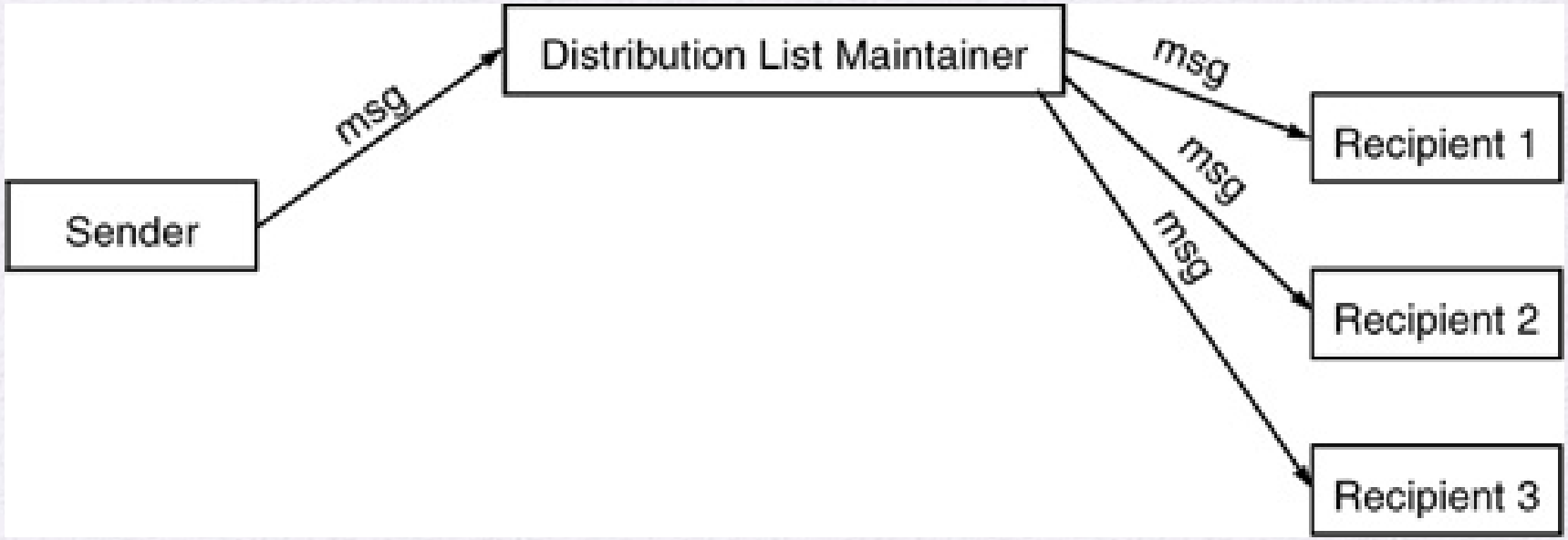
Pretty Good Privacy (PGP)

- Provides a confidentiality and authentication service that can be used for electronic mail and file storage applications
- Developed by Phil Zimmermann
  - Selected the best available cryptographic algorithms as building blocks
  - Integrated these algorithms into a general-purpose application that is independent of operating system and processor and that is based on a small set of easy-to-use commands
  - Made the package and its documentation, including the source code, freely available via the Internet, bulletin boards, and commercial networks
  - Entered into an agreement with a company to provide a fully compatible, low-cost commercial version of PGP

