Network Security

Acknowledgement: Slides from Cunsheng Ding, revised by Mahendra Pratap Singh

Electronic Mail Security

Outline of this Lecture

1. Email security issues.
2. Detailed introduction of PGP.

About Electronic Mail

1. In virtually all distributed environment, electronic mail is one of the most heavily-used network-based applications.
2. It is also a distributed application that is widely used across all architectures and platforms (PC, UNIX, Macintosh, etc).

Consequence: With the explosively growing reliance on electronic mail, there is a growing demand for authentication and confidentiality services.

Developing a System for Electronic Mail Security

Having learned the basics of ciphers, digital signature, and authentication, you are asked to design a system to support the following for electronic email communication:

1. confidentiality of message;
2. nonrepudiation of the sender; and
3. authentication of message.

Question: How do you design your system?

Developing a System for Electronic Mail Security

Answer: You need to carry out the following:

1. Select the best available cryptographic algorithms as building blocks; and
2. integrate 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.

This is how PGP and S/MIME were developed.

PGP: Pretty Good Privacy

S/MIME: Secure/Multipurpose Internet Mail Extension

PGP: Pretty Good Privacy

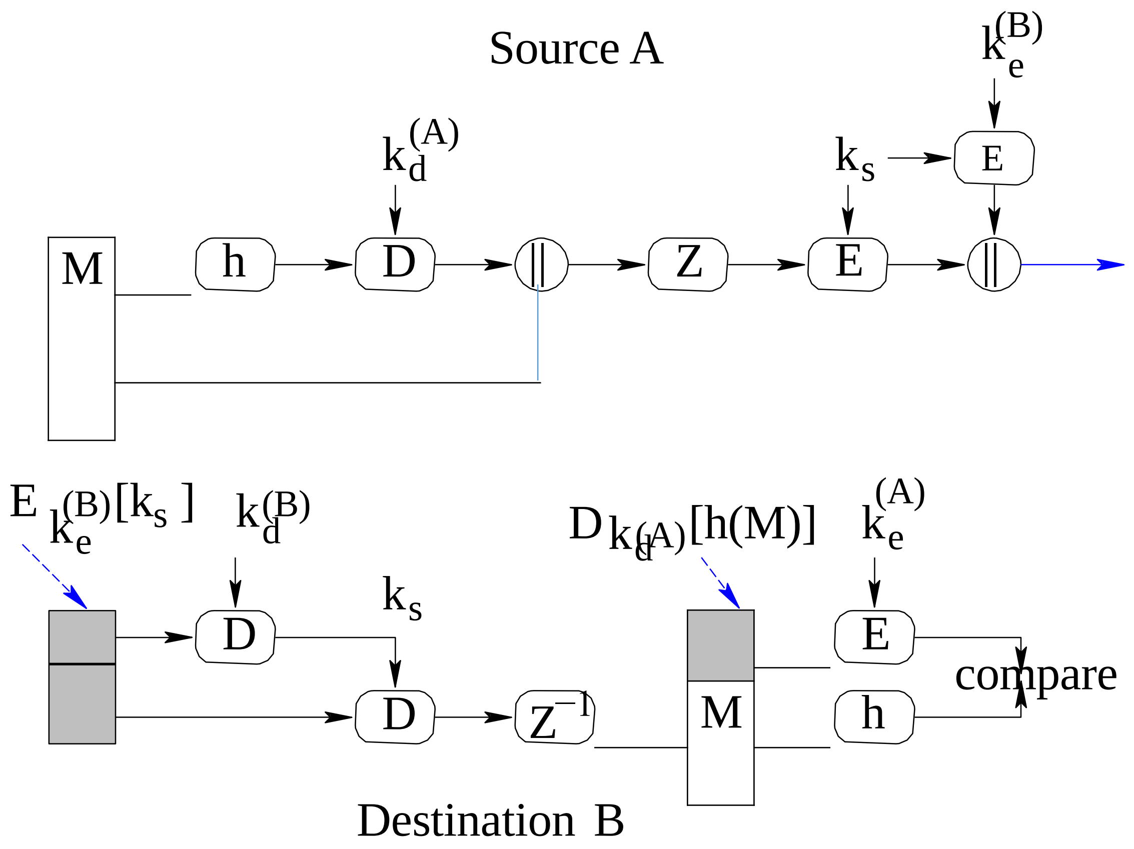
1. It is a program for email communication security.
2. Phil Zimmermann started writing PGP in the mid 1980s and finished the first version in 1991.
3. It is available free worldwide in versions than runs on a variety of platforms, including DOS/Windows, UNIX, Macintosh, and many more.
4. It is based on cryptographic algorithms that have survived extensive public review.
5. It has a wide range of applicability: within corporations and for individuals within themselves.

