Electronic Mail Security -Privacy- -Chapter 18.1: PGP-

- Email privacy, without some security precautions, can be compromised because:
  - email messages are generally not encrypted.
  - We mail messages have to go through intermediate computers before reaching their destination, meaning it is relatively easy for others to intercept and read messages.
  - many Internet Service Providers (ISP) store copies of email messages on their mail servers before they are delivered. The backups of these can remain for up to several months on their server, despite deletion from the mailbox.
  - the "Received:"-fields and other information in the email can often identify the sender, preventing anonymous communication.

Electronic Mail Security

- Solution: Use of Cryptography Protocols and Applications

- For example, Virtual Private Networks or the Tor anonymity network can be used to encrypt traffic from the user machine to a safer network while GPG, PGP, SMEmail, or S/MIME can be used for end-to-end message encryption, and SMTP STARTTLS or SMTP over Transport Layer Security/Secure Sockets Layer can be used to encrypt communications for a single mail hop between the SMTP client and the SMTP server.

- Additionally, many mail user agents do not protect logins and passwords, making them easy to intercept by an attacker. Encrypted authentication schemes such as SASL prevent this.

  http://en.wikipedia.org/wiki/Email#Privacy_concerns

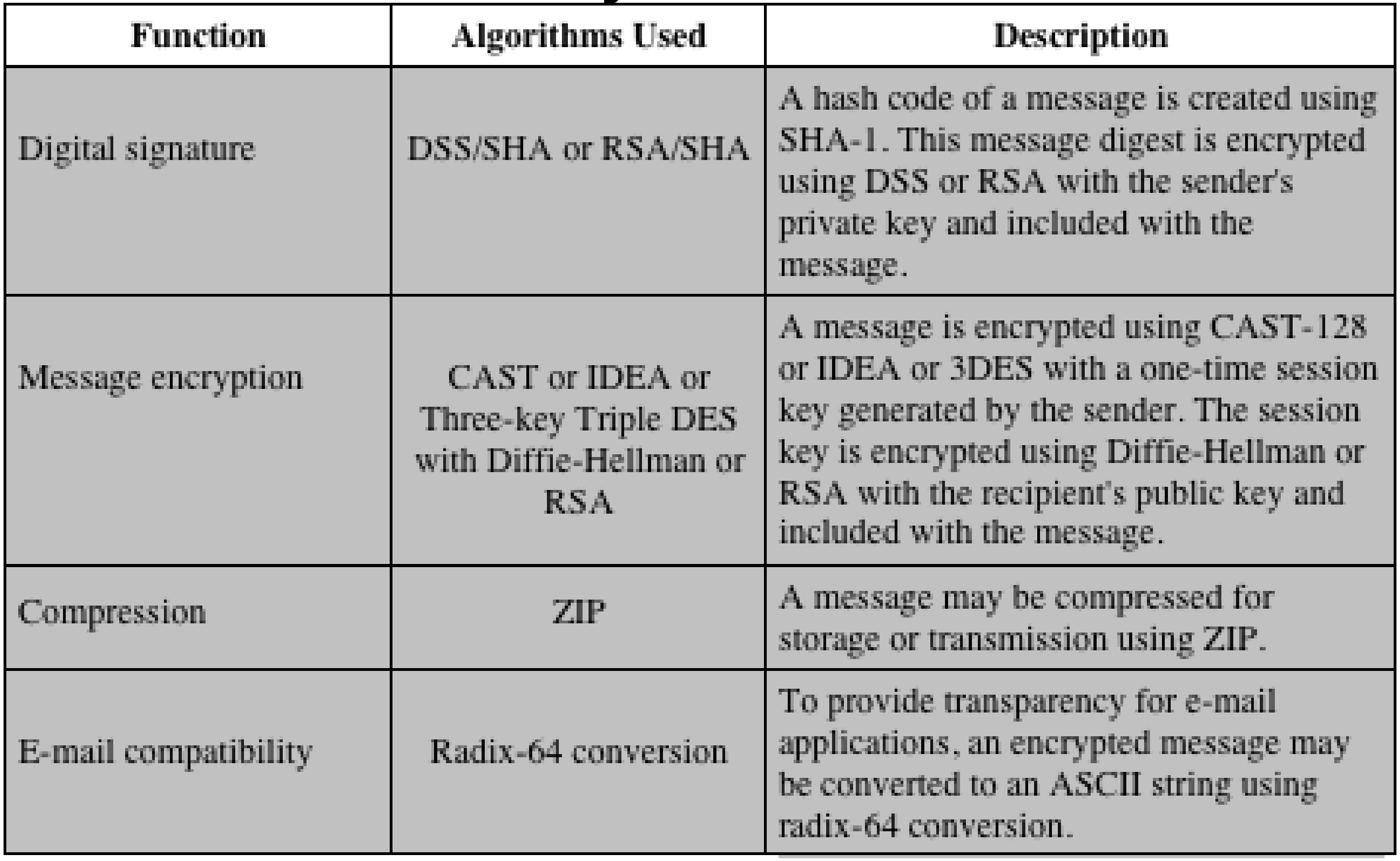
Electronic Mail Security

- Pretty Good Privacy (PGP) : data encryption and decryption computer program that provides cryptographic privacy and authentication for data communication.
- PGP is often used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications.
- It was created by Phil Zimmermann in 1991.
- PGP and similar software follow the **OpenPGP** standard (RFC 4880) for encrypting and decrypting data.

Electronic Mail Security

- Pretty Good Privacy (PGP) incorporates tools for "developing a **public-key trust model** and **public-key management**".

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.

PGP Services

- PGP Consists of <u>five</u> services:
  - Authentication
  - Confidentiality
  - Compression
  - E-mail compatibility
  - Segmentation and Reassembly

