CS-446: Information Systems Security

Chapter # 16: Email Security (PGP S/MIME)

Prof. Dr. Sufian Hameed
Department of Computer Science
FAST-NUCES

Overview

- *Email Security*
  - *S/MIME*
  - *PGP*

Email Security

- Email is one of the most widely used and regarded network services
- Currently message contents are not secure
  - may be inspected either in transit
  - or by suitably privileged users on destination system

Threats to E-mail

- Loss of confidentiality.
  - E-mails are sent in clear over open networks.
  - E-mails stored on potentially insecure clients and mail servers.
- Loss of integrity.
  - No integrity protection on e-mails; anybody be altered in transit or on mail server.

Threats to E-mail

- Lack of data origin authentication.
  - Is this e-mail really from the person named in the From:field?
- Lack of non-repudiation.
  - Can I rely and act on the content? (integrity)
  - If so, can the sender later deny having sent it? Who is liable if I have acted?

Threats to E-mail

- Lack of notification of receipt.
  - Has the intended recipient received my e-mail and acted on it?
  - A message locally marked as ‘sent’ may not have been delivered.

E-mail security

- Software for encrypting email messages has been widely available for more than 20 years, but the email-using public has failed to adopt secure messaging. This failure can be explained through a combination of:
  - technical,
  - community,
  - and usability factors

E-mail Security

- **Why Don’t People Use Email Security?**
  - I don’t because I don’t care.
  - I doubt any of my usual recipients would understand
  - the significance of the signature.
  - Never had the need to send these kinds of emails.
  - I don’t think it’s necessary to encrypt my email.
  - it’s just another step & something else I don’t have time

Why do I want secure email ?

- Protect sensitive data
- Prove authenticity to recipients
- Send attachments normally filtered
- Avoid the junk folder!

How does Secure Email works ?

- Secure email uses a set cryptographic tools to encapsulate a message into a specially formatted envelope.

Encryption

- Means of hiding a message through substitution or rearranging letters
- Requires a “key” to unlock the original message

Digital Signature

- A string of characters that uniquely identifies the signer of an electronic message.
- Recipients are able to
  - Verify message was from purported sender
  - Verify message was not modified in transit
- Sender cannot deny being originator of message

Pick your poison

- Most popular secure email standards
  - S/MIME
  - PGP (OpenPGP)
- How are these different?
  - Similar services
  - Different trust models

Hierarchical Trusts

- All users directly trust some central authority (CA) and the CA issue them a Digital Certificate
- Alice trusts Bob if Bob's "chain of trust" traces back to the central authority
- Example: driver's license
  - Issued by state authority to prove identity to others

Getting a Digital Certificate

- Must be issued by an authority
  - Organizational PKI
  - Third-party vendor
- Free personal certificates available
  - VeriSign
  - GeoTrust
  - Startcom
  - CACert
  - Comodo

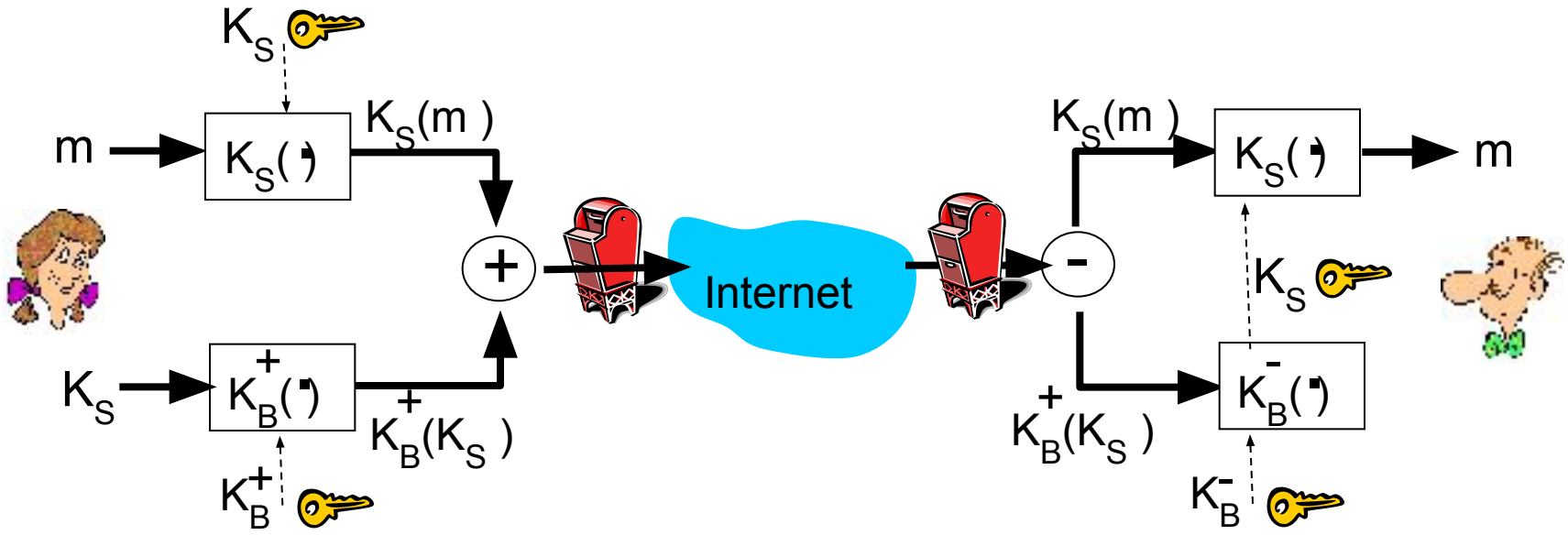
http://www.sslshopper.com/email-certificates-smime-certificates.html