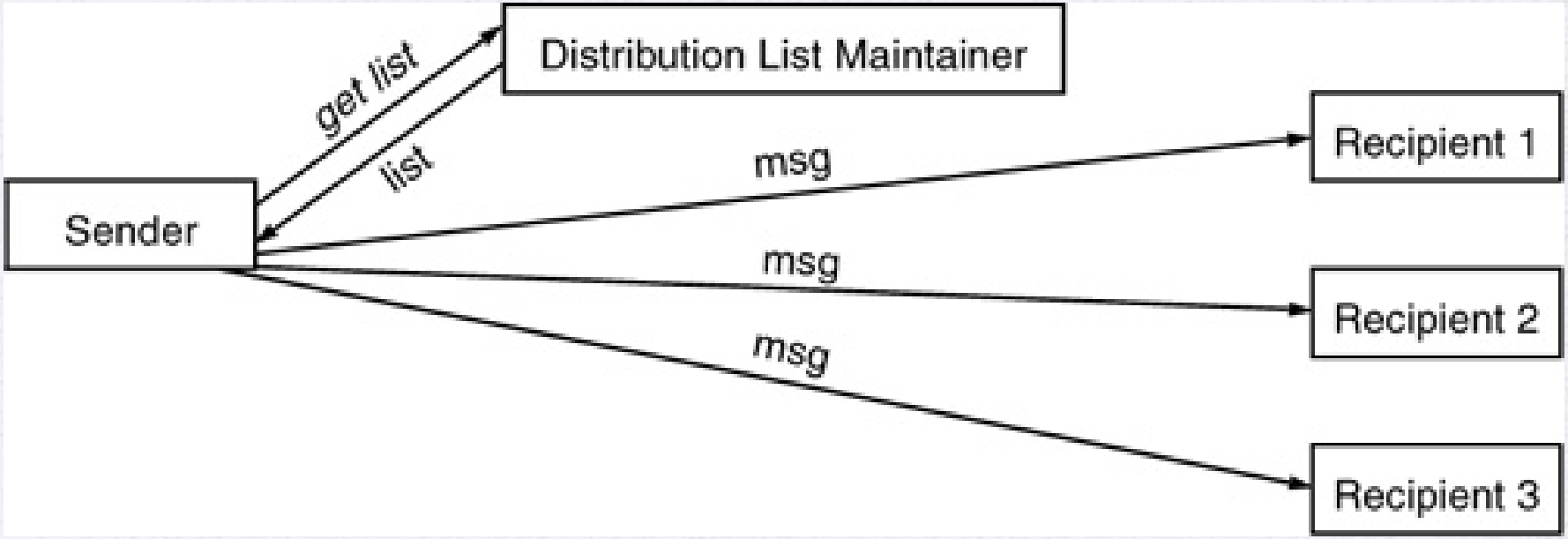
Distribution List

- user to send a message to one or more recipients
- mail is often sent to a distribution list.

Remote Exploder



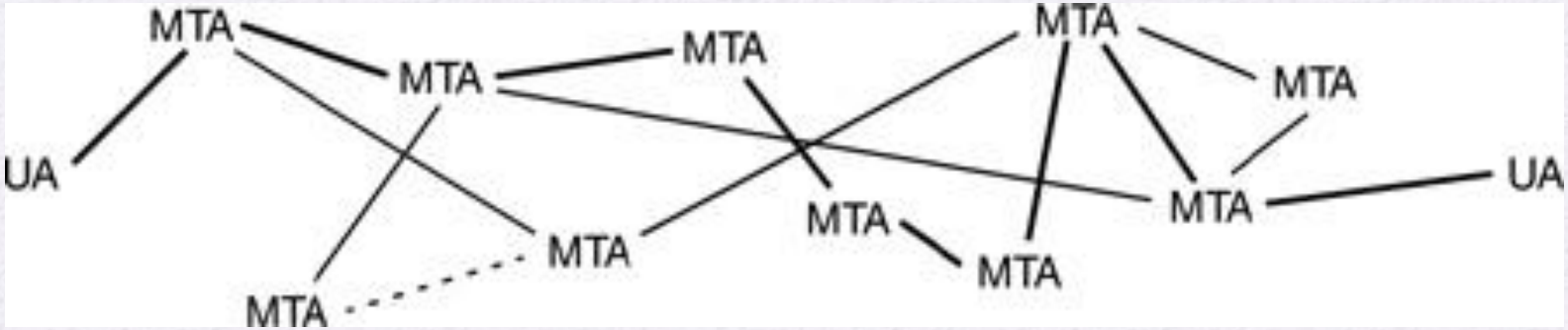
Local Explder



Store & Forward

- it is necessary for both the source and destination machines to be running, and reachable from each other on the network. This might be especially inconvenient if the user machines are only occasionally connected.