- The last three are **transparent** to the user

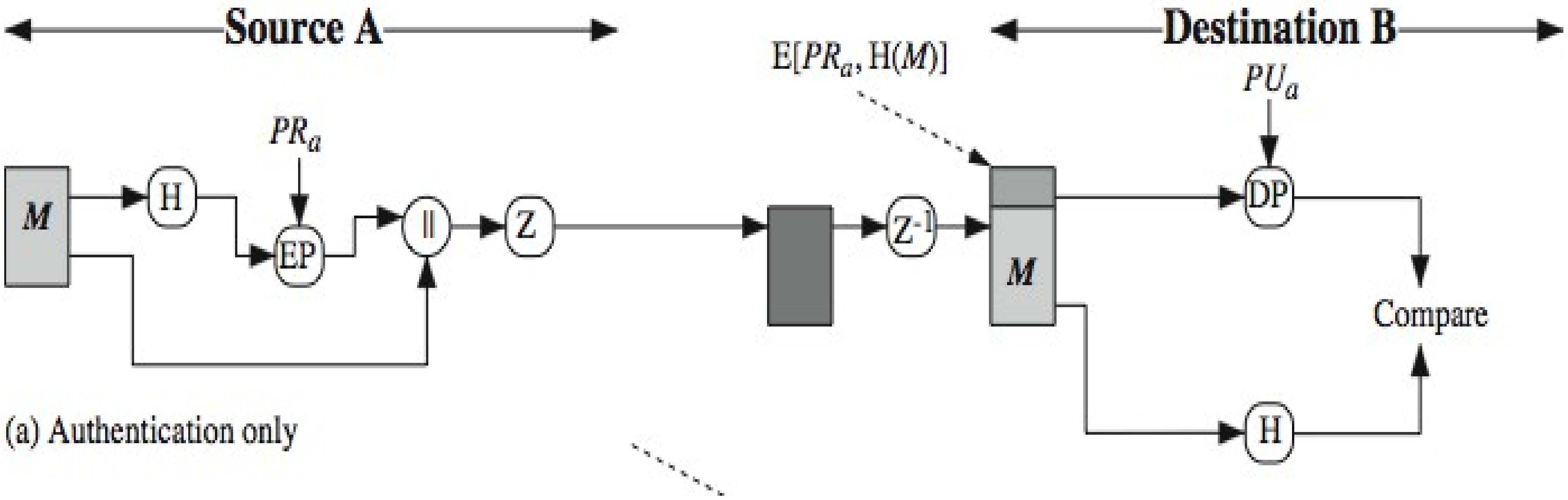
PGP Authentication

- 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
  - 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
  - As an alternative, signatures can be generated using DSS/SHA-1
  - Detached signatures are supported
  - Each person’s signature is independent and therefore applied only to the document

PGP: Authentication steps

- This is the same as the digital signature

- Digital signatures can be store for future use
- **Sender:**
  - Creates a message
  - Hashes it to 160-bits using SHA1
  - Encrypts the hash code using sender's private key, forming a signature
  - Attaches the signature to message



Authentication steps

Stallings, Fig 18.1a

M = original message
H = hash function
| | = concatenation (join)
Z = compression
Z^{-1} = decompression

EP = public key encryption
DP = public key decryption
KR_a = A's private key
KU_a = A's public key

PGP Confidentiality

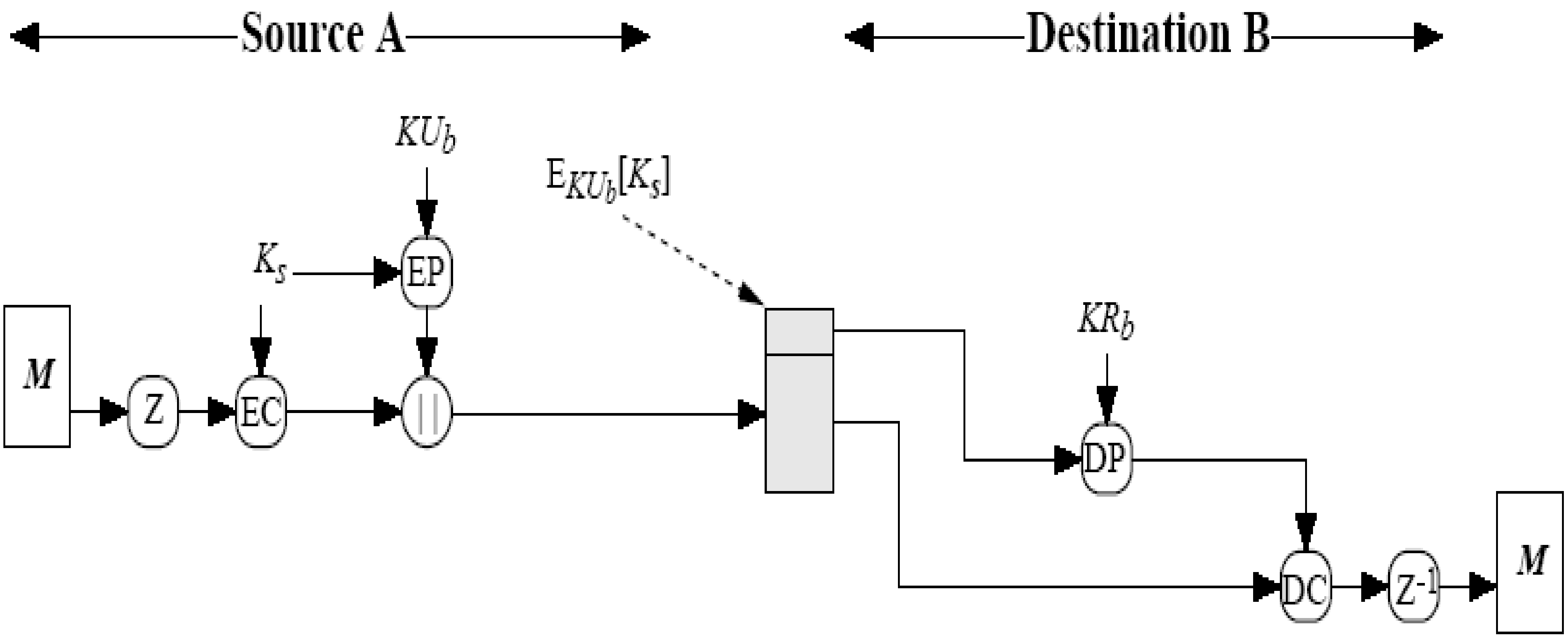
- Provided by encrypting messages to be transmitted or to be stored locally as files
  - In both cases the symmetric encryption algorithm CAST-128 may be used
  - Alternatively IDEA or 3DES may be used
  - The 64-bit cipher feedback (CFB) mode is used
  - As an alternative to the use of RSA for key encryption, PGP uses ElGamal, a variant of Diffie-Hellman that provides encryption/decryption

PGP: Confidentiality

- **Sender:**
  - Generates message and a random number (session key) only for this message
  - Encrypts message with the session key using AES, 3DES, IDEA or CAST-128
  - Encrypts session key itself with recipient's public key using RSA
  - Attaches it to message

PGP: Confidentiality

- **Receiver:**
  - Recovers session key by decrypting using his private key
  - Decrypts message using the session key.
  - The randomly chosen session key used for this is sent encrypted using the recipient’s public RSA key.



EC = symmetric encryption
DC = symmetric decryption
K_s = session key
Z = compression
Z^{-1} = Decompression

Table 18.1b Stallings

PGP 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 plaintext message plus signature is encrypted using CAST-128 (or IDEA or 3DES) and the session key is encrypted using RSA (or ElGamal)
- When both services are used:

PGP 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
  - The placement of the compression algorithm is critical
    - Applying the hash function and signature after compression would constrain all PGP implementations to the same version of the compression algorithm
    - Message encryption is applied after compression to strengthen cryptographic security
  - The compression algorithm used is ZIP

