IT 422 Network Security

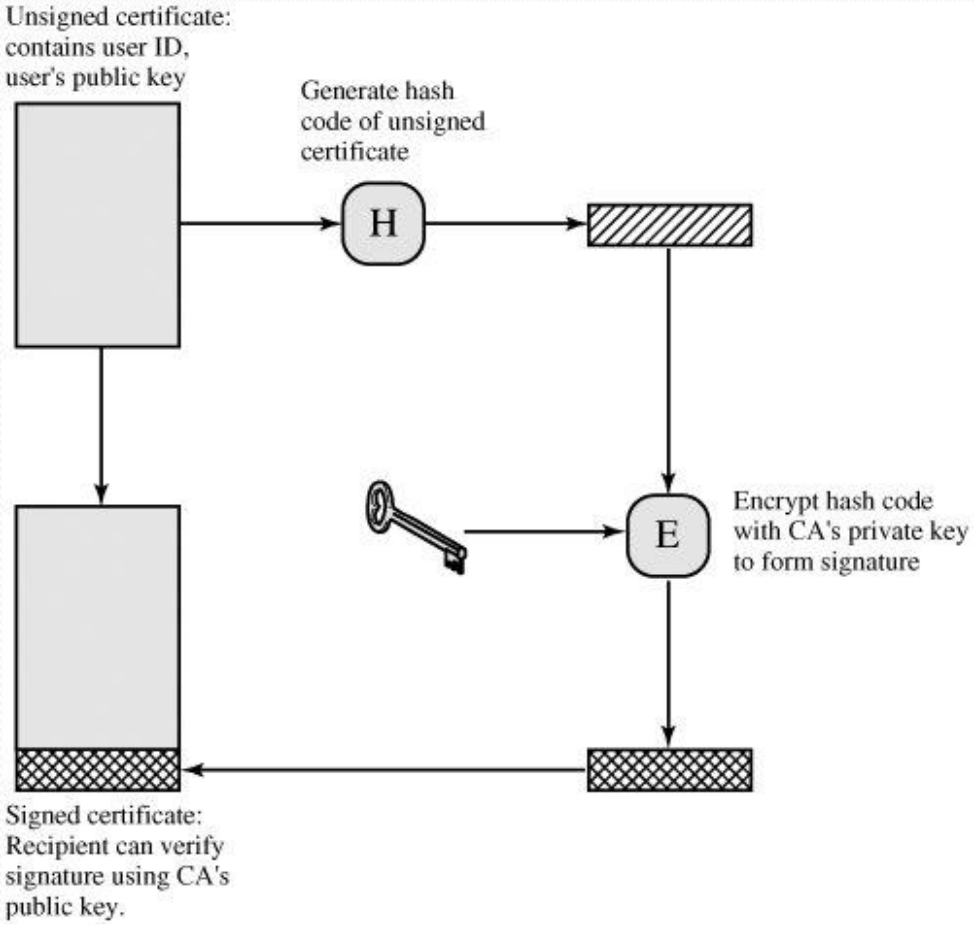
Email Security

Yasser F. O. Mohammad

REMINDER 1: What is X.509?

- Part of X.500 standard for directory services
- Recommended by ITU-T in 1988
- Used in many applications
  - SSL/TLS
  - S/MIME
  - IP Security
  - SET
  - etc
- No specific public key algorithm but usually RSA