A Summary of PGP Services

1. Nonrepudiation and authentication (Digital signature using DSS/SHA or RSA/SHA).
2. Message confidentiality (encryption with CAST or IDEA or 3DES, and session key encryption with ElGamal or RSA).
3. Compression (using ZIP) – A message may be compressed, for storage or transmission.
4. Email compatibility (using radix-64 conversion):

   To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix-64 conversion.
5. Segmentation – to accommodate maximum message size limitations, PGP performs segmentation and reassembly.

Authentication, Confidentiality, Nonrepudiation in PGP



DSS/SHA-2 or RSA/SHA-2, Z = ZIP algorithm, RSA or ElGamal, CAST-128 or IDEA or 3DES or AES. k_s the session key.

Compression in PGP (1)

Why compression? Save space both for email transmission and for file storage, and for enhancing security.

Placement of compression: After applying the signature, but before encryption. Z indicates compression and Z^{-1} decompression.

Why should Z be before encryption? Compression reduces the redundancy of messages and makes cryptanalysis more difficult!

Why signature before compression? Left to you.

Comment: It is interesting to note that finding the right placement of a building block is quite important for the whole system!

Remark: Details of ZIP are available on the Internet.

Email Compatibility

The problem: When PGP is used, at least part of the block to be transmitted is encrypted, consists of a stream of arbitrary 8-bit octets. However, many electronic mail systems only permits the use of blocks consisting of ASCII text.

Solution: 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 is the "Radix-64 conversion".

Comment: The use of Radix-64 conversion expands a message by 33%. Fortunately, the compression should be more than enough to compensate for the Radix-64 conversion.

Remark: Details of Radix-64 are available on the Internet.

Radix-64 Conversion in PGP

- Many electronic mail systems can only transmit blocks of ASCII text.
- This can cause a problem when sending encrypted data since ciphertext blocks might not correspond to ASCII characters which can be transmitted.
- PGP overcomes this problem by using **radix-64 conversion**.

PGP E-Mail Compatibility: Example

- Suppose the email message is: new
- ASCII format: 01101110 01100101 01110111
- After encryption: 10010001 10011010 10001000
- The problem after encryption:
  - The three bytes do not represent any key board ASCII characters.
  - Most email systems cannot transmit and process such a piece of ciphertext.

Radix-64 Conversion

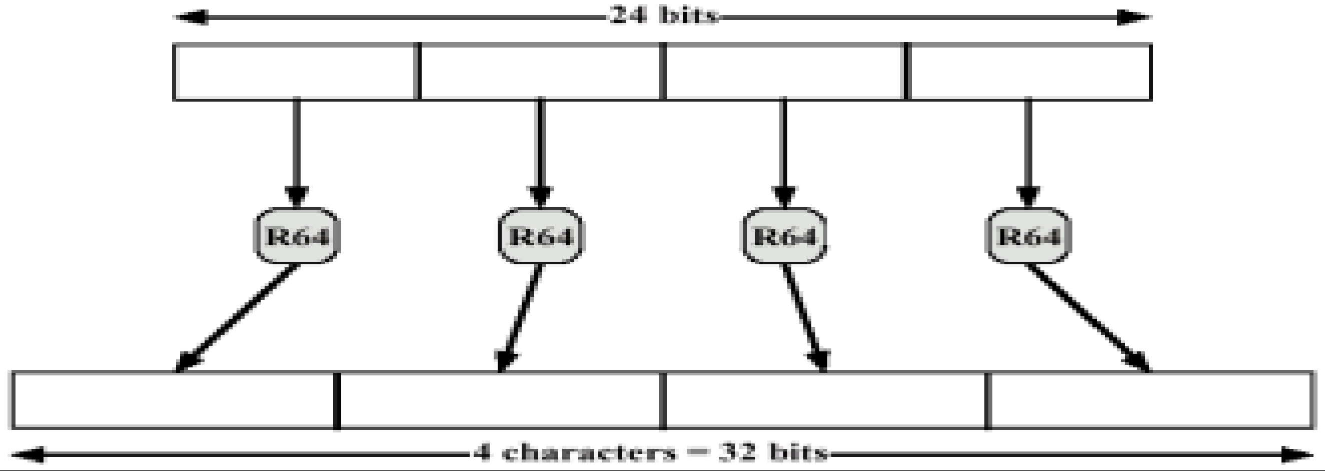
- ❑Convert the text to be encrypted into binary using ASCII coding.
- ❑ Encrypted the binary to give a ciphertext stream of binary.
- ❑ Radix-64 conversion maps arbitrary binary into printable characters as follows:

1. The binary input is split into blocks of 24 bits (3 bytes).
2. Each 24 bits block is then split into four sets each of 6-bits.
3. Each 6-bit set will then have a value between 0 and 2^6-1 (=63).
4. This value is encoded into a printable character.

Pictorial Description

Radix-64 Conversion

- To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII string using radix-64 conversion
- Radix-64 expands a message by 33%



1

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)	=

Radix-64 Conversion: Example

- Suppose the email message is: new
- ASCII format: 01101110 01100101 01110111
- After encryption: 10010001 10011010 10001000
- The Radix-64 conversion:
  - The 24-bit block: 10010001 10011010 10001000
  - Four 6-bit blocks: 100100 011001 101010 001000
  - Integer version: 36 25 38 8
  - Printable version: k Z m I

Segmentation and Reassembly

The problem: Email facilities often are restricted to a maximum message length (e.g., 50, 000 octets). Any message longer than that must be broken into smaller segments, each of which is mailed separately.

Solution: To accommodate this restriction, PGP automatically subdivides a message that is too large into segments that are small enough to send via email.

When is segmentation done? After all of the other processing, including the Radix-64 conversion.

Reassembly: The session key component and signature component appear only once, at the beginning of the first segment. At the receiving end, PGP must strip off all email headers and reassemble the entire original block before performing the steps illustrated in the figure of the previous page.

Keys used in PGP

1. One-time session keys.
2. Public and private keys.
3. Passphrase-based keys.

Key Requirements in PGP

- A means of generating unpredictable session keys is needed.
- A user is allowed to have multiple public/private key pairs.

  (A user may wish to have multiple key pairs at a given time to interact with different groups of correspondents or simply to enhance security by limiting the amount of material encrypted with any one key.)

  Hence there is not a one-to-one correspondence between users and their public keys.
- Each PGP entity must maintain a file of its own public/private key pairs as well as a file of public keys of correspondents.

Session Key Generation

Definition: Each is associated with a single message and is used only for encrypting and decrypting that message using a symmetric cipher.

Symmetric ciphers: CAST-128, IDEA (128-bit key), 3DES (168-bit key), AES.

Session Key Generation: Using CAST-128 (block size 64) as example

$$k_S = \text{CAST128}_{CFB}(k, N),$$

where k is a 128-bit key for CAST-128, and $N = N_2||N_1$ are two 64-bit blocks. All three (k, N_1, N_2) are based on a keystroke input from the user. N is encrypted using CAST-128 in CFB mode.

Remark: No need to get more details of the session key generation.

Key Identifiers (1)

Problem: Recall that A sends $E_{k_e^{(B)}}[k_S]||E_{k_s}[x]$ to B if encryption is needed. But in the system B could have more than one private/public key pairs. How could B know which of his public key was used by A?

Solution 1: Transmit the public key $k_e^{(B)}$ together with that message.

Then B could check that it is indeed one of his public keys.

Disadvantages: But it is a waste of resource, as a public key could have hundreds of digits in length.

Key Identifiers (2)

Problem: Recall that A sends $E_{k_e(B)}[k_s]||E_{k_s}[x]$ to B if encryption is needed. But in the system B could have more than one private/public key pairs. How could B know which of his public key was used by A?