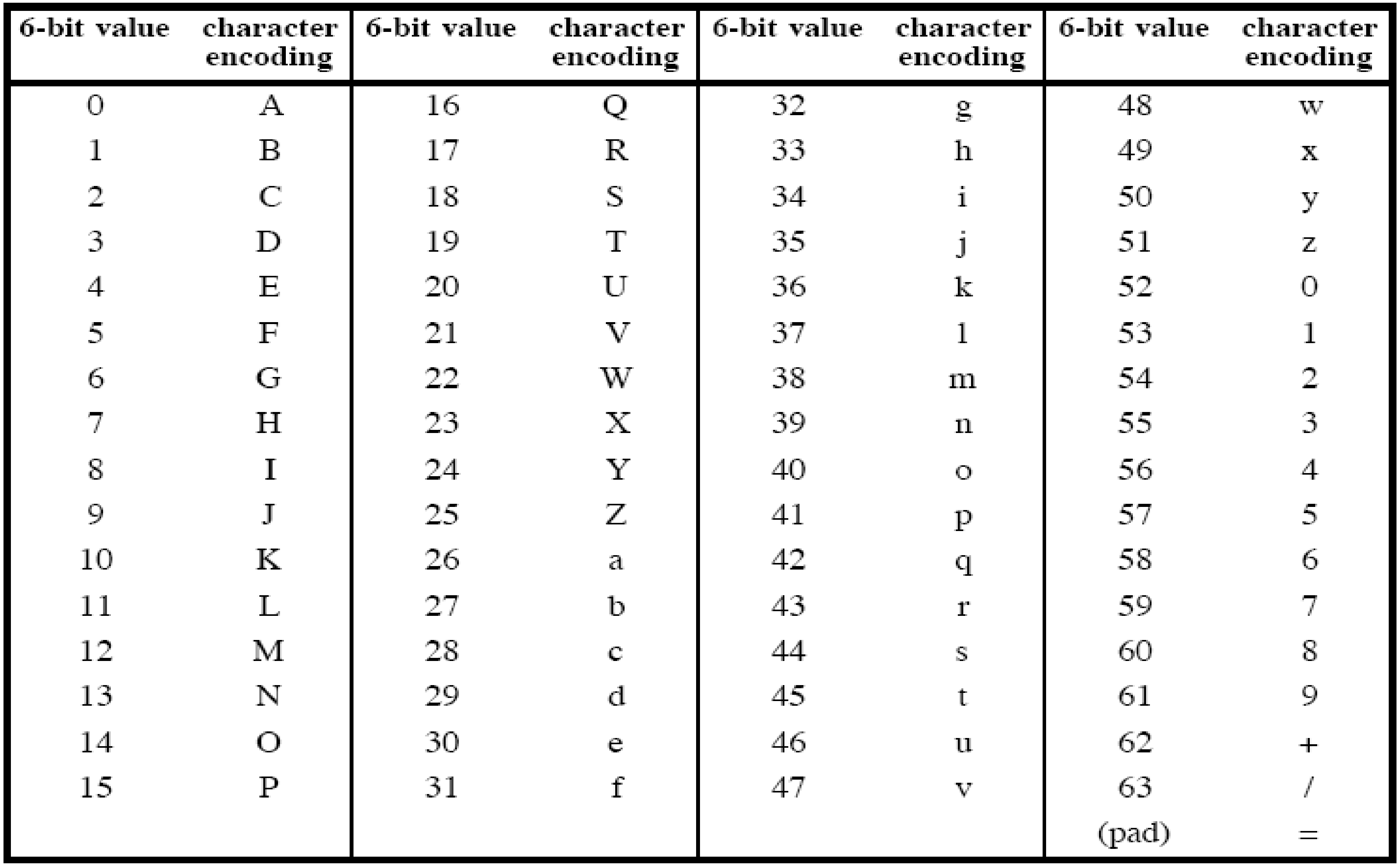
PGP E-mail Compatibility

- 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

PGP Email Compatibility

- Maps 3 binary bytes to 4 printable characters
- Appends a CRC (cyclic redundancy check) – to check for errors
- PGP also segments messages if too big

RADIX-64 encoding



6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding	6-bit value	character encoding
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/
						(pad)	=

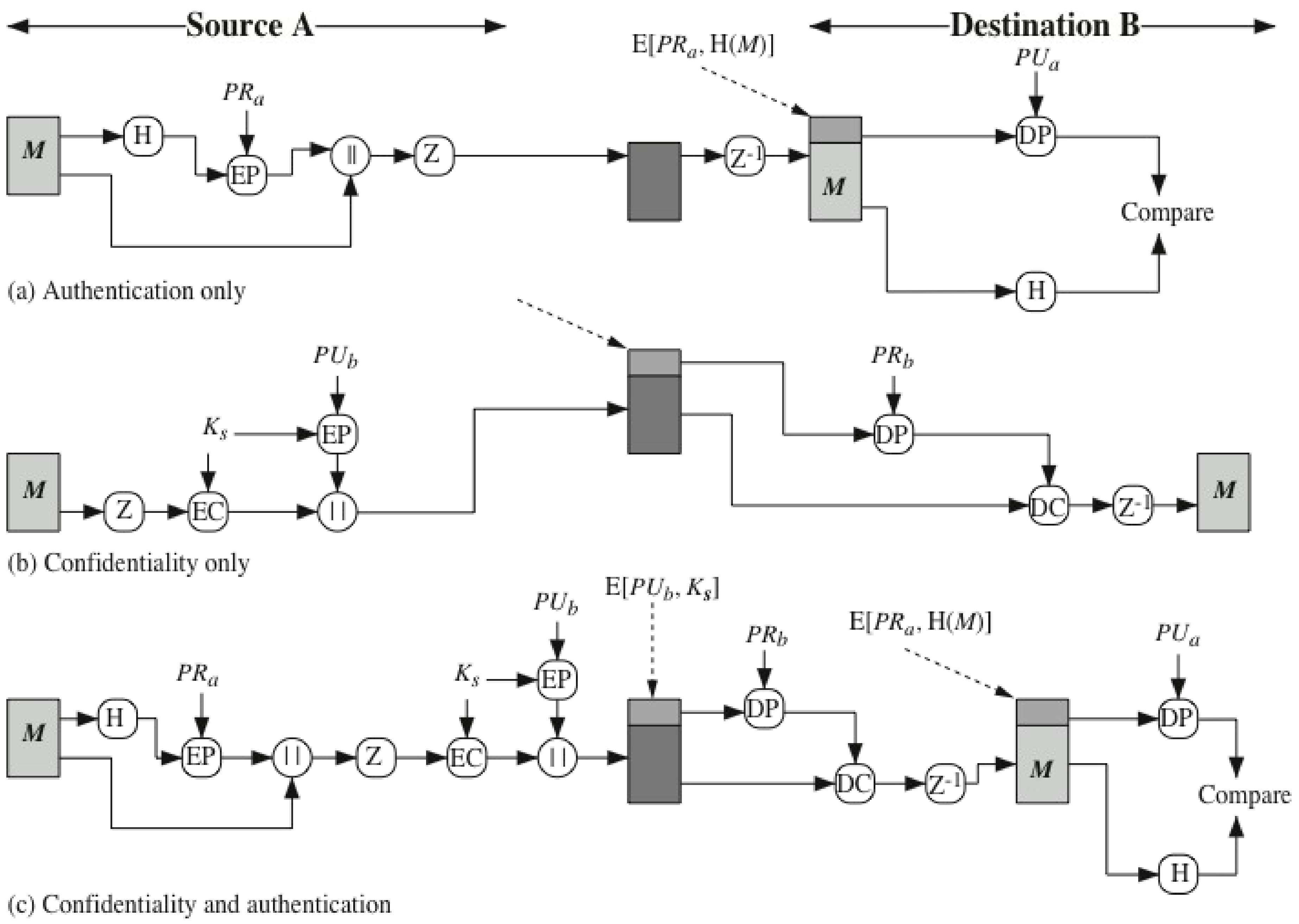
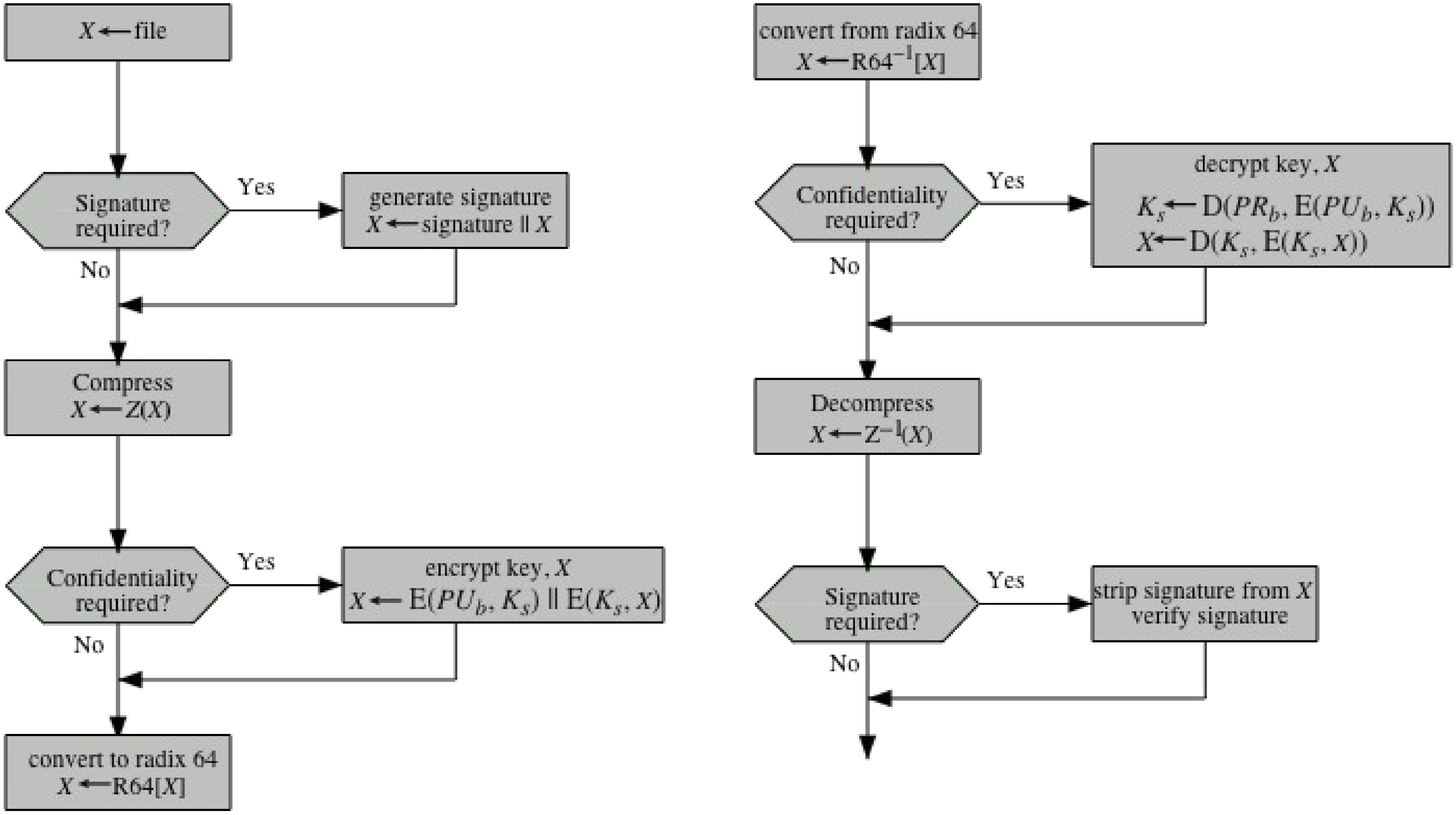