MTA-msg t/f agent UA-user agent



- **Privacy** ability to keep anyone but the intended recipient from reading the message. **Authentication** assurance to the recipient of the identity of the ender
- **Integrity** assurance to the recipient that the message has not been altered since it was transmitted by the sender

- **non-repudiation** the ability of the recipient to prove to a third party that the sender really did send the message.
- proof of submission verification given to the sender that the message was handed to the mail delivery system
- proof of delivery verification that the recipient received the message.

- message flow confidentiality: an extension of privacy such that Carol not only cannot know the content of the message Alice sent Bob, but cannot even determine whether Alice sent Bob a message.

- Anonymity the ability to send a message so that the recipient can't find out the identity of the sender.
- Containment the ability of the network to keep certain security levels of information from leaking out of a particular region.
- **Audit** the ability of the network to record events that might have some security relevance
- Accounting the ability of the mail system to maintain system usage statistics

- self destruct: Snapmail does not allow to forward.
- message sequence integrity

Privacy of Email

- Most people think that electronic mail is private.
- Ways to read your messages.
  - An eavesdropper can listen to the message while it is being transmitted
  - Relay nodes (routers or mail forwarders) might have software to store messages and divulge them to people

End-to-End Privacy

- if Alice wants to send Bob an encrypted message, she encrypts it using Bob's public key.
- If Alice has a long message to send to multiple recipients.

- Bob's name; $K_{Bob}\{S\}$
- Carol's name; $K_{Carol}\{S\}$
- Ted's name; $K_{Ted}\{S\}$
- $S\{m\}$

Privacy with Distribution List Exploders

- Alice is sending a message to a distribution list which will be remotely exploded.
- Alice will choose a random per-message secret key S, and encrypt the message with S. The distribution list exploder will decrypt S, and re-encrypt S with the key for each recipient

Authentication of the Source

- **Source Authentication Based on Public Key Technology**
- Assuming Bob knows Alice's public key, Alice can digitally sign the message, using her private key, which will assure Bob that Alice wrote the message. The method usually chosen to sign a message is for Alice to first compute a hash of the message (for instance using MD5), and then to sign the message digest

- To: Bob
- Subject: I got PEM working
- Date: Fri, 01 Apr 94 10:12:37 -0400
- From: Alice
- -----BEGIN PRIVACY-ENHANCED MESSAGE-----
- Proc-Type: 4,MIC-CLEAR
- Content-Domain: RFC822
- DEK-Info: DES-CBC,31747476B4831B1D
- Originator-ID-Asymmetric: MEMxCzAJBgNVBAYTAlVTMSYwJAYDVQQKEx1EaWd
- pdGFsIEVxdWlwbWVudCBDb3Jwb3JhdGlvbjEMMAoGA1UECxMDTEtH,02
- MIC-Info: RSA-MD5,RSA,u1OHP1RwLqePAoaN5nPk9W7Q2EfjaP+yDBSaLyMcSgc
- MK2YicGSAqLz47Ol+TUR4YmMD/JnHMtsCJerV2QFtFQ==

- ... and I'm sending this message with PEM.
- -----END PRIVACY-ENHANCED MESSAGE-----

Source Authentication Based on Secret Keys

- If Alice has a shared key with Bob, then she can reassure Bob that she is Alice by proving she knows the shared secret key. She does this by performing some cryptographic computation on the message using the shared secret key.

Source Authentication with Distribution Lists

- With public keys, source authentication is easy, even with distribution lists
- With secret keys it is more complicated. We can't assume Alice will share a secret key with every recipient

Message Integrity

- When Bob receives a message from Alice, how does he know that Carol did not intercept the message and modify it? For instance, she might have changed the message
- Fire Carol immediately to Promote Carol immediately.