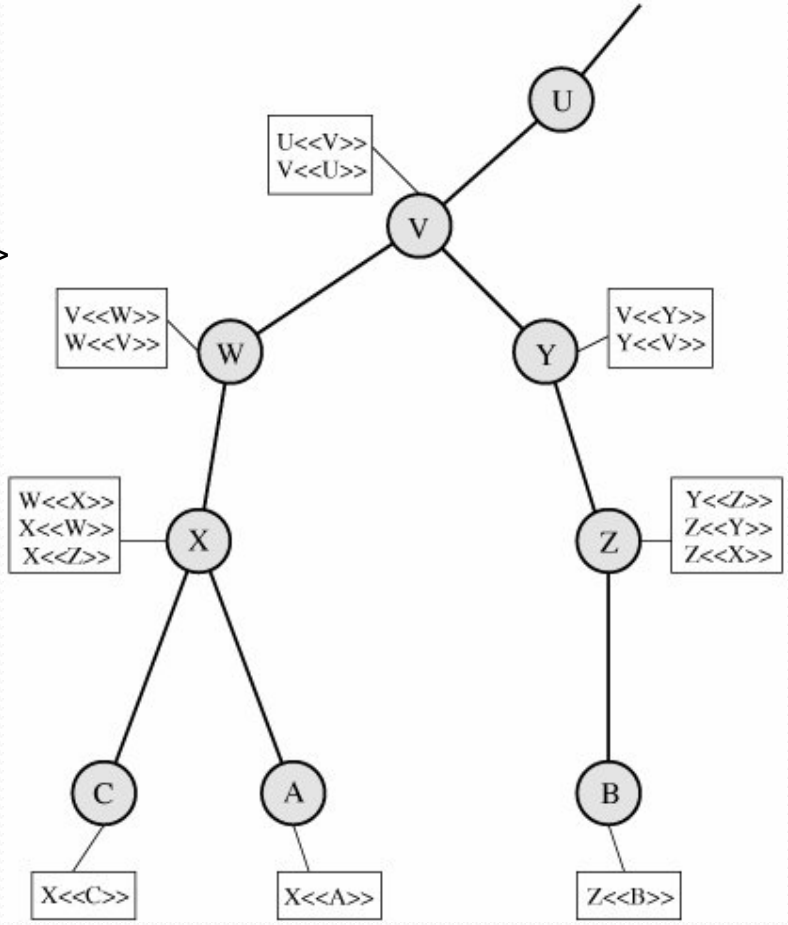
REMINDER 2: How certificates are signed?



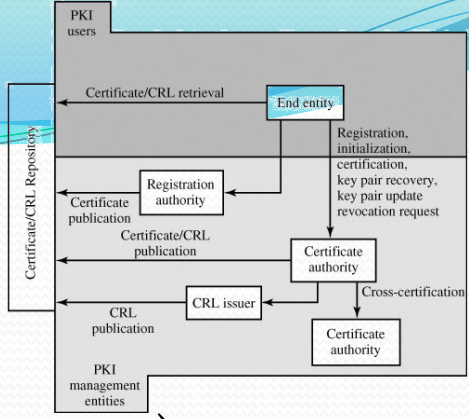
REMINDER 3: Example CA hierarchy

B can get the public key of A using the chain:

Z<<Y>> Y <<V>> V <<W>> W <<X>>X <<A>>



REMINDER 4: PKIX



- **End entity:** End users, devices (e.g., servers, routers), etc
- **Certification authority (CA):** The issuer of certificates and (usually) certificate revocation lists (CRLs).
- **Registration authority (RA):** The RA is often associated with the End Entity registration process.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** Any method for storing certificates and CRLs.

Email Security

- Current email protocol is not secure
- Any one can read the message
  - During transmission
  - In destination (with appropriate privilage)

How can Email be enhanced

- confidentiality
  - protection from disclosure
- authentication
  - of sender of message
- message integrity
  - protection from modification
- non-repudiation of origin
  - protection from denial by sender

What will we talk about

- PGP
  - Pretty Good Privacy
- S/MIME
  - International Standard

PGP

- widely used

- developed by Phil Zimmermann
  - Selected best cryptographic building blocks
  - integrated into a single program
  - available on many platforms
  - Both free and commercial version are available