Figure 19.1 PGP Cryptographic Functions



(a) Generic Transmission Diagram (from A)

(b) Generic Reception Diagram (to B)

Figure 19.2 Transmission and Reception of PGP Messages

PGP Services

- PGP Consists of <u>five</u> services:
  - Authentication
  - Confidentiality
  - Compression
  - E-mail compatibility
  - Segmentation and Reassembly

E-mail Compatibility

The scheme used is radix-64 conversion

.

The use of radix-64 expands the message by 33%.

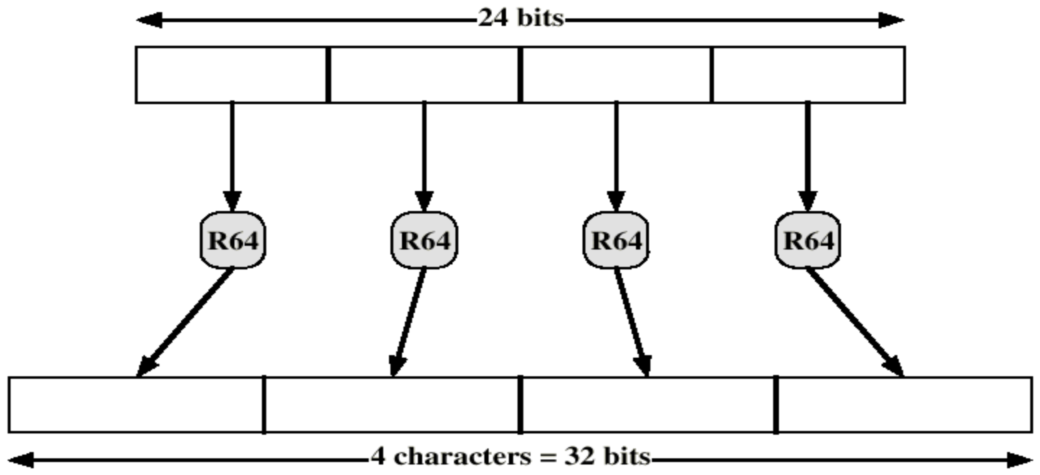
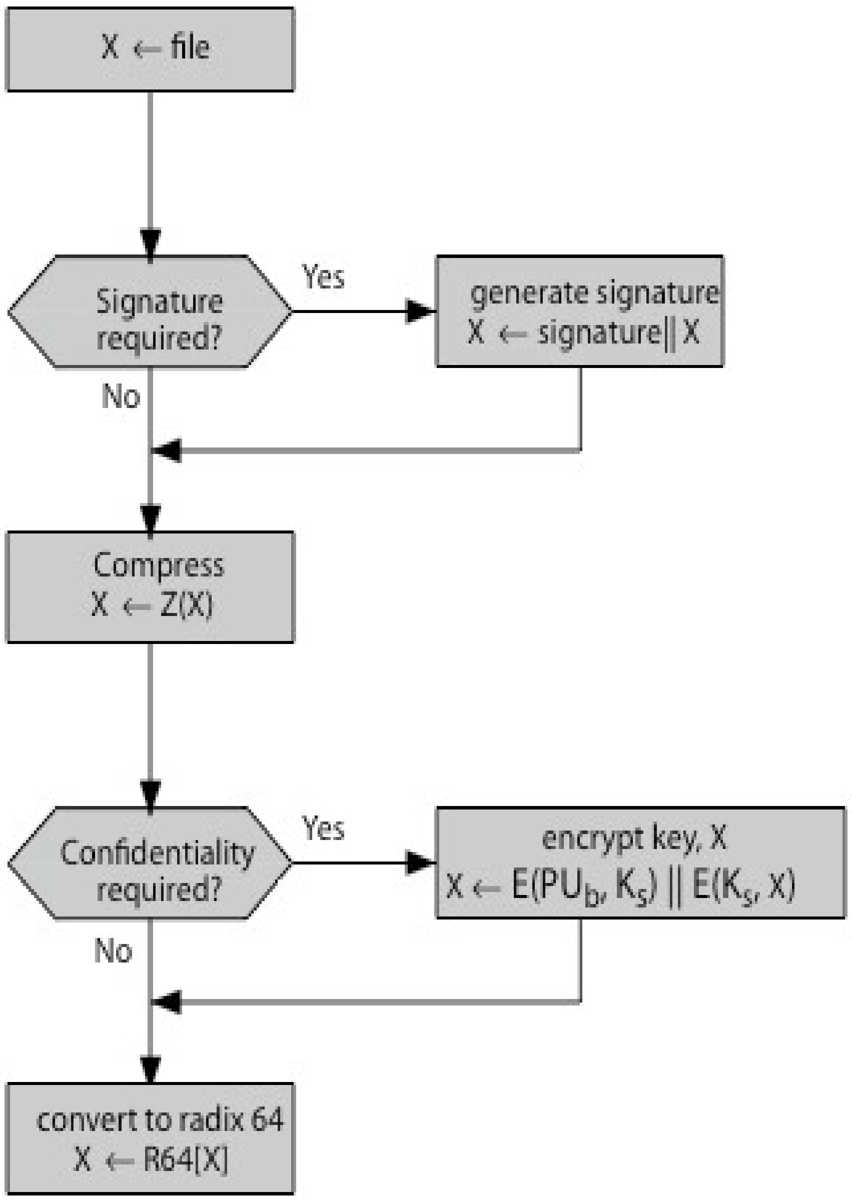