Message Integrity without Source Authentication

- **Message Integrity without Source Authentication:**
  - If the message is encrypted (where the sender knows the public key of the recipient), it is possible to do something that could be considered integrity protection without source authentication

Non Repudiation

- **Repudiation** is the act of denying that you sent a message
- it means that if Alice sends a message to Bob, Alice cannot later deny that she sent the message. Bob can prove to a third party that Alice did indeed send the message

Non-Repudiation Based on Public Key Technology

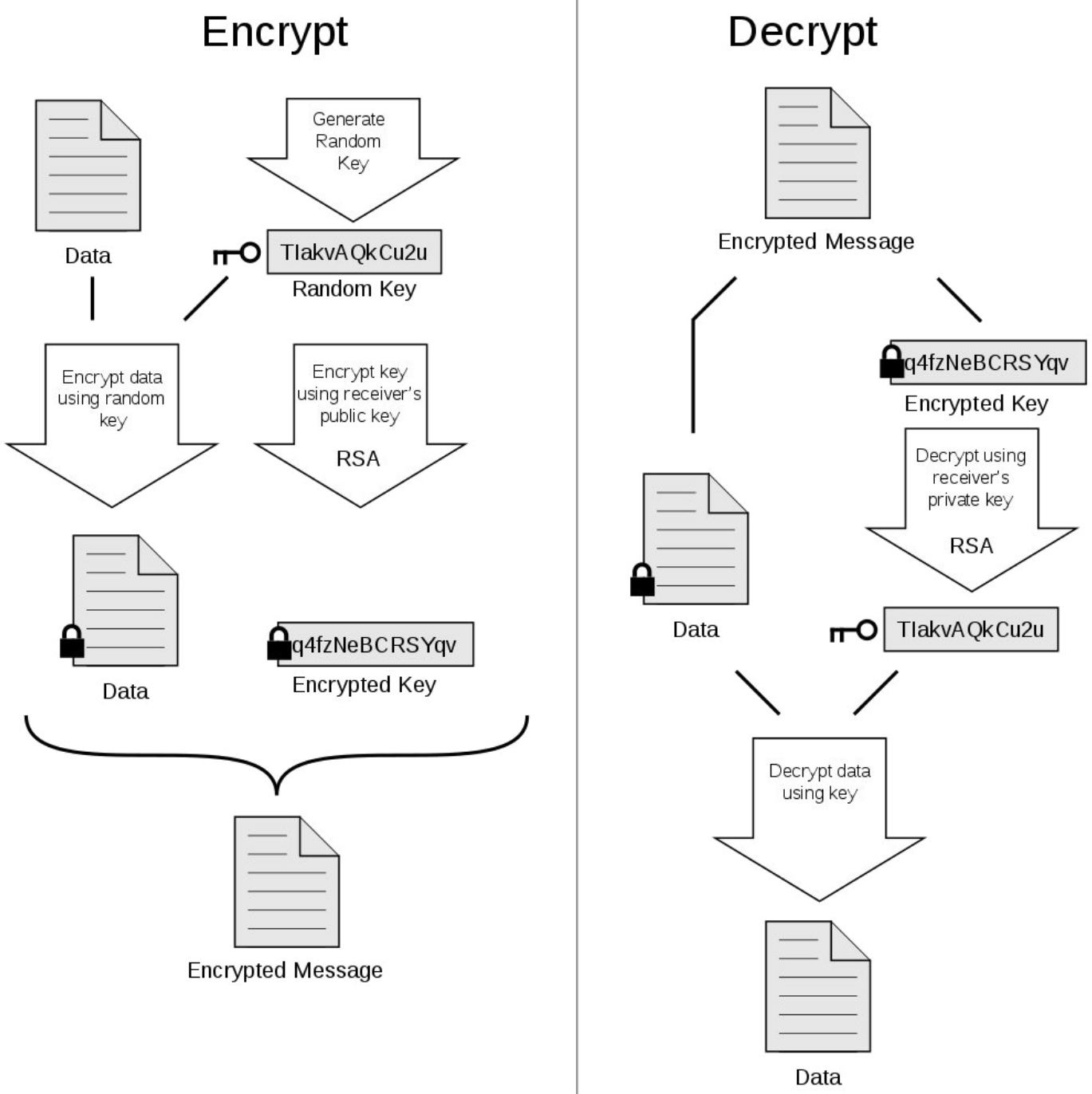
- **Non-Repudiation Based on Public Key Technology:**
  - Use public key signature on the message digest of the message, using her private key

  Non-Repudiation with Secret Keys

  Alice sends the message to N, and does source authentication with N so that N knows the message came from Alice. N does some computation on the message and Alice's name using a secret quantity S_N that N shares with nobody.

