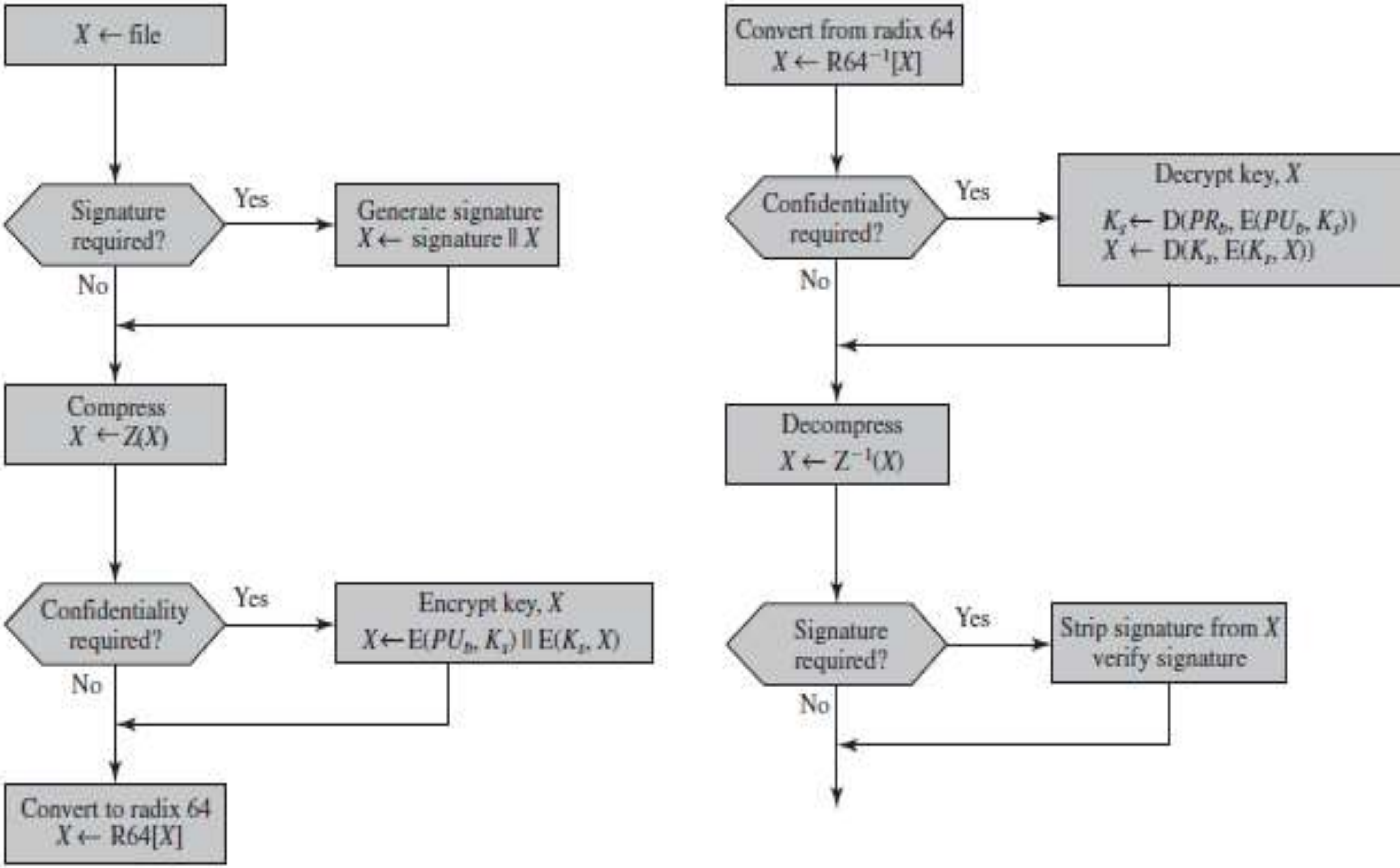
- The use of radix 64 expands a message by 33%. Fortunately, the session key and signature portions of the message are relatively compact, and the plaintext message has been compressed.