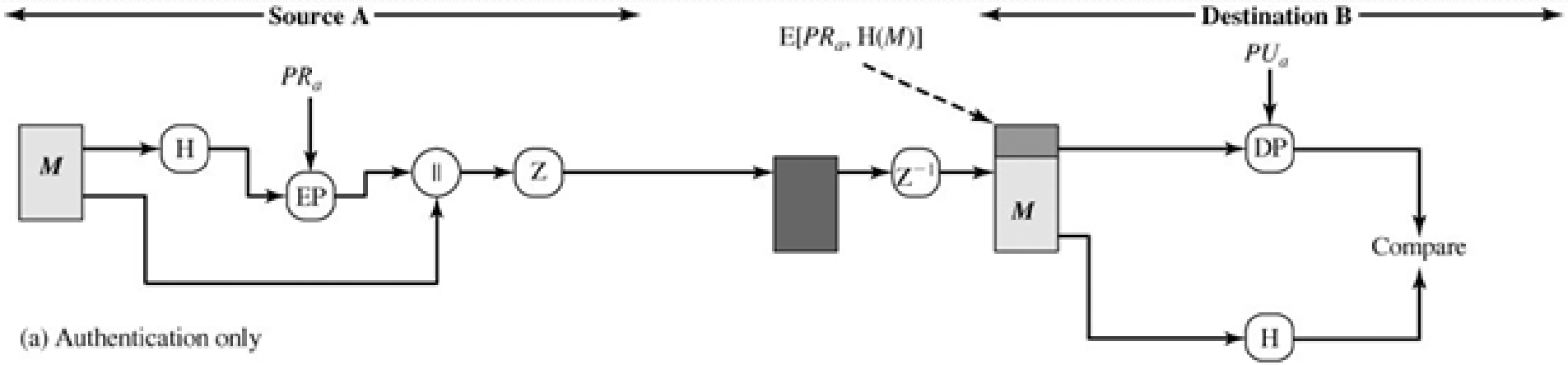
Notation

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

Building Blocks

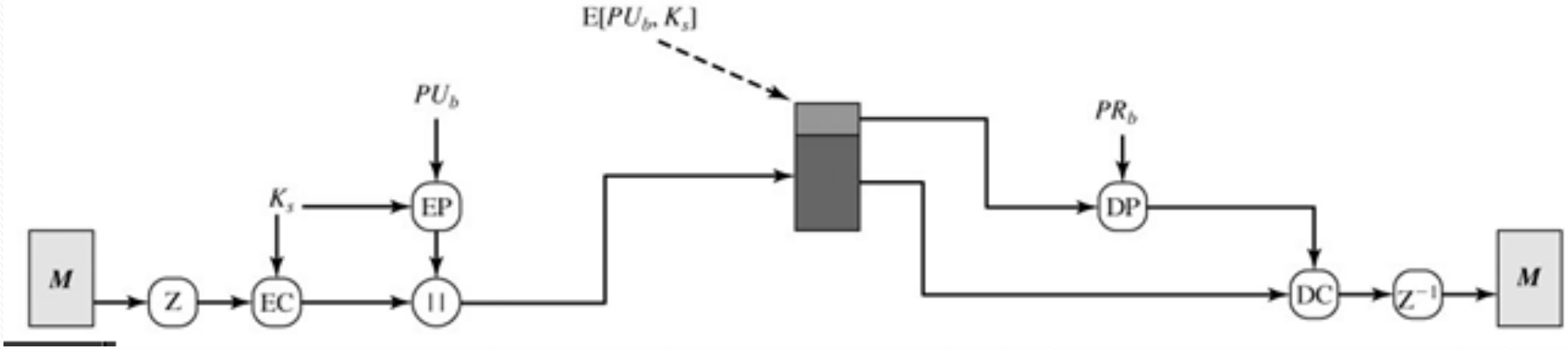
Function	Algorithms Used	Description
Digital signature	DSS/SHA or RSA/SHA	A hash code of a message is created using SHA-1. This message digest is encrypted using DSS or RSA with the sender's private key and included with the message.
Message encryption	CAST or IDEA or Three-key Triple DES with Diffie-Hellman 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.
Email compatibility	Radix 64 conversion	To provide transparency for email applications, an encrypted message may be converted to an ASCII string using radix 64 conversion.
Segmentation	—	To accommodate maximum message size limitations, PGP performs segmentation and reassembly.