PGP Growth

- It is available free worldwide in versions that run on a variety of platforms
- The commercial version satisfies users who want a product that comes with vendor support
- It is based on algorithms that have survived extensive public review and are considered extremely secure
- It has a wide range of applicability
- It was not developed by, nor is it controlled by, any governmental or standards organization
- Is now on an Internet standards track, however it still has an aura of an antiestablishment endeavor


Encrypt
Data
Generate Random Key
TIakvAQkCu2u
Random Key
Encrypt data using random key
Encrypt key using receiver's public key
RSA
Data
q4fzNeBCRSYqv
Encrypted Key
Encrypted Message
Decrypt
Encrypted Message
q4fzNeBCRSYqv
Encrypted Key
Decrypt using receiver's private key
RSA
Data
TIakvAQkCu2u
Decrypt data using key
Data

PGP

- Notation

K_s = session key	H = hash function
KP_a = public key of user A	KP_b = public key of user B
KS_a = private key of user A	KS_b = private key of user B
E = conventional encryption	D = conventional decryption
E_p = public-key encryption	D_p = public-key decryption
Z = compression using zip algorithm	Z^{-1} = decompression
$\|\|$ = concatenation	

Electronic Mail Security

- **PGP (**Pretty Good Privacy)
  - Use
    - Digital signature
    - Encryption
    - Compression (zip)
    - Radix-64 conversion