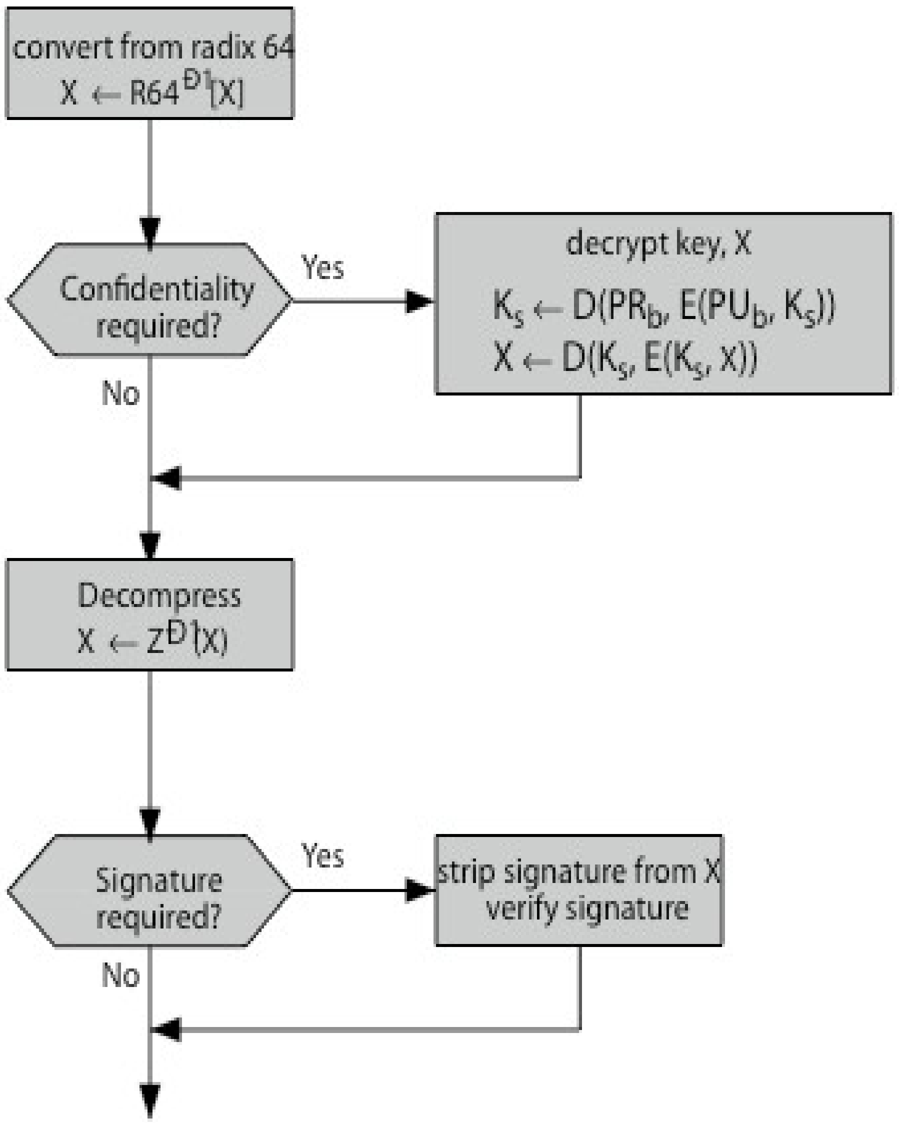
Figure 5.11 Printable Encoding of Binary Data into Radix-64 Format

Segmentation and Reassembly

- Often restricted to a maximum message length of 50,000 octets.
- Longer messages must be broken up into segments.
- PGP automatically subdivides a message that is to large.
- Segmentation (parçalama) is done after all other processing
- The receiver strips off all e-mail headers and reassemble the block.



(a) Generic Transmission Diagram (from A)



(b) Generic Reception Diagram (to B)

Cryptographic Keys and Key Rings

- PGP makes use of 4 types of keys:
  - One-time session symmetric keys
  - Public keys
  - Private Keys
  - Passphrase-based symmetric Keys

Key Requirements

- A Means of generating unpredictable <u>session</u> keys is needed
- A user is allowed to have <u>multiple public/private key pairs</u> and there must be a way to identify particular keys
- Each PGP entity must maintain a file of its own public/private key pair as well as those of its correspondents

Session keys

- Each session key is associated with a single message and is used only once to encrypt and decrypt that message

- Message encryption is done with a symmetric encryption algorithm
  - CAST, IDEA use 128 bit keys
  - 3DES uses a 168 bit key
  - Keystrokes and timing are used to generate a “random” stream, which is combined with previous session key to produce a new unpredictable one.