Authentication



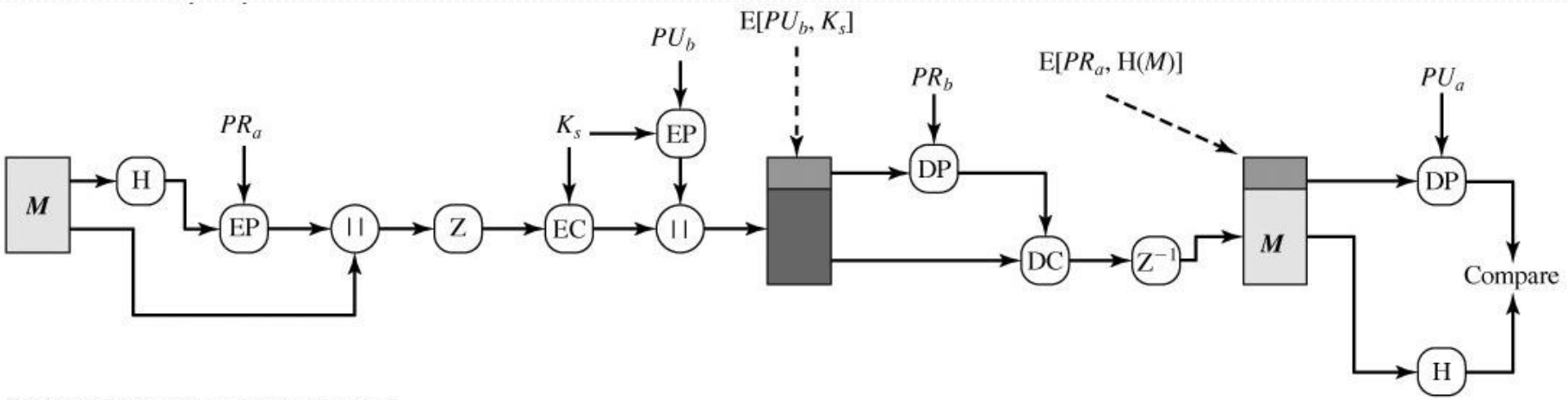
(a) Authentication only

Confidentiality



- A→B: $E(PU_B,K_{ab})||E(K_{ab},M)$
- Symmetric key
  - 128 CAST or IDEA or 3DES
- Public Key
  - RSA or ElGamal
- *No shared key distribution problem*

Authentication with Confidentiality



(c) Confidentiality and authentication

Compression

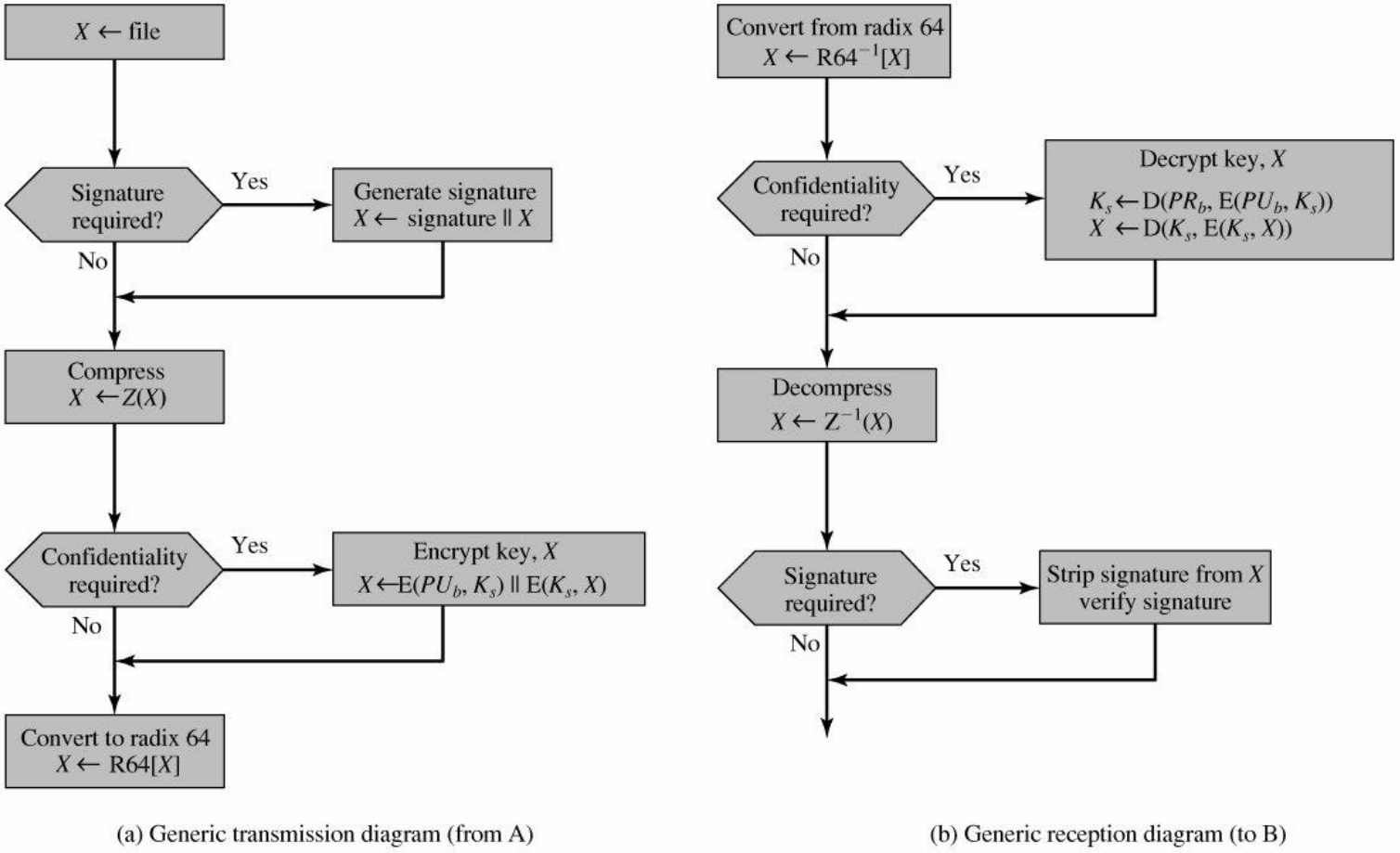
- Signature before Compression:
  - No need to store the compressed version for future verification
  - Many compression variations exist for different compression ratios.
- Encryption after compression:
  - Less redundancy in plain text