PGP

Confidentiality via Encryption

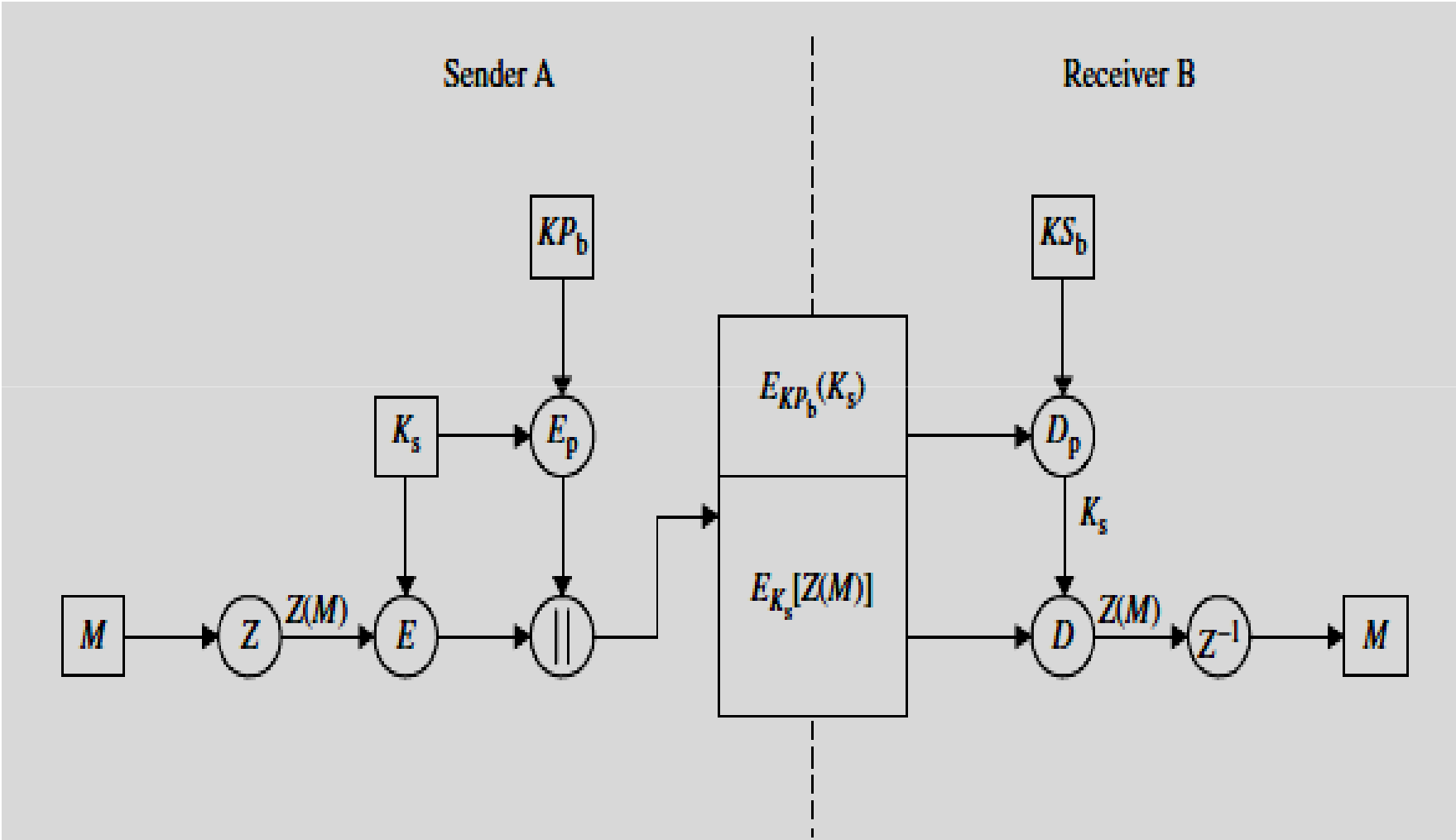


Figure 9.1 PGP confidentiality computation scheme with compression/decompression Algorithms.