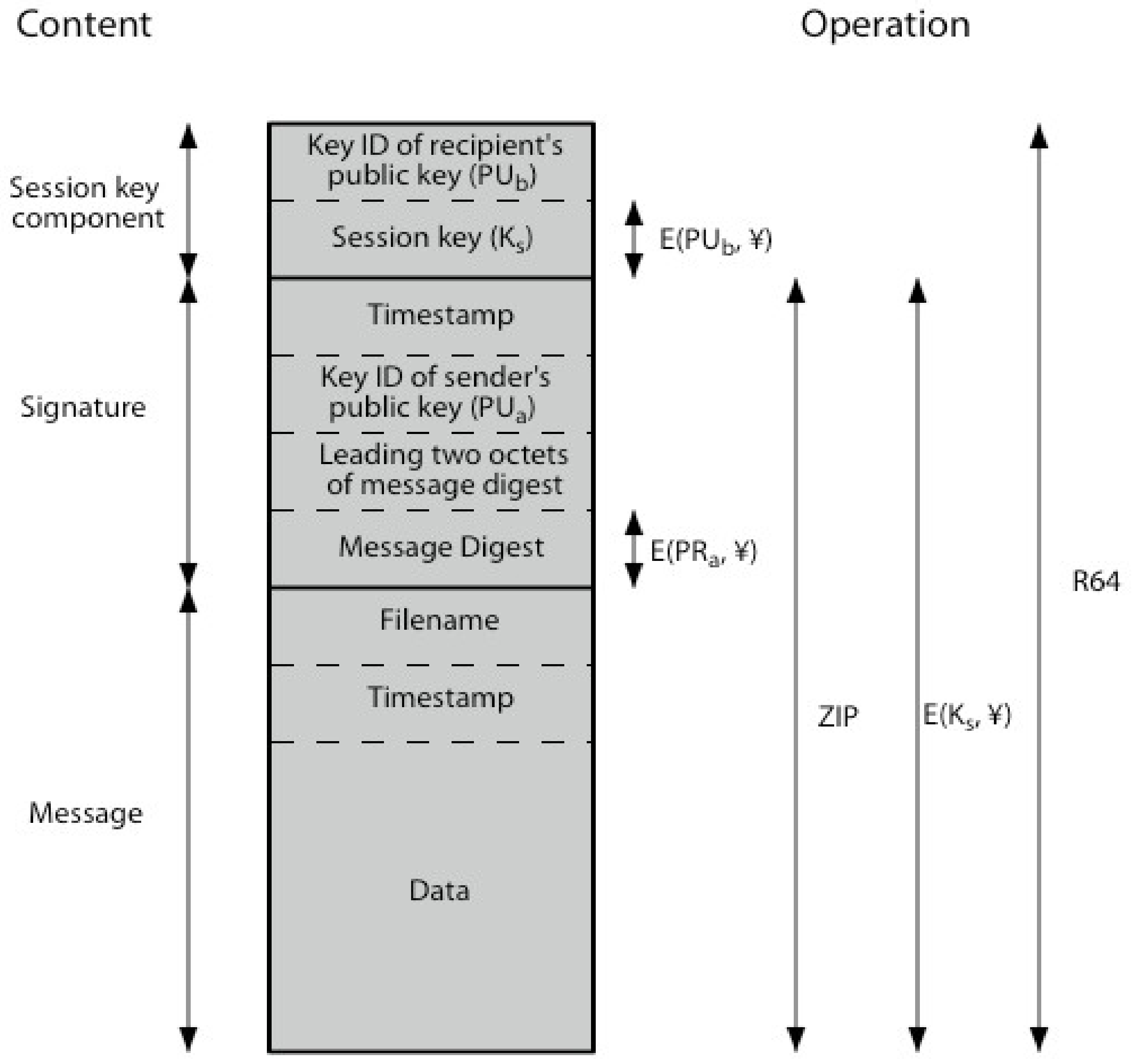
PGP Key Identifiers

- What is a **key identifier (key ID)**
- Consider this:
  - A user may have many public/private key pairs
  - He wishes to encrypt or sign a message using one of his keys
  - How does he let the other party know which key he has used?
  - Attaching the whole public key every time is inefficient
- Solution: Generate a **key identifier** (least significant 64-bits of the key)
  - This will most likely be unique and can also be used for signatures

Format of PGP Message

- A message may consist of:
- A Message component – data to be stored or transmitted
- A Signature component (optional)
  - Timestamp
  - Message digest encrypted with sender's private signature key
- A Session key (optional)
  - Session key as well as the key used to encrypt the session key
  - ZIPPED and then encoded with radix-64 encoding

Format of PGP Message



PGP Key Rings

- Each PGP user has a pair of key rings:
- PGP uses key rings to identify the key pairs that a user **owns** or **trusts**
- **Private-key ring** contains public/private key pairs of keys he **owns,** indexed by key ID and encrypted
- **Public-key ring** contains public keys of others he **trusts,** indexed by key ID

PGP Key Rings

Private Key Ring

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

Public Key Ring

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

* = field used to index table

PGP Message Generation

Key rings are used in message transmission .
The sender:

- 1. Signs the message:
  a. PGP retrieves the sender's private key from the private-key ring using your_userid as an index. If not provided, the first private key on the ring is retrieved.
  b. Prompts user for the passphrase to recover unencrypted private key
  c. The signature component of the message is constructed.

PGP Message Generation

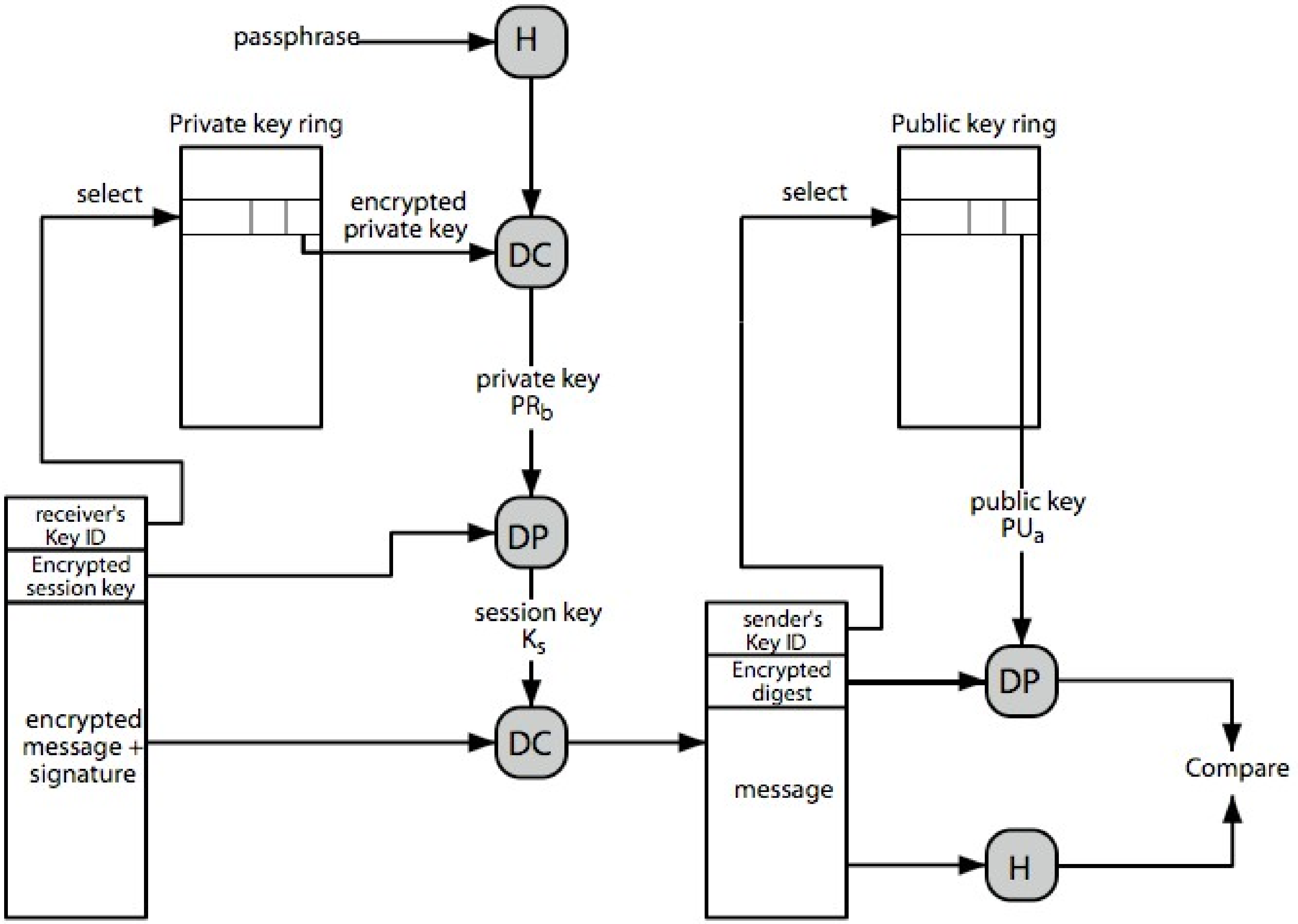
- 2. Encrypting the message:

  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 Generation



PGP Message Reception

- Receiving PGP 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

  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.