Solution 2: Associate an identifier with each public key that is unique at least within each one user. That is, user ID plus key ID would be sufficient to identify a key uniquely.

Disadvantages: It leads to 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.

Key Identifiers (3)

Problem: Recall that A sends $E_{k_e^{(B)}}[k_S] || E_{k_s}[x]$ to B if encryption is needed. But in the system B could have more than one private/public key pairs. How could B know which of his public key was used by A?

Solution adopted in PGP: ID of a public key $k_e^{(B)}$ is defined to be $k_e^{(B)} \bmod 2^{64}$.

Comments: Hence with very high probability that the IDs of a user's public keys are unique.

Is key ID needed for PGP signature? Yes. Key ID is also included in the component of PGP signature.

Key Rings

Observation: Two key IDs $ID(k_e^{(A)})$ and $ID(k_e^{(B)})$ are included in any PGP message that provides both confidentiality and authentication.

Question: How to store and organize them in a systematic way for efficient and effective use by all parties?

Scheme used in PGP: It provides a pair of data structure 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.

The data structures are referred to, respectively, as the **private-key ring** and **public-key ring.**

Private Key Ring: 1-Row 1-Key Pair

T	key ID*	public key	encrypted private key	user ID*
:	:	:	:	:
T_i	$k_e^{(i)} \bmod 2^{64}$	$k_e^{(i)}$	$E_{h(P_i)}[k_d^{(i)}]$	user i
:	:	:	:	:

The private-key ring can be indexed by either User ID or KeyID.

The private-key ring is stored only on the machine of the user that created and owns the key pair, and is accessible only to that user.

The user selects a passphrase P_i, computes $h(P_i)$ = SHA-2(P_i). The private key is encrypted using part of $h[p_i]$ as the key.

Private Key Ring

More about the passphrase:

1. When a user accesses his/her private key, he/she must supply the passphrase. PGP will retrieve the encrypted private key.
2. When the system generates a new public/private key pair, it also asks the user for the passphrase.

Hence the security of the system depends on that of the password!!!

Public Key Ring: 1-Row per Public Key

Timestamp	key ID$*$	public key	user ID$*$
:	:	:	:
T_i	$k_e^{(i)}$ mod 2^{64}	$k_e^{(i)}$	user i
:	:	:	:

Definition: It is used to store public keys of other users that are known to this user.

Remark: The public-key ring can be indexed by either User ID or Key ID. We will see the need for both means of indexing later.

Public-Key Management in PGP

Comment: PGP is intended for use in a variety of formal and informal environments, no rigid public-key management scheme is set up!

Comment: One should update and verify the correctness of the information in his/her public key rings.

Key Revocation

Why key revocation? Because compromise is suspected or a user wants to avoid the use of the same key for an extended period.

How to do this? The owner issues a key revocation certificate, signed by the owner. This certificate has the same form as a normal signature certificate, but includes an indicator that purpose of this certificate is to revoke the use of this public key.

Remark: The corresponding private key must be used to sign a certificate that revokes a public key.

Comments: An opponent who has compromised the private key of an owner can also issues such a certificate. However, this would deny the opponent as well as the legitimate owner the use of the public key.

Format of Transmitted Messages in PGP

Signature Component (1):

$$D_{k_d^{(A)}}[\text{SHA-1}(M \| T_2)] \| L \| ID(k_e^{(A)}) \| T_2$$

Here M is the message data excluding the header fields (file name and timestamp of the message).

Timestamp: T_2, the time at which the signature was made.

Message digest: SHA-1$(M \| T_2)$

Leading two octets of message digest: L

Key ID of sender's public key:$ID(k_e^{(A)})$

Signature Component (2)

<u>Roles of the building blocks:</u>

Message digest: SHA-1($M|T_2$)

1. Why should T_2 be involved here?

   (Against replay types of attacks.)

2. Why the filename and the timestamp T_1 of the message component are excluded in the computation of the message digest?

   (Ensure that detached signatures are exactly the same as attached signatures prefixed to the message. Detached signatures are calculated on a separate file that has none of the message component header fields.)

Signature Component (3)

Roles of the building blocks:

Leading two octets of message digest: L,

To enable the recipient to determine if the correct public key $(k_e^{(A)})$ 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.