- In fact, the compression should be more than enough to compensate for the radix-64 expansion.

Segmentation & Reassembly

- E-mail facilities often are restricted to a maximum length. To accommodate this, PGP automatically subdivides a messsage that is too large into segments that are small enough to send via e-mail.

- The segmentation is done after all of the other processing, including the radix-64 conversion.



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Figure 18.2 Transmission and Reception of PGP Messages

Summary of PGP Services

Table 18.1 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.

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, and
4. passphrase-based symmetric keys

- Three separate requirements can be identified with respect to these keys.

1. generating unpredictable session keys is needed
2. PGP would like to allow a user to have multiple public-key/private-key pairs. One reason is that the user may wish to change his or her key pair from time to time.
3. Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of OTHERS.

Session key Generation

- Each session key is used encrypting and decrypting only one message.
- Message encryption/decryption is done with a symmetric encryption algorithm. CAST-128 and IDEA use 128-bit keys; 3DES uses a 168-bit key.
    - Let assume CAST-128
- Random 128-bit numbers are generated using CAST-128 itself.
- 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.
- The CAST-128, is to produce a sequence of session keys that is effectively unpredictable.
- The algorithm that is used is based on ANSI X12.17

key Identifiers

- An encrypted message is attached by an encrypted form of the session key that was used for message encryption. The session key itself is encrypted with the recipient’s public key.

- Hence, only the recipient will be able to recover the session key and therefore recover the message.

- If each user have single public/private key pair, then the recipient would automatically know which key to use to decrypt the session key.

- However, we have stated a requirement that any given user may have multiple public/private key pairs.

- One simple solution would be to transmit the public key with the message This scheme would work, but it is unnecessarily wasteful of space. An RSA public key may be hundreds of decimal digits in length.

- Another solution would be to associate an identifier with each public key that is unique. That is, the combination of user ID and key ID would be sufficient to identify a key uniquely however, it raises a management and overhead problem.

- Key IDs must be assigned and stored so that both sender and recipient could map from key ID to public key. This seems unnecessarily burdensome.

- 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.

- A key ID is also required for the PGP digital signature. Because a sender may use one of a number of private keys to encrypt the message digest, the recipient must know which public key is intended for use.

- A message consists of three components:

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

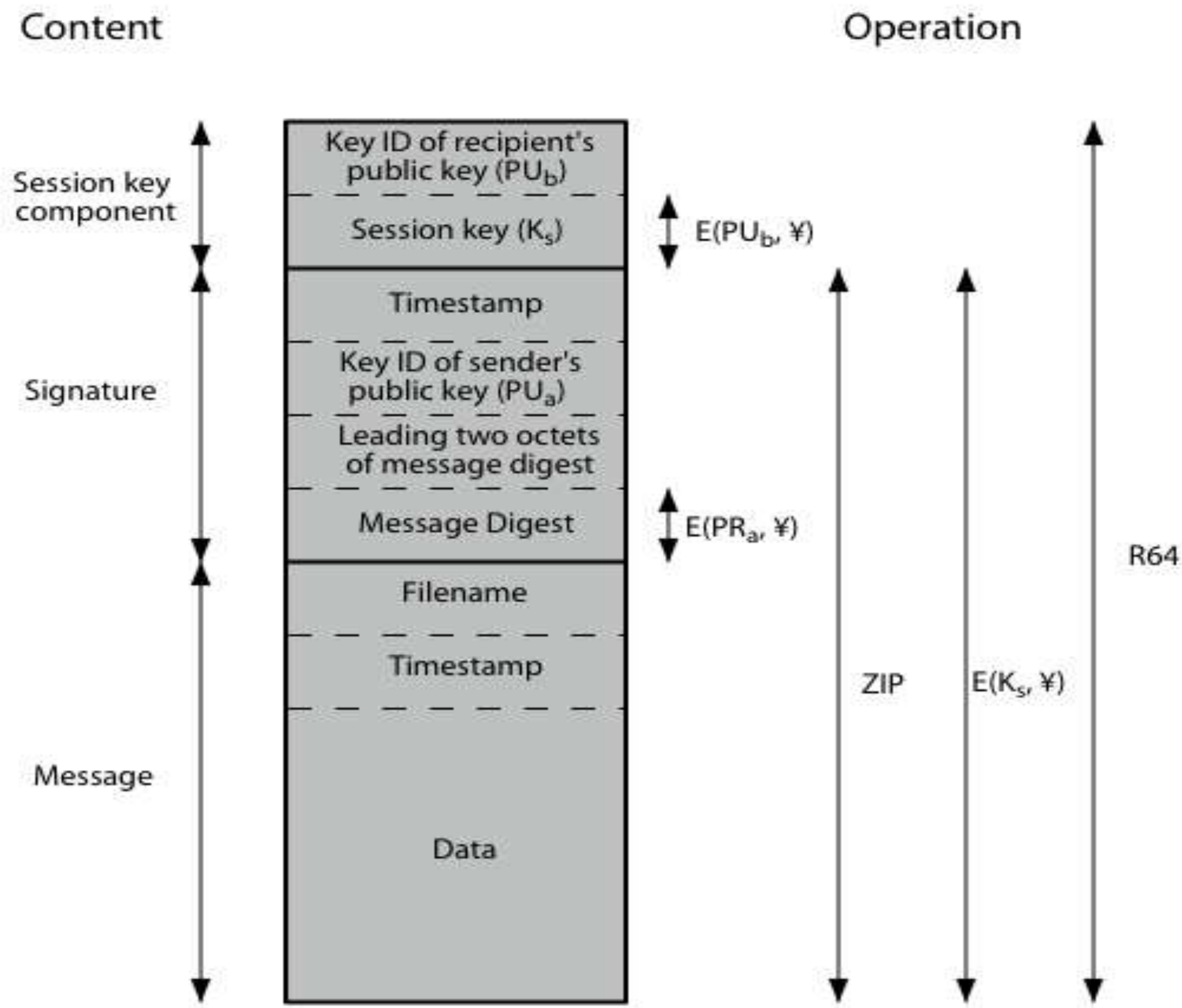
- Leading two octets of message digest:

  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.


Content
Operation
Session key component
Key ID of recipient's public key (PU_b)
Session key (K_s)
$E(PU_b, ¥)$
Signature
Timestamp
Key ID of sender's public key (PU_a)
Leading two octets of message digest
Message Digest
$E(PR_a, ¥)$
Message
Filename
Timestamp
Data
ZIP
$E(K_s, ¥)$
R64

- 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.

key Rings

- We have seen how key IDs are critical in the the operation of PGP

- 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 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 key rings are referred to, respectively, as the private-key ring and the public-key ring.

Private-key Ring

- 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.
- **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).

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
•	•	•	•	•
•	•	•	•	•
•	•	•	•	•

Public-key Ring

- **public-key ring**. This ring 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.

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	$trust\_flag_i$	User i	$trust\_flag_i$		
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•
•	•	•	•	•	•	•	•

* = field used to index table

- 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.