PGP Message Reception

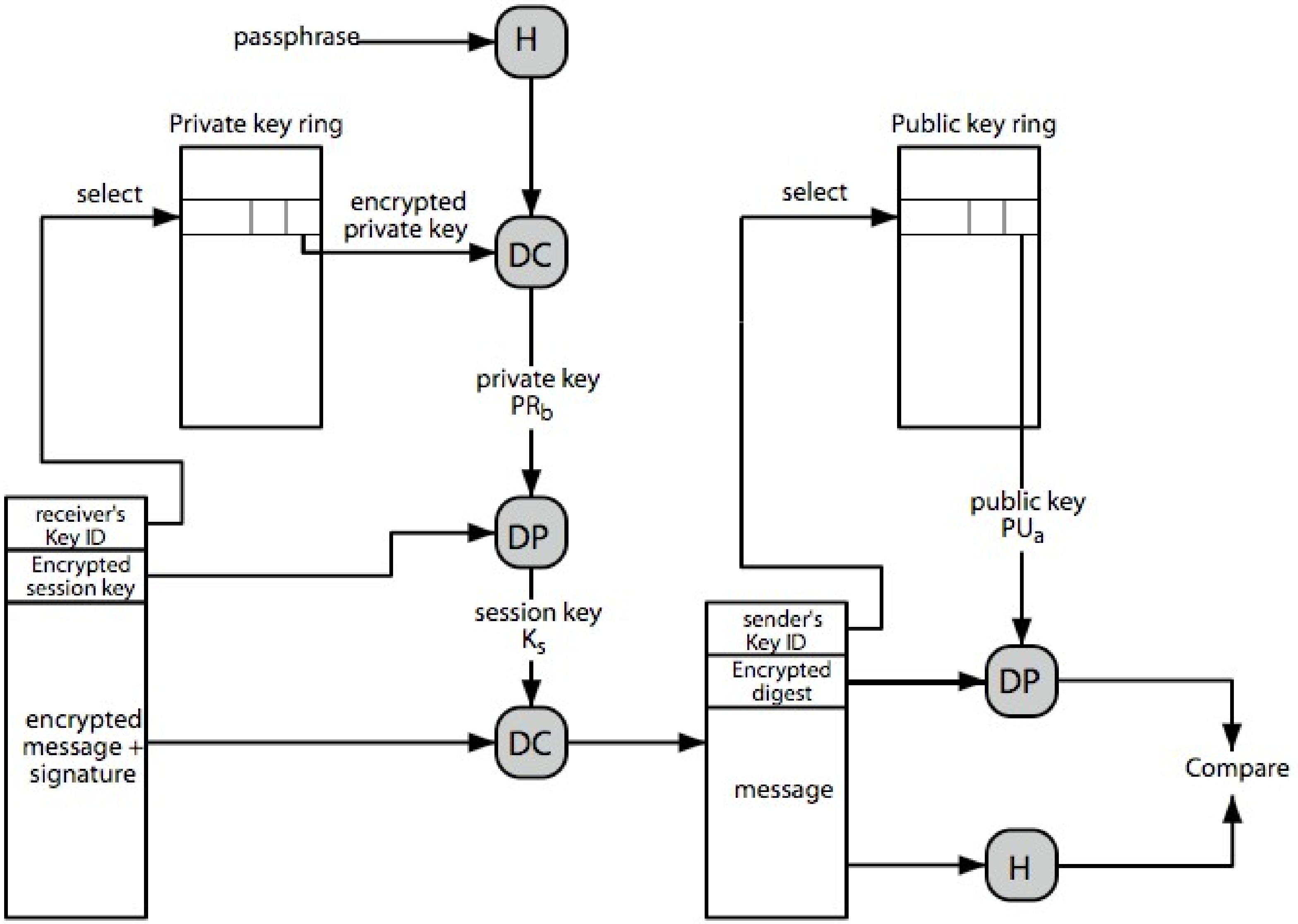
- 2. Authenticating the message:

  a. PGP retrieves the sender's public key from the public-key ring, using the Key ID 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 Reception



PGP Public Key Management

- Key rings are different from certificate chains used in X.509
  - There the user only trusts CAs and the people signed by the CAs
  - In PGP he or she can trust anyone and can add others signed by people s/he trusted
  - Forms a **"web" of trust**
  - Users can revoke own keys
- Thus, users do not rely on external CAs
  - Every user is his/her own CA