E-mail compatibility

- RADIX64 is used to convert the binary converted part to ASCII for traditional email systems
  - Each 6 bits are converted to 8

- Can be applied to whole message or encrypted parts only

Transmission and Reception



(a) Generic transmission diagram (from A)

(b) Generic reception diagram (to B)

Segmentation and Reassembly

- If message size is too large
  - Segment it after all other steps
  - Reassemble it before all other steps

Key types

- Session Keys
- Public Keys
- Private Keys
- Pass-phrases

- How to generate them?
- How to allow multiple Public/Private key pairs (updates)?
- How to store my private and others public keys?

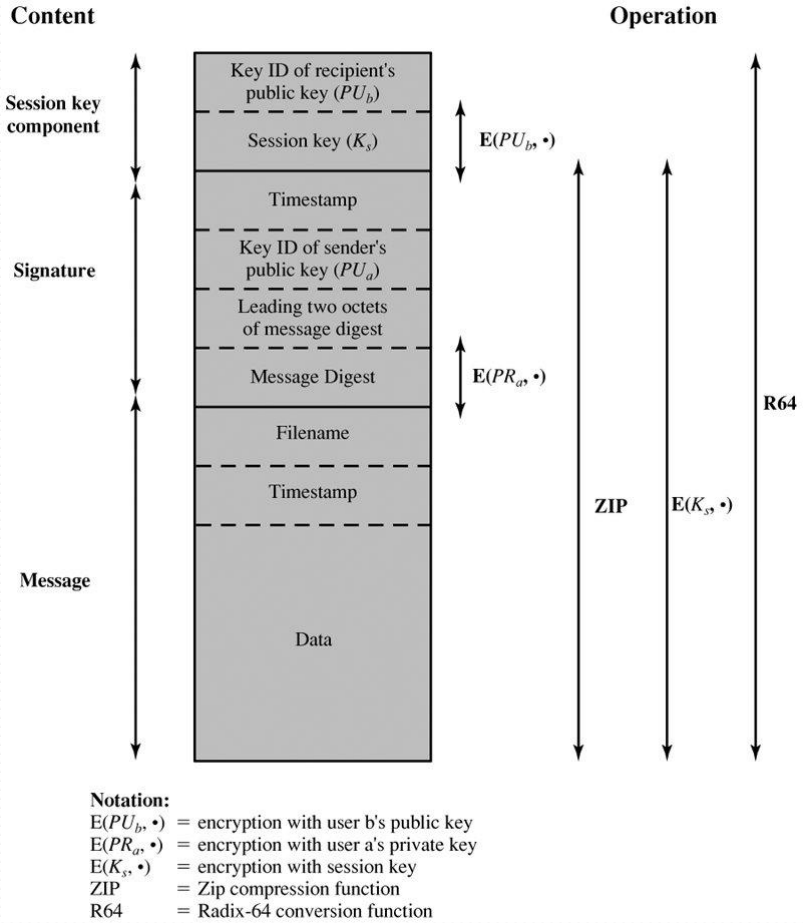
Session Key Generation

- Use timing of keyboard strokes to generate a 128 bits random number