Web of Trust

- Incorporates user perception of trust
- Any user can be an authority to verify others
- Users can assign levels of trust
  - Not all authorities are equal
- “Alice and Bob think she is Carol, and that’s good enough for me.”

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

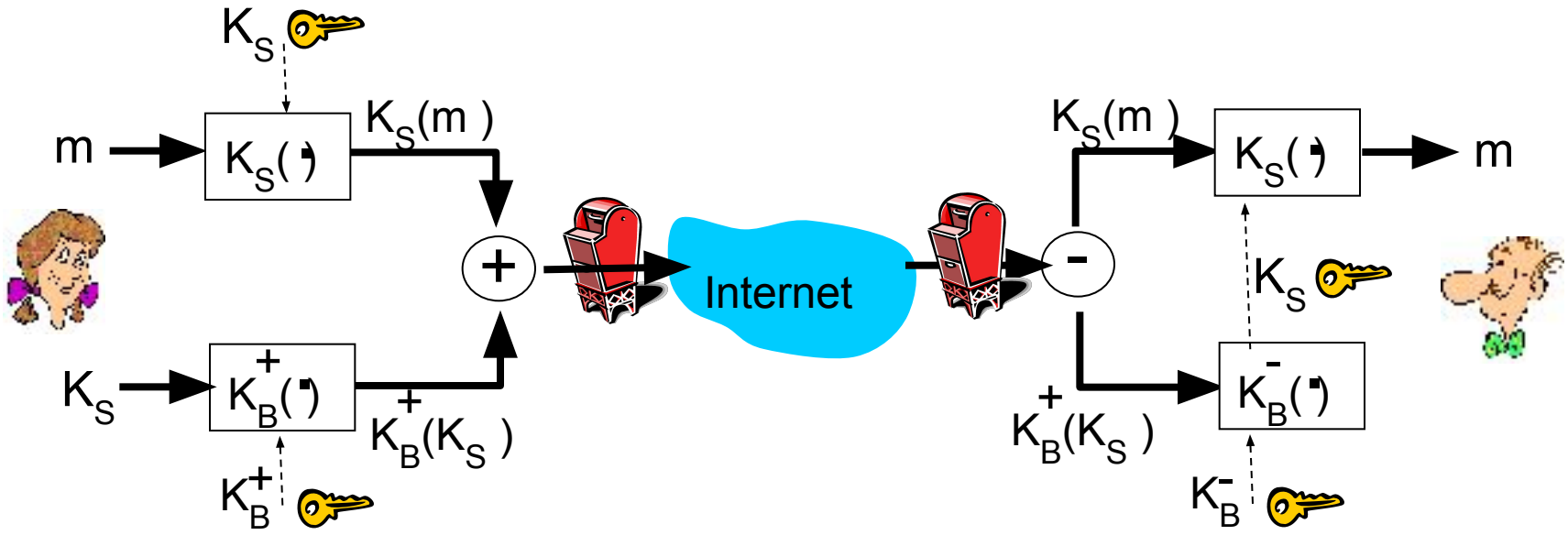


Alice:

- generates random *symmetric* private key, K_S
- encrypts message with K_S (for efficiency)
- also encrypts K_S with Bob's public key
- sends both $K_S(m)$ and $K_B(K_S)$ to Bob

Secure e-mail

- Alice wants to send confidential e-mail, m, to Bob.