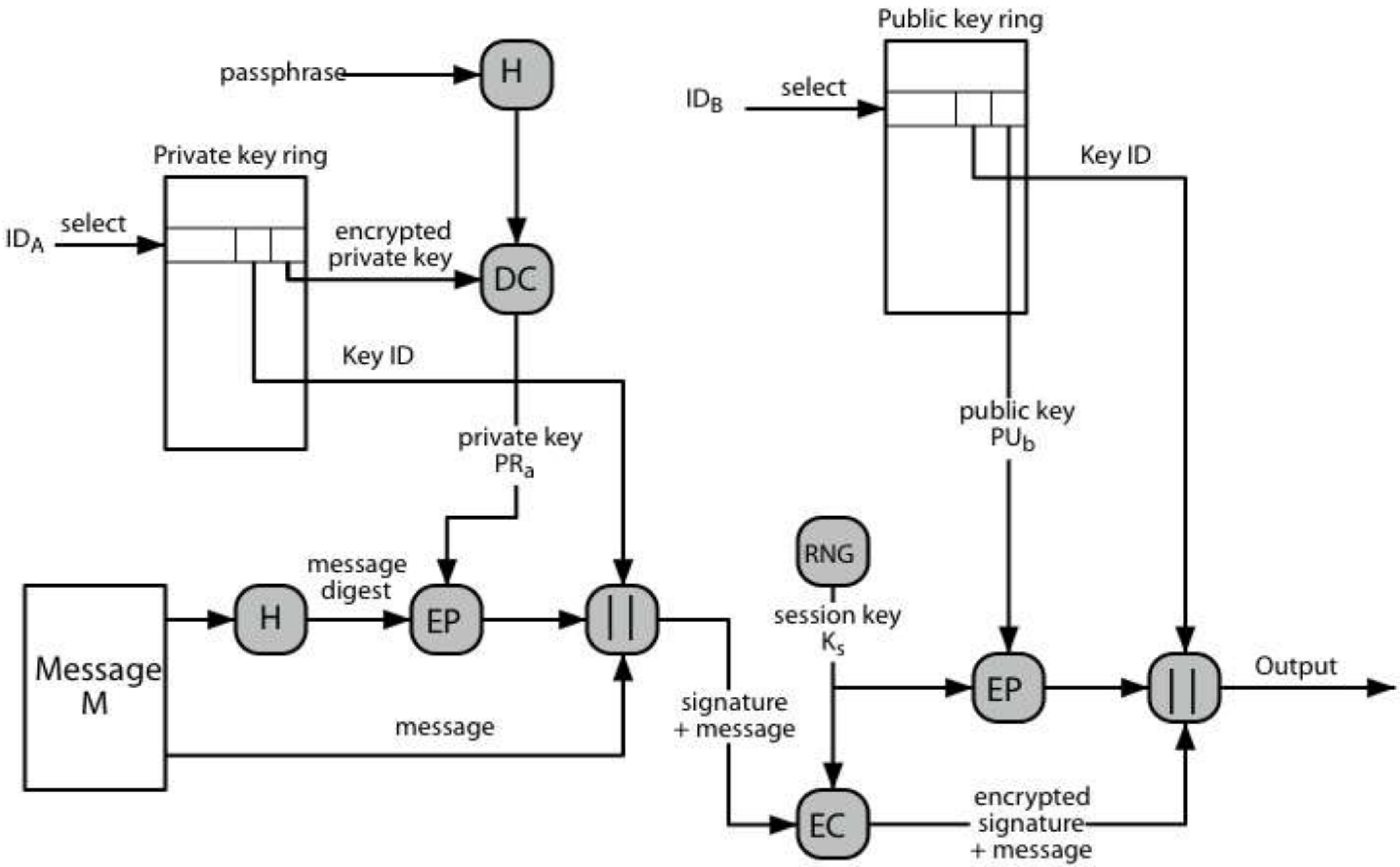
2. Encrypting the message:

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 Msg Generation



- 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.