- Apply CAST to this number as 2 input blocks using CFB mode and some fixed random key
- The output (2 blocks) is used as the session key

Multiple Public keys

- Each public key of each user has an ID
- User ID + Key ID specify the Public key pair used
- KeyID is transmitted with the message in plain
- KeyID = Least significant 64 bits of the Public Key

General PGP message



Key Ring

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

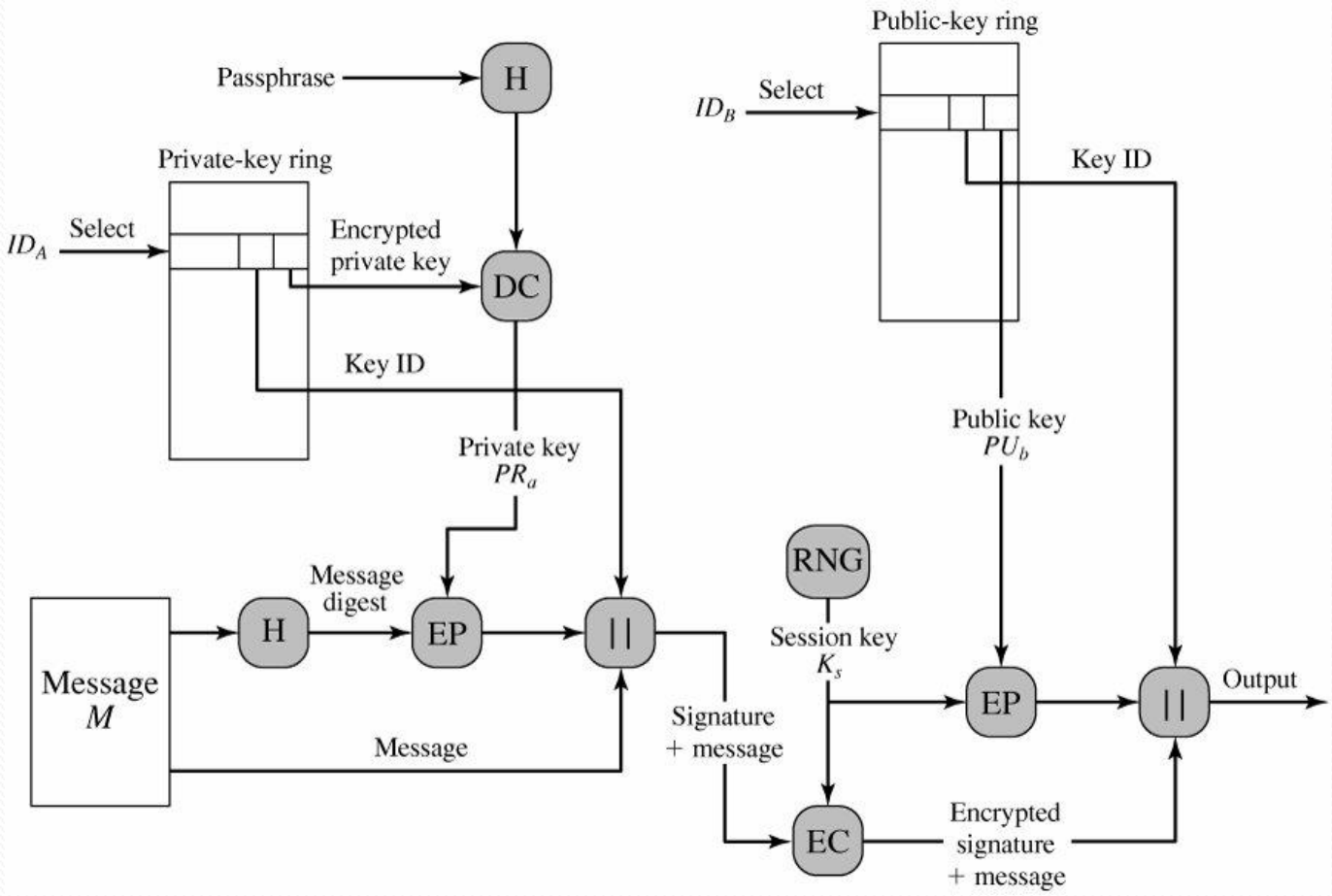
Timestamp	Key ID*	Public Key	Owner Trust	User ID*	Key Legitimacy	Signature(s)	Signature Trust(s)
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •
T_i	$PU_i \bmod 2^{64}$	PU_i	$trust\_flag_i$	User i	$trust\_flag_i$		
• • •	• • •	• • •	• • •	• • •	• • •	• • •	• • •