These octets also serve as a 16-bit frame-check sequence for the message, for authentication and error detection.

Format of Transmitted Messages in PGP

Roles of the building blocks:

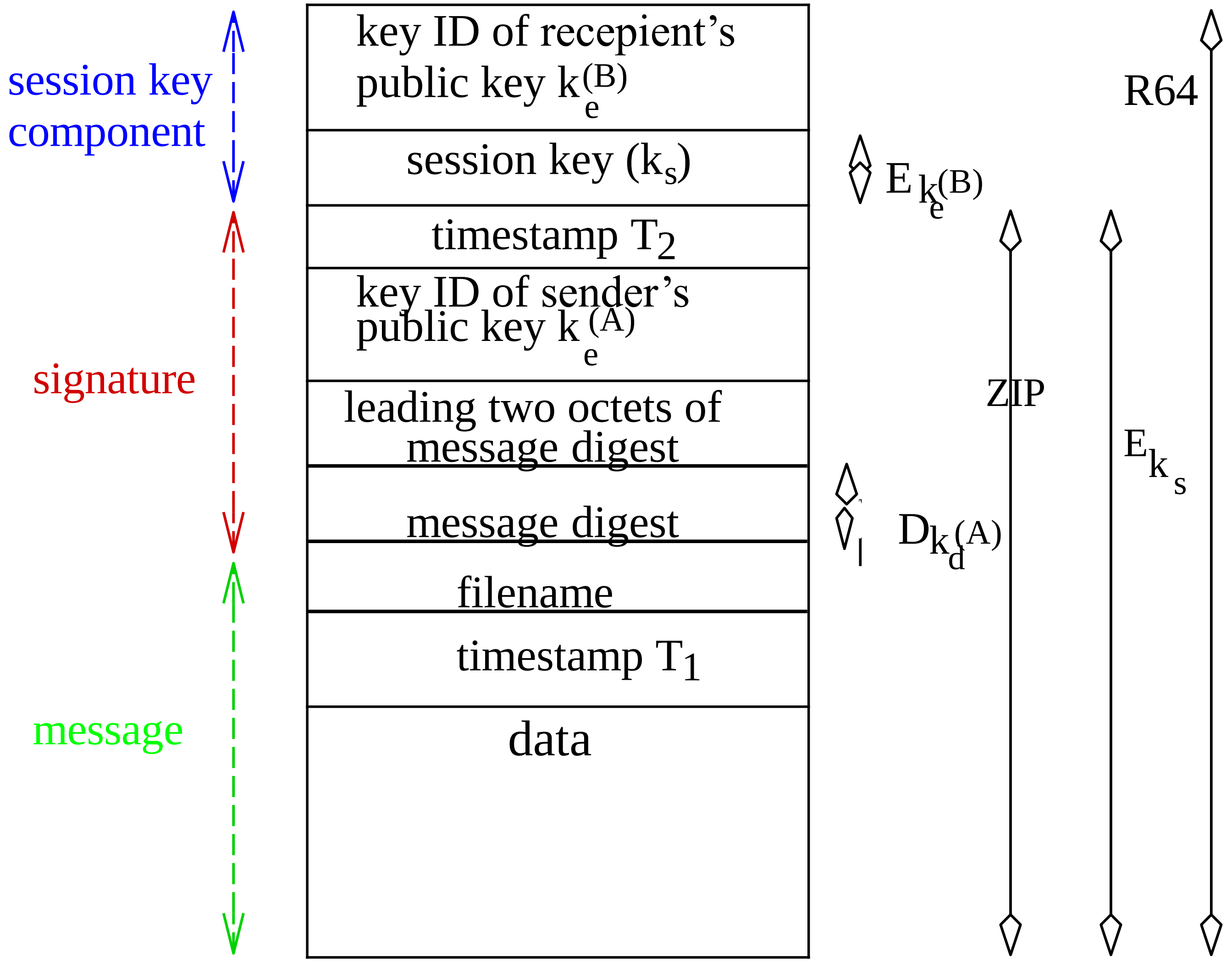
Session Key Component:

$$E_{k_e^{(B)}}[k_S] \| ID(k_e^{(B)})$$

Other operations on the components:

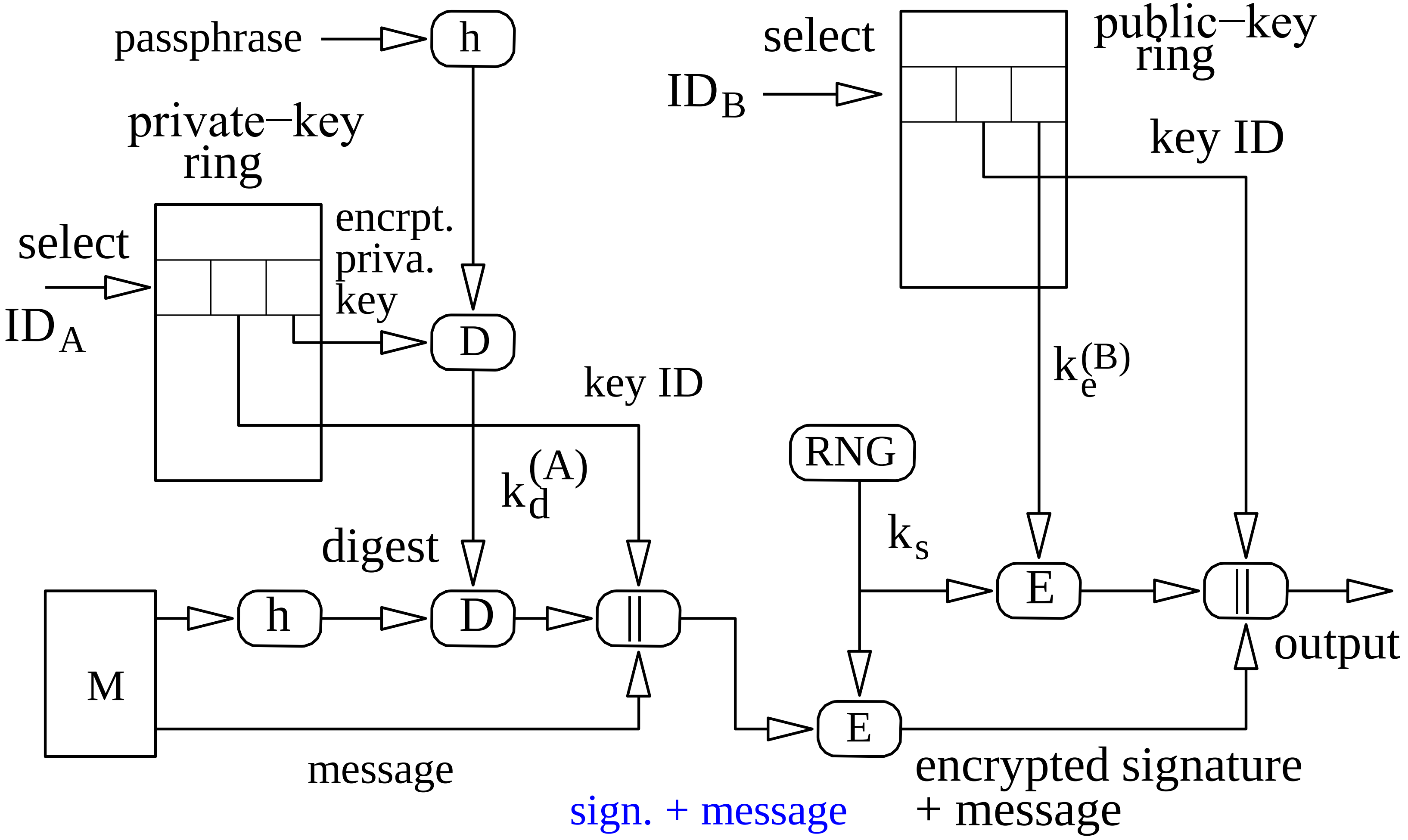
The message component and optional signature component may be compressed using ZIP and may be encrypted using a session key.