PGP

- **Authentication via Digital Signature**

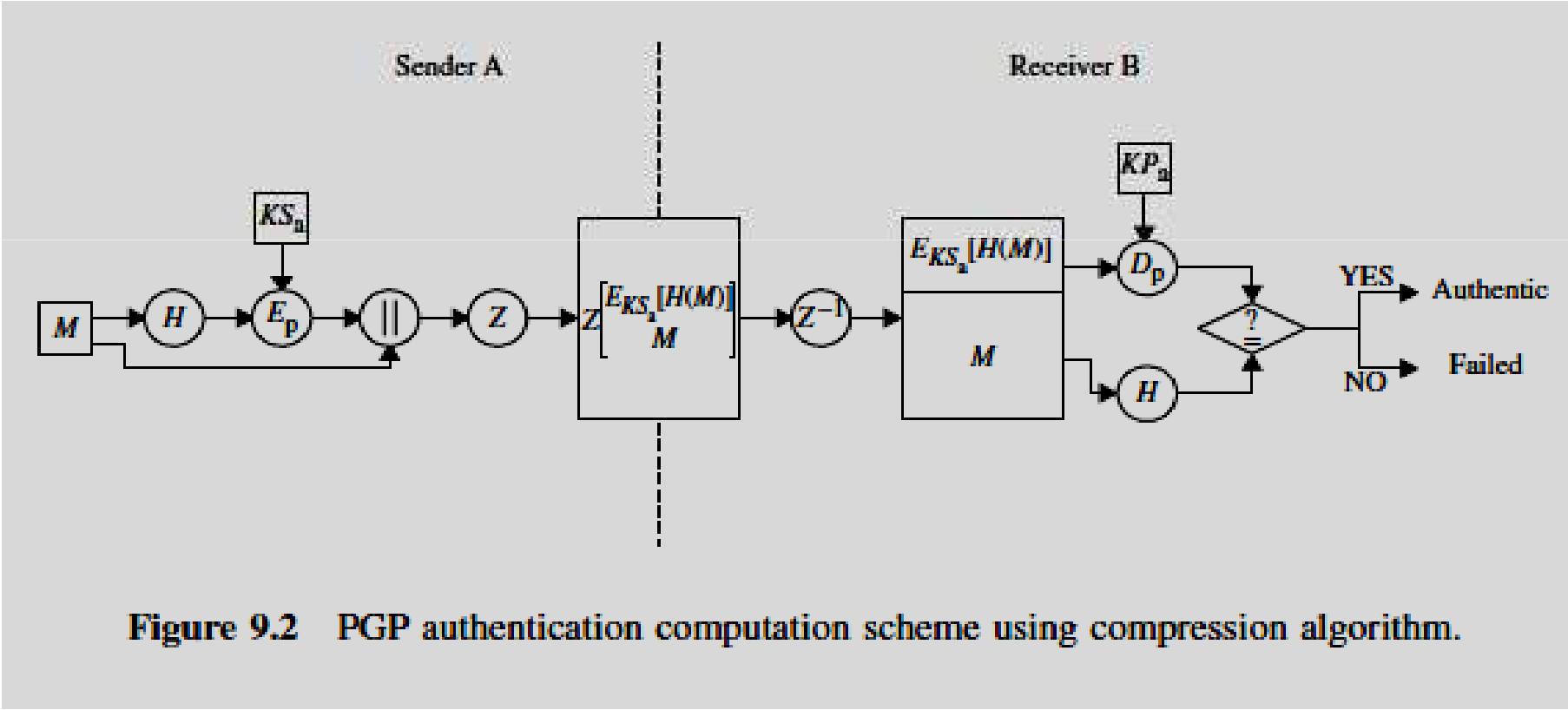


Figure 9.2 PGP authentication computation scheme using compression algorithm.

PGP

- **Compression**
  - *Z compression*
  - *Z*−1 decompression
  - Saving space
    - e-mail transmission
    - file storage
  - Compression will add difficulty- different trade-offs in running speed versus compression ratio produce different compressed forms