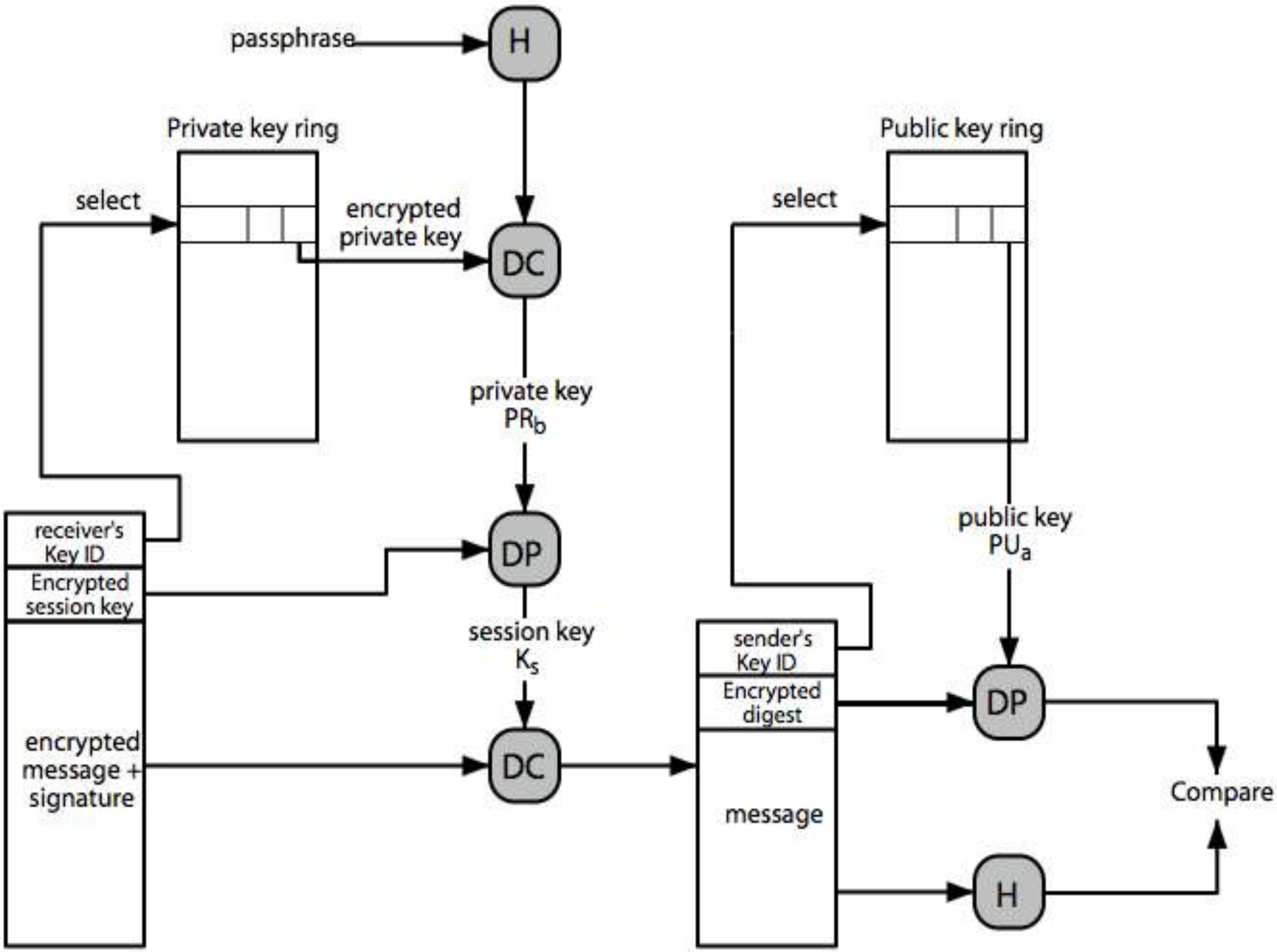
2. Authenticating the message:

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



Public-key Management

- In practical public key applications, protecting public keys from tampering is the single most difficult problem It is the “Achilles heel” of public key cryptography, and a lot of software complexity is tied up in solving this one problem.

APPROACHES TO PUBLIC-KEY MANAGEMENT

- A number of approaches are possible for minimizing the risk that a user's public-key ring contains false public keys