Format of Transmitted Messages in PGP: A 1→ B)



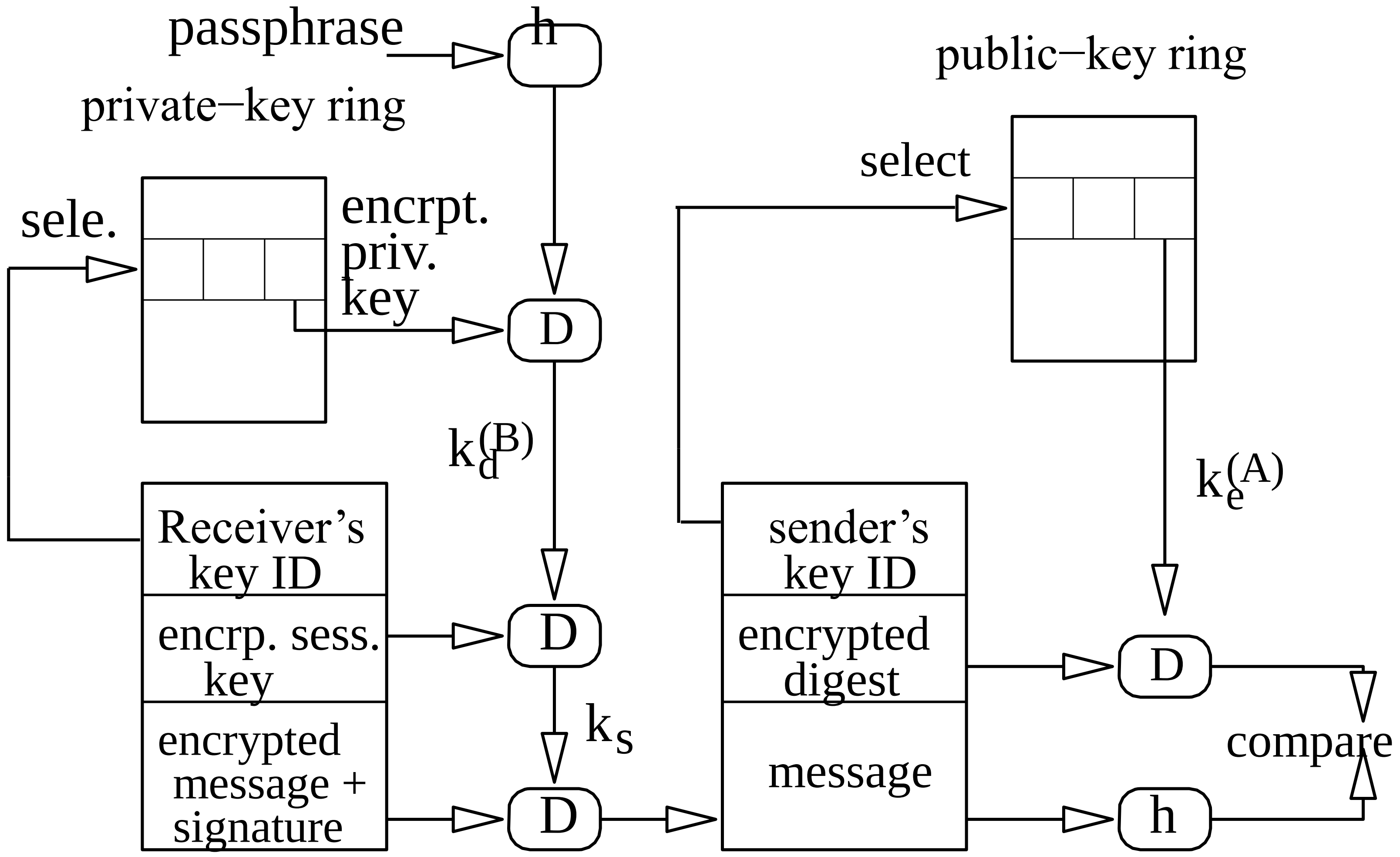
PGP Message Generation

Generation At sender A: ZIP, R64, L, T omitted



PGP Message Reception

At receiver B:ZIP, R64, L, T omitted



Email Systems Supporting PGP

- Use PGP with Pegasus mail
- Use PGP with Simeon (ExacMail)
- Use PGP with Eudora, Outlook
- Use PGP with Herald (WING)
- Use PGP with Pine and ELM on UNIX