* = field used to index table

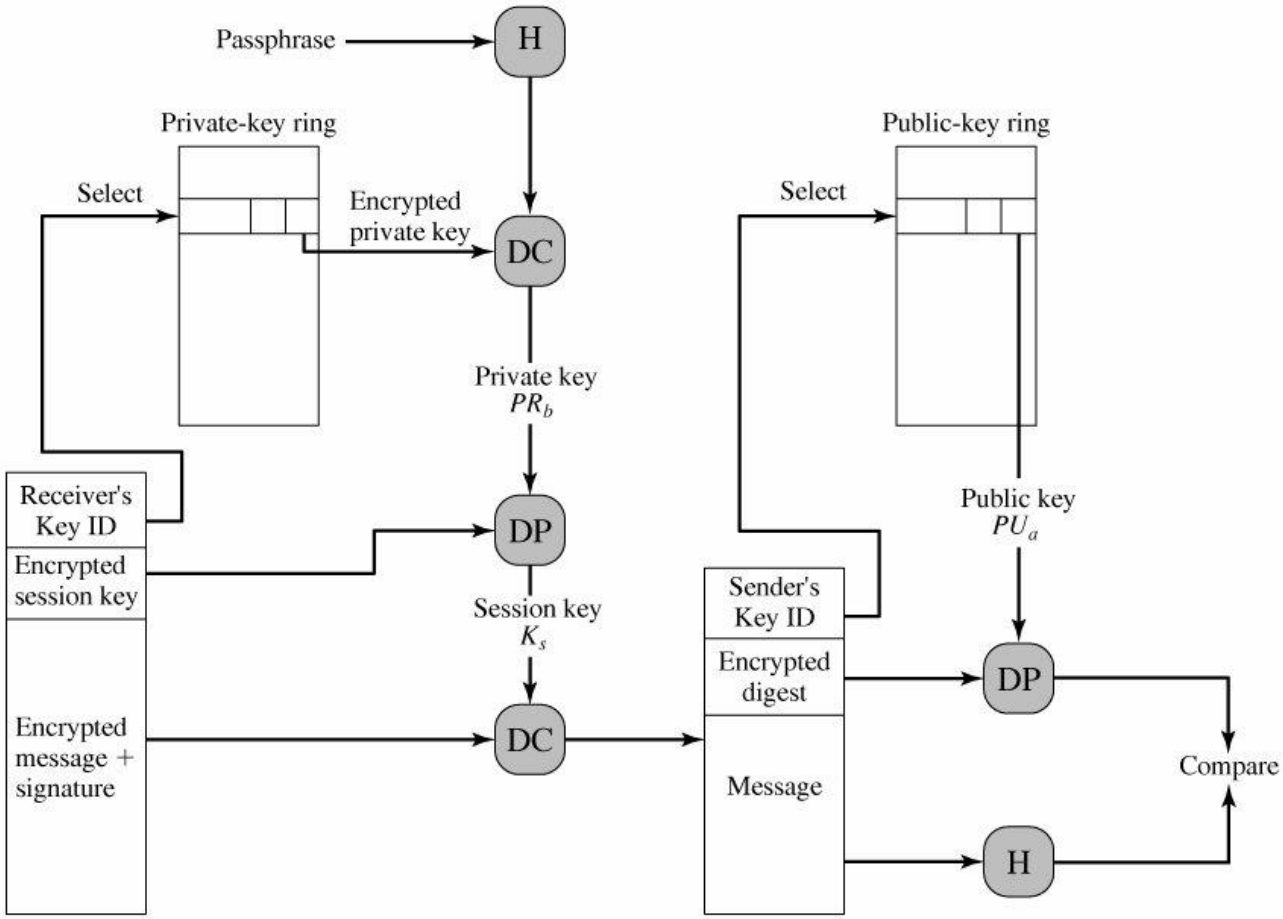
How to protect the private key

- Store E(K,PR)
- K is generated from a user defined pass-phrase

Sending a message



Receiving a message



Public Key Management

- Self Read

S/MIME

- S/MIME (Secure/Multipurpose Internet Mail Extension) is a security enhancement to the MIME Internet e-mail format standard, based on technology from RSA Data Security

- Expected to be the standard Email security scheme while PGP stays a preferred personal solution

RFC 822 (EMAIL)

- Message= Envelop + Content
  - Envelop is used for transmission
  - Content is delivered to recipient

- Envelop = Headers
  - Separated from content by an empty line!!

```
Date: Tue, 16 Jan 1998 10:37:17 (EST)
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 822
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.
```

MIME

- Multipurpose Internet Mail Extensions
  - Five new header fields
  - A number of content formats
  - Transfer encoding

New Header Fields

- MIME-Version: Must have the parameter value 1.0.
- Content-Type: Describes the data contained in the body.
- Content-Transfer-Encoding: Indicates the type of transformation that has been used.
- Content-ID: Used to identify MIME entities uniquely in multiple contexts.
- Content-Description: A text description of the object with the body.

MIME Content types

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.
	octet-stream	General binary data consisting of 8-bit bytes.

Example

- From: Nathaniel Borenstein <nsb@bellcore.com>
- To: Ned Freed <ned@innosoft.com>
- Subject: Sample message
- MIME-Version: 1.0
- Content-type: multipart/mixed; boundary="simple boundary"

  This is the preamble. It is to be ignored, though it is a handy place for mail composers to include an explanatory note to non-MIME conformant readers.
  simple boundary
  This is implicitly typed plain ASCII text. It does NOT end with a linebreak.
  simple boundary
  Content-type: text/plain; charset=us-ascii