1. Physically get the key from B.

2. Verify a key by telephone.

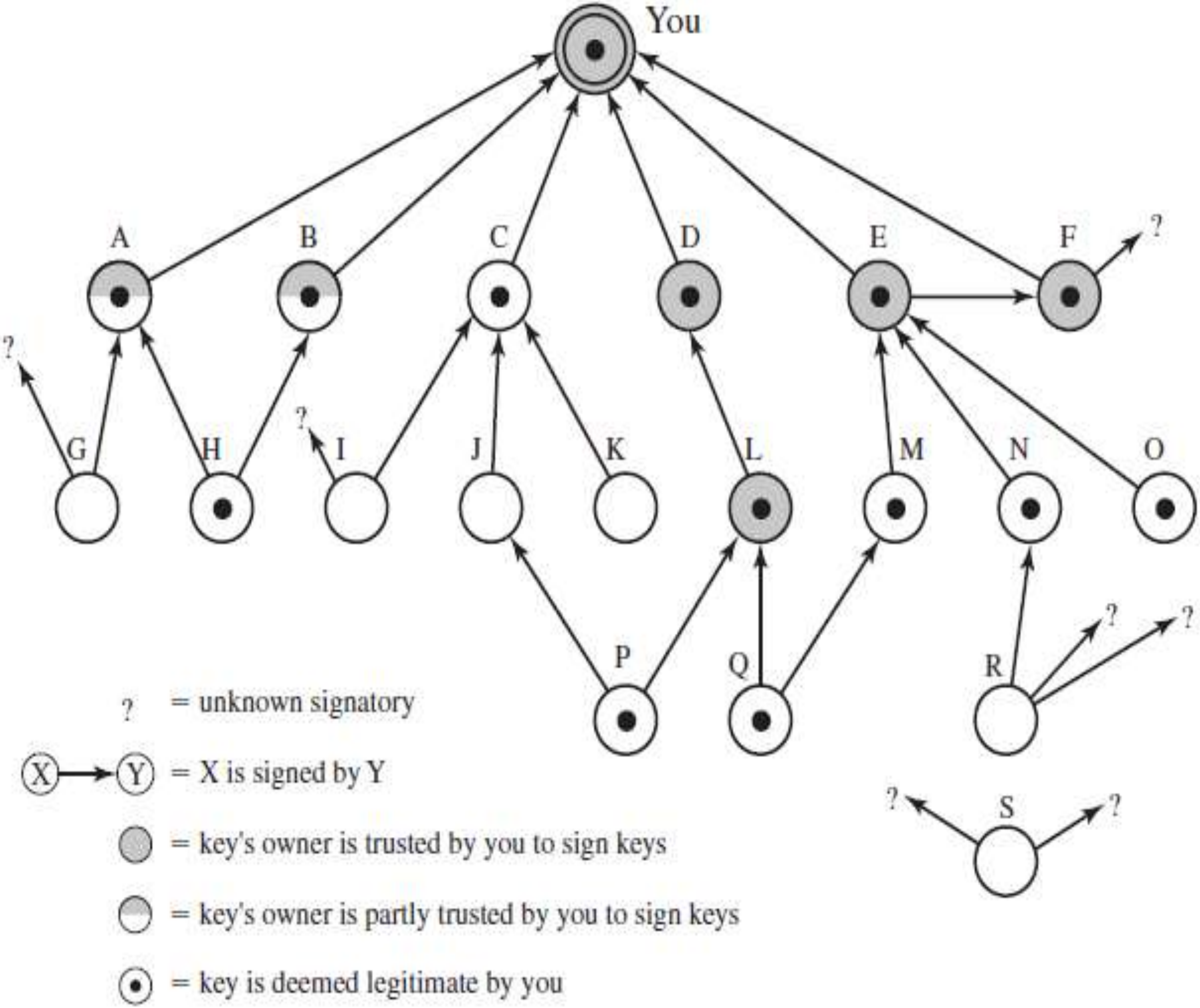
3. Obtain B's public key from a mutual trusted individual D. For this purpose, the introducer D, creates a signed certificate.

4. Obtain B's public key from a trusted certifying authority. Again, a public-key certificate is created and signed by the authority.

PGP Trust Model

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

- So the shading, of the nodes in Figure 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 a 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.



You
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
? = unknown signatory
X → Y = X is signed by Y
= key's owner is trusted by you to sign keys
= key's owner is partly trusted by you to sign keys
= key is deemed legitimate by you

Thank You..!