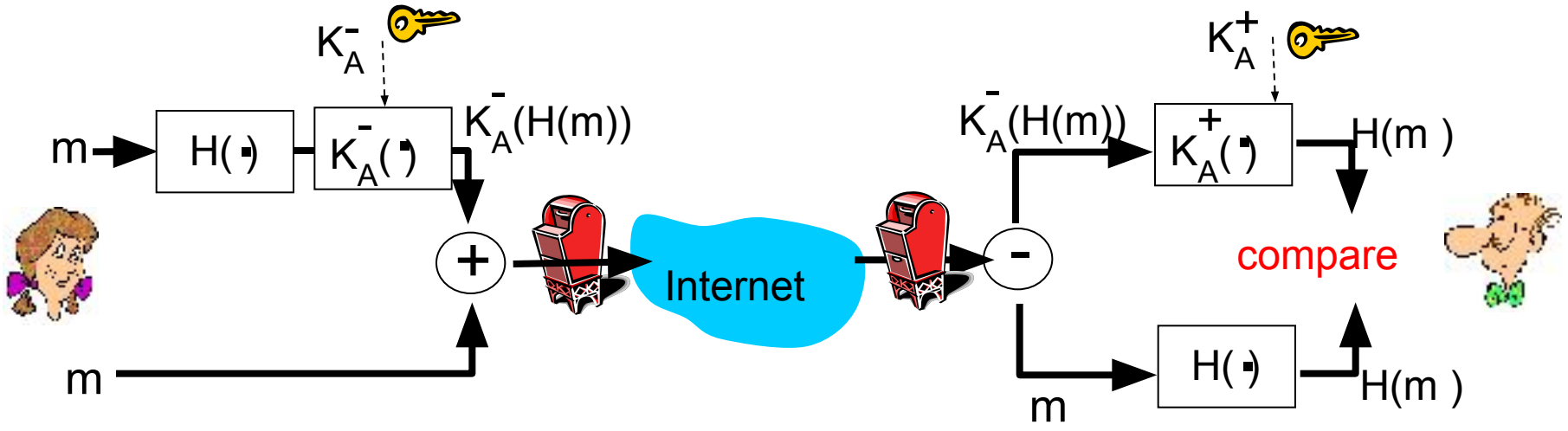


Bob:

- uses his private key to decrypt and recover K_S
- uses K_S to decrypt $K_S(m)$ to recover m

Secure e-mail (continued)

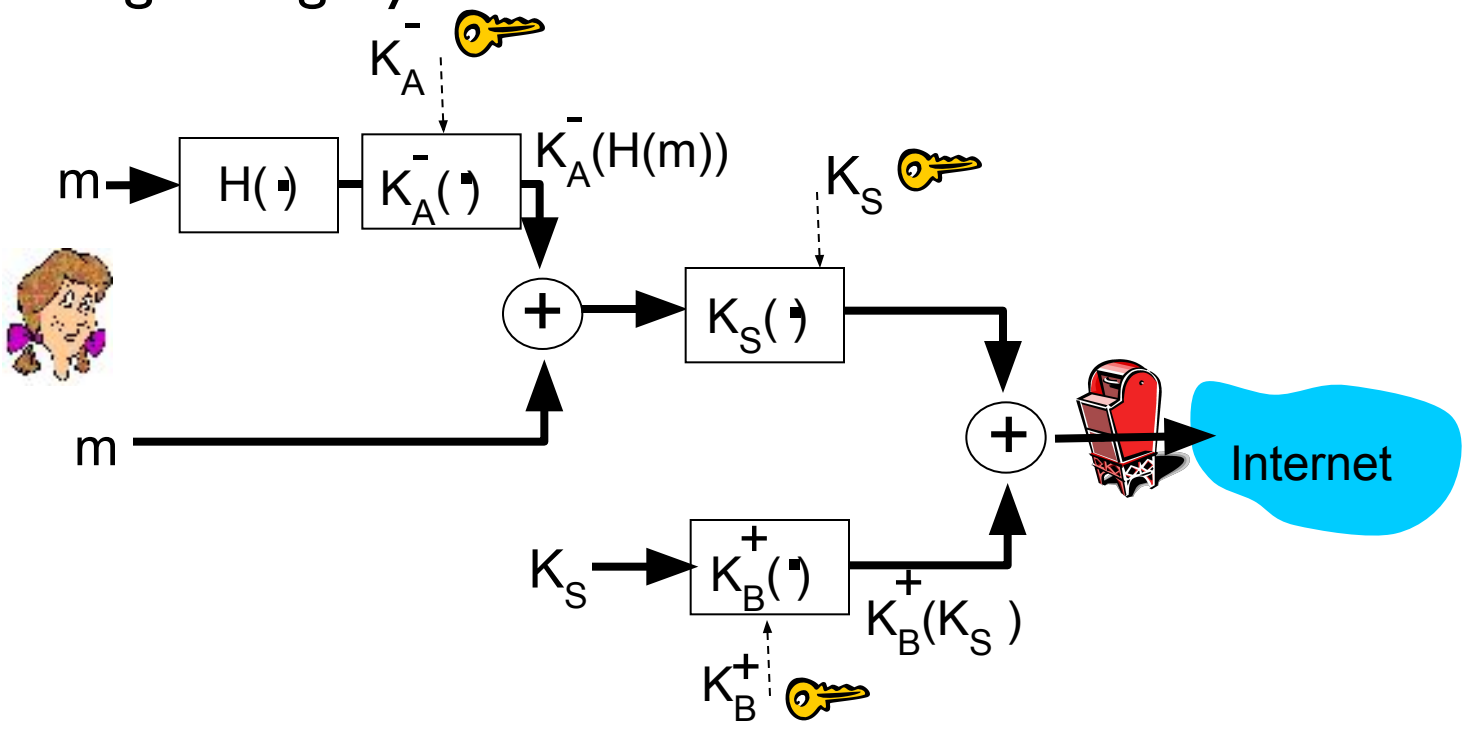
- Alice wants to provide sender authentication message integrity