PGP

- **Radix-64 Conversion:**

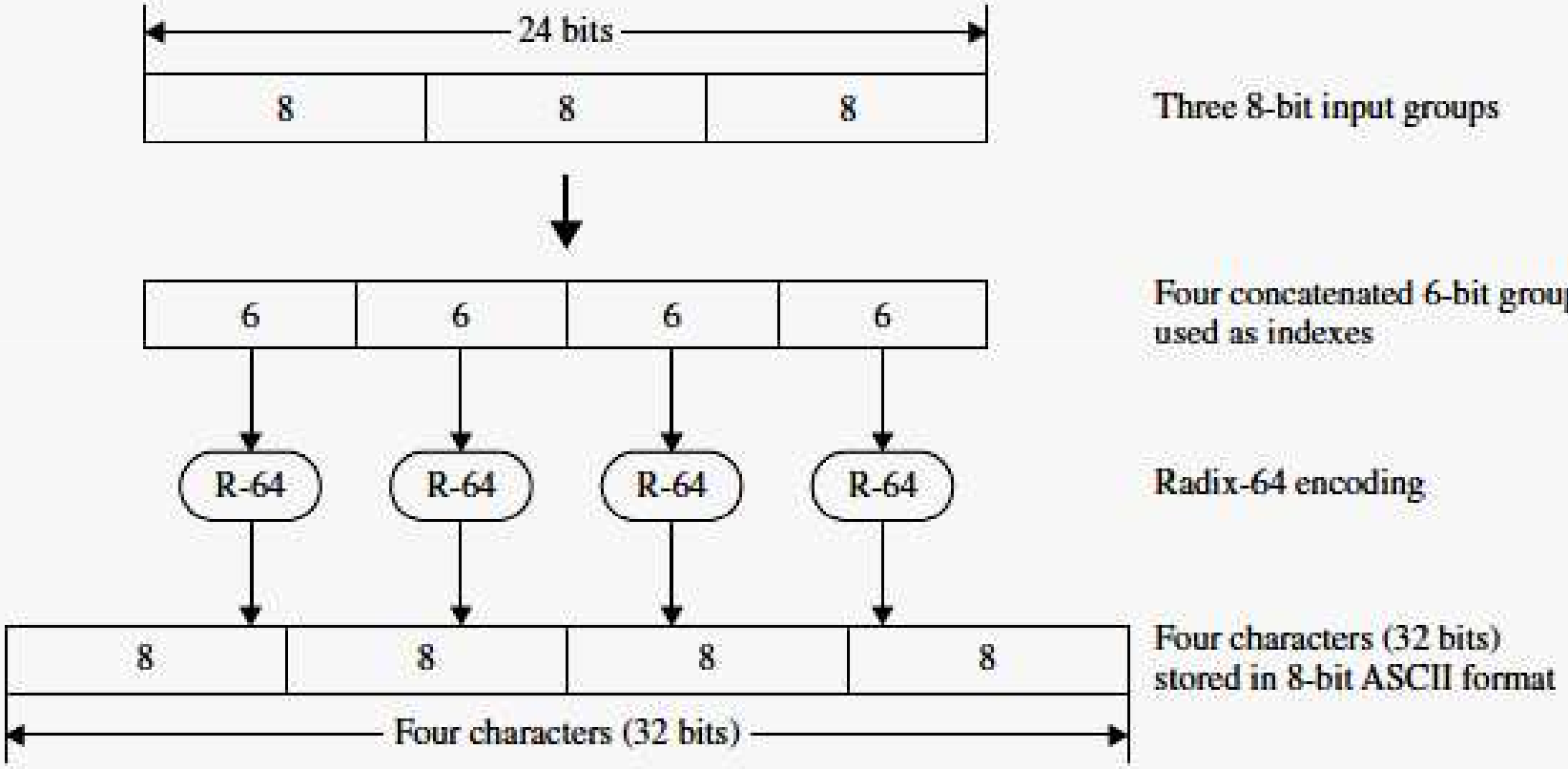


Figure 9.3 Radix-64 printable encoding of binary data.

PGP

- **Radix-64 Conversion**

Table 9.1 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)	=

PGP

- **Radix-64 Conversion**

Raw text	b2	63	29	
24-bit raw text:	10110010	01100011	00101001	
Arranging in blocks of 6 bits	101100	100110	001100	101001
Decimal values are	44,	38,	12,	41
Referring to Table				
Radix-64 encoding	s	m	M	p
ASCII format - hexadecimal	73	6d	4d	70
Binary	0111 0011	0110 1101	0100 1101	0111 0000

PGP

- **Radix-64 Conversion:** *ASCII Armor Format*
- ASCII Armor:
  - Armor head line
  - Armor headers
  - Blank line
  - ASCII-Armored data
  - Armor checksum
  - Armor tail

Example of an ASCII Armored Message

```
-----BEGIN PGP MESSAGE-----
Version: OpenPrivacy 0.99

yDgBO22WxBHv7O8X7O/jygAEzol56iUKiXmV+XmpCtmpqQUKiQrFqclFqUDBovzS
vBSFjNSiVHsuAA==
=njUN
-----END PGP MESSAGE-----
```

PGP

- **Algorithms for PGP 5.x**

Public-Key Algorithms

ID	Algorithm
1	RSA (encrypt or sign)
2	RSA encryption only
3	RSA sign only
16	ElGamal (encrypt only)
17	DSA (DSS)
18	Reserved for elliptic curve
19	Reserved for ECDSA
20	ElGamal (encrypt or sign)
21	Reserved for Diffie–Hellman
100–110	Private/experimental algorithm

PGP

- **Algorithms for PGP 5.x**

Symmetric-Key Algorithms

ID	Algorithm
0	Plaintext or unencrypted data
1	IDEA
2	Triple DES (DES–EDE)
3	CAST 5 (128-bit key)
4	Blowfish (128-bit key, 16 rounds)
5	SAFER-SK128 (13 rounds)
6	Reserved for DES/SK
7	Reserved for AES (128-bit key)
8	Reserved for AES (192-bit key)
9	Reserved for ASE (256-bit key)
100–110	Private/experimental algorithm

PGP

- **Algorithms for PGP 5.x**

Compression Algorithm

ID	Algorithm
0	Uncompressed
1	ZIP (RFC 1951)
2	ZLIB (RFC 1950)
100–110	Private/experimental algorithm

PGP

- **Algorithms for PGP 5.x**

Hash Algorithms

ID	Algorithm
1	MD5
2	SHA-1
3	RIPE-MD/160
4	Reserved for double-width SAH (experimental)
5	MD2
6	Reserved for TIGER/192
7	Reserved for HAVAL (5 pass, 160-bit)
100–110	Private/experimental algorithm