  This is explicitly typed plain ASCII text. It DOES end with a linebreak.

  simple boundary
  This is the epilogue. It is also to be ignored.

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.

Functions of S/MIME

- Enveloped Data
  - Confidentiality
- Signed Data
  - Authentication
- Clear-signed Data
  - Authentication (RADIX64 applied to signature only for readability)
- Signed and Enveloped Data
  - Confidentiality and Authentication

S/MIME Algorithms

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 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 message.	Sending and receiving agents MUST support Diffie-Hellman. Sending agent SHOULD support RSA encryption with key sizes 512 bits to 1024 bits. Receiving agent SHOULD support RSA decryption.
Encrypt message for transmission with one-time session key.	Sending agents SHOULD support encryption with tripleDES and RC2/40. Receiving agents SHOULD support decryption using tripleDES and MUST support decryption with RC2/40.

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-signature	—	The content type of the signature subpart of a multipart/signed message.
	pkcs10-mime	—	A certificate registration request message.

EnvelopedData

1. Generate a pseudorandom session key for a particular symmetric encryption algorithm (RC2/40 or tripleDES).

2. For each recipient, encrypt the session key with the recipient's public RSA key.

3. For each recipient, prepare a block known as RecipientInfo that contains an identifier of the recipient's public-key certificate,[3] an identifier of the algorithm used to encrypt the session key, and the encrypted session key.

4. Encrypt the message content with the session key.

SignedData

- Select a message digest algorithm (SHA or MD5).
- Compute the message digest, or 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.

Types of Verisign Certificates

Class	Identity Checks	Usage
1	name/email check	web browsing/email
2+	enroll/addr check	email, subs, s/w validate
3+	ID documents	e-banking/service access