- Alice digitally signs message
- sends both message (in the clear) and digital signature

Secure e-mail (continued)

- Alice wants to provide secrecy, sender authentication, message integrity.



Alice uses three keys: her private key, Bob's public key, newly created symmetric key

S/MIME (Secure/Multipurpose Internet Mail Extension)

- Originated from RSA Data Security Inc. in 1995.
- Further development by IETF S/MIME working group at: www.ietf.org/html.charters/smime-charter.html.
- Version 3.2 specified in RFC 5751.
- Allows flexible client-client security through encryption and signatures.
- Widely supported, e.g. in Microsoft Outlook, Thunderbird, Lotus Notes.

Understanding What S/MIME Does

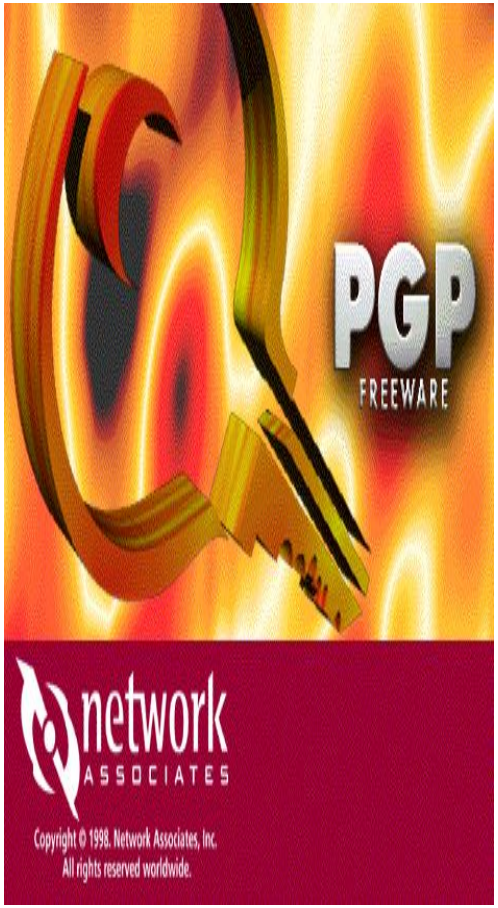
- S/MIME provides two security services:
  - Digital signatures
  - Message encryption

http://technet.microsoft.com/en-us/library/aa995740(v=exchg.65).aspx

Key Wrapping and Content Encryption in S/MIME

- RSA is used as Key wrapping algorithm
- Content encryption is based on session keys.
  - AES-128 CBC added as MUST, AES-192 and AES-256 CBC are optional

PGP (Pretty Good Privacy)



Pretty Good Privacy (**PGP**) is an encryption program that provides cryptographic privacy and authentication for data communication. PGP is used for signing, encrypting, and decrypting texts, e-mails, files, directories, and whole disk partitions and to increase the security of e-mail communications. Phil Zimmermann developed PGP in 1991.[3]