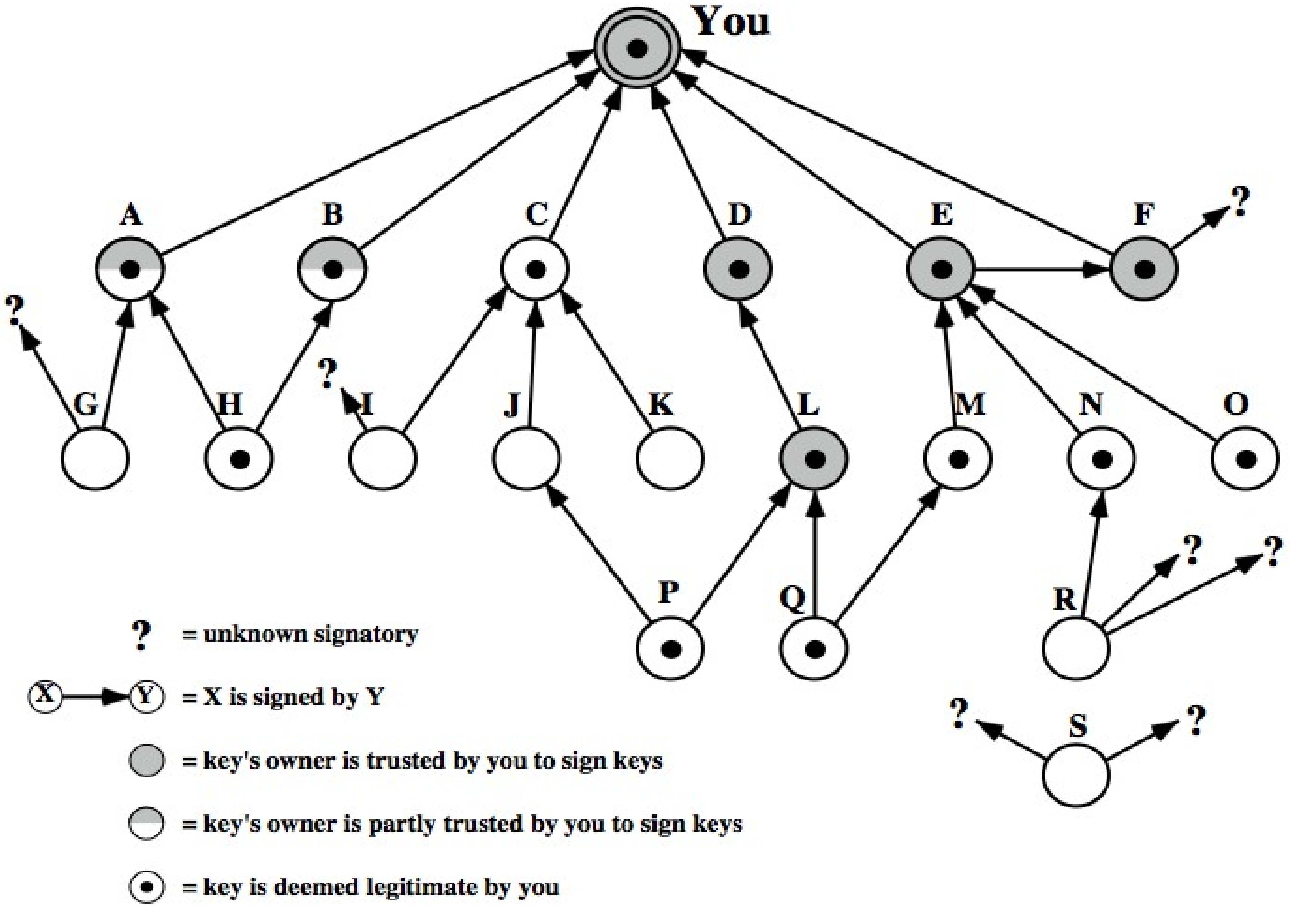
PGP Public key management

- Why is there a problem?
- Protecting public keys from tampering is the most difficult problem in public key applications
- Approaches to minimize the risk that user's public key ring contains false public keys:
  - A physically gets key from B
  - Verify key by phone
  - Get B's public key from mutually trusted individual
  - Obtain B's key from trusted certifying authority

The Use of Trust

- PGP provides a means of using trust, associating trust with public keys and exploiting trust information
- Each entry in the key ring is a public key certificate which has:
  - Key legitimacy field (indicates level of trust)
  - Signature trust field (degree to which user trusts signer)
  - Owner trust field (degree to which public key is trusted to sign other certificates)

PGP Trust Model Example


You
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
?
X
Y
= unknown signatory
= 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

Revoking Public Keys

- The owner issues a key revocation certificate.
- Normal signature certificate with a revoke indicator.
- Corresponding private key is used to sign the certificate.

GNUPG

GnuPG is the GNU project's complete and free implementation of the OpenPGP standard as defined by RFC4880 (http://www.gnupg.org/)

GnuPG allows to encrypt and sign your data and communication, features a versatile key management system as well as access modules for all kinds of public key directories.

GnuPG, also known as GPG, is a command line tool with features for easy integration with other applications. A wealth of frontend applications and libraries are available. Version 2 of GnuPG also provides support for S/MIME.

Electronic Mail Security

- GNU Privacy Guard (GnuPG)
  - a GPL Licensed alternative to the PGP suite of cryptographic software.
  - compliant with RFC 4880, which is the current IETF standards track specification of OpenPGP.
  - Current versions of PGP are interoperable with GnuPG and other OpenPGP-compliant systems.
  - a part of the Free Software Foundation's GNU software project, and has received major funding from the German government

    http://en.wikipedia.org/wiki/GNU_Privacy_Guard

GNU Privacy Guard (GnuPG) --Usage--

- **Linux: integrated into KMail and Evolution, the graphical e-mail clients found in GNOME (Application: Seahorse) and in KDE (Application: KGPG), the most popular Linux desktops.**

- **Seahorse : GNOME front-end application for managing PGP and SSH keys. Seahorse integrates with Nautilus, gedit and Evolution for encryption, decryption and other operations.**

  Screenshots: http://support.real-time.com/open-source/seahorse/index.html

- **KGpg : KDE graphical frontend for GnuPG, which includes a key management window and an editor. Users can easily create cryptographic keys, and write, encrypt, decrypt, sign, or verify messages http://utils.kde.org/projects/kgpg/**

- **MacGPG: a GnuPG front end for Mac OS X http://sourceforge.net/projects/macgpg/**

- **Gpg4win : GNU Privacy Guard for Windows http://gpg4win.org/**
- **More info about GUI, Mail User Agent, Chat, Network related front-ends, etc**

  http://www.gnupg.org/related_software/frontends.html

GnuPG Demo

- **Creating GPG Keys**

  http://fedoraproject.org/wiki/Creating_GPG_Keys

  GnuPG Commands – Examples

  http://www.spywarewarrior.com/uiuc/gpg/gpg-com-4.htm

  Checking integrity of file downloaded

  Examine the following page to study md5sum or sha1sum:

  http://linuxsagas.digitaleagle.net/2012/05/11/checking-files-with-md5sum-or-sha1sum/

- **Origin and integrity of the file can be ensured by verifying its signature**

  Details can be found from

  https://www.torproject.org/docs/verifying-signatures.html.en

  One can perform these operations on a file and its signature downloaded from

  https://www.torproject.org/download/download-easy.html.en

Gnupg Examples

http://www.gnupg.org/gph/en/manual/x110.html

- Generation of GPG key

  gpg --gen-key

- Priting GPG key's simplified hash value

  gpg --fingerprint jqdoe@example.com

- Sending/uploading GPG keys to public key server

  gpg --keyserver hkp://subkeys.pgp.net --send-key KEYNAME

  Here at the beginning of KEYNAME, which is key ID and learnt from the output of the previous command.

- Listing all keys on the user's public ring

  gpg --list-keys

- Adding new public key to the user's public ring.
- ***gpg --import blake.gpg***

Gnupg Örnekler

- Signing the file doc with GPG key to get its digital signature

  gpg --output doc.sig --sign doc

- Verifying the signature of the file doc using the GPG key

  gpg --output doc --decrypt doc.sig

- Performing GPG encryption

  gpg --output doc.gpg --encrypt --recipient blake@cyb.org doc

  Here the receiver: blake@cyb.org ***; doc*** file is going to be encrypted ; for this encryption, a secret (session) key of symmetric encryption is created and then it will be encrypted by the public key of the recipient .

- Performing GPG decryption (by ***blake@cyb.org***)

  gpg --output doc --decrypt doc.gpg

  blake@cyb.org, uses his passphrase secret key to decrypt his private key. Then he decrypts the encrypted secret (session) key using his private key. This session key is used to decrypt ***doc.gpg*** in order to recover the file doc.

- The following command is used to encrypt the file doc. Encrypted file is stored in the host PC.

  gpg --output doc.gpg --symmetric doc

Enigmail: Thunderbird Addon

- Enigmail is a security extension to Mozilla Thunderbird and Seamonkey. It enables you to write and receive email messages signed and/or encrypted with the OpenPGP standard.

  Sending and receiving encrypted and digitally signed email is simple using Enigmail.

  https://www.enigmail.net/home/index.php
- Screenshoots

https://www.enigmail.net/documentation/screenshots.php

- Installation and Usage

https://addons.mozilla.org/en-US/thunderbird/addon/enigmail/

http://fedoraproject.org/wiki/Using_GPG_with_Thunderbird