- PGP and similar software follow the OpenPGP, an open standard

PGP (Pretty Good Privacy)

- Open source, freely available software package for secure e-mail
- De facto standard for secure email
- Developed by Phil Zimmermann
- Selected best available crypto algos to use
- Runs on a variety of platforms like Unix, PC, Macintosh and other systems
- Originally free (now also have commercial versions available)

PGP (Pretty Good Privacy)

- **PGP Algorithms**
  - **Symmetric encryption:**
    - DES, 3DES, AES and others.
  - **Public key encryption of session keys:**
    - RSA or ElGamal.
  - **Hashing:**
    - SHA-1, MD-5 and others.
  - **Signature:**
    - RSA, DSS, ECDSA and others.

Key Management and Web of Trust

PGP use:

- **public keys** for encrypting session keys / verifying signatures.
- **private keys** for decrypting session keys / creating signatures.

- PGP adopts a trust model called the *web of trust*.
- No centralised authority
- Individuals sign one another's public keys, these "certificates" are stored along with keys in key rings.

In cryptography, a web of trust is a concept used in PGP, GnuPG, and other OpenPGP-compatible systems to establish the authenticity of the binding between a public key and its owner.

Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such).

As with computer networks, there are many independent webs of trust, and any user (through their identity certificate) can be a part of, and a link between, multiple webs.

The web of trust concept was first put forth by PGP creator Phil Zimmermann in 1992 in the manual for PGP version 2.0:

As time goes on, you will accumulate keys from other people that you may want to designate as trusted introducers. Everyone else will each choose their own trusted introducers.

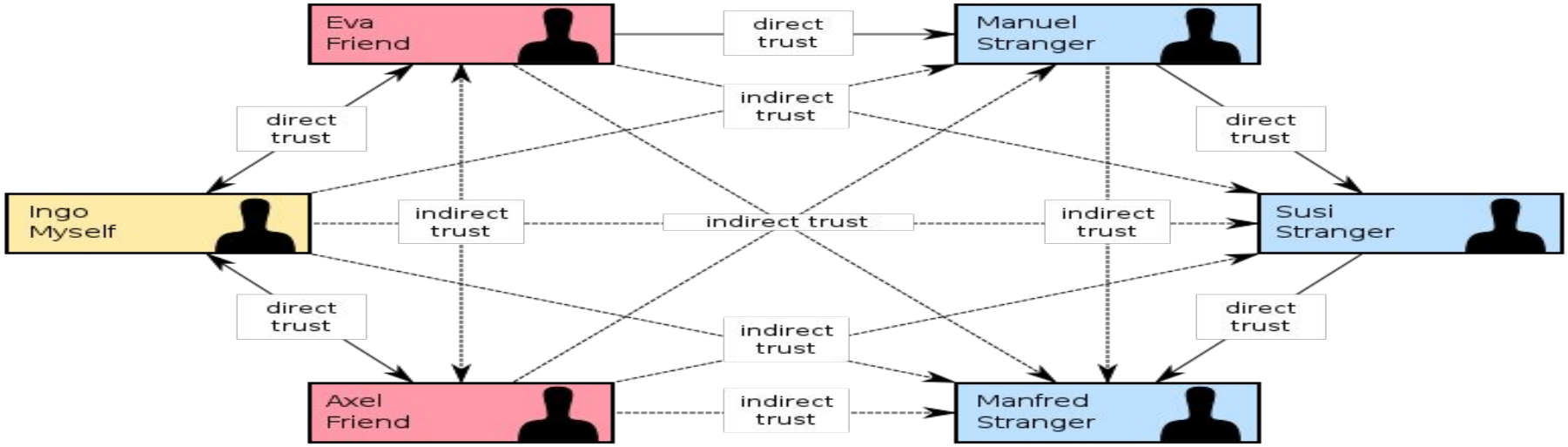
And everyone will gradually accumulate and distribute with their key a collection of certifying signatures from other people, with the expectation that anyone receiving it will trust at least one or two of the signatures.

Simplified explanation of Web of Trust:

There are two keys pertaining to a person: a public key which is shared openly and a private key that is withheld by the owner.

The owner's private key will decrypt any information encrypted with its public key. In the web of trust, each user has a ring with a group of people's public keys.

Users encrypt their information with the recipient's public key, and only the recipient's private key will decrypt it. Each user then digitally signs the information with their private key, so when the recipient verifies it against the users own public key, they can confirm that it is the user in question. Doing this will ensure that the information came from the specific user and has not been tampered with, and only the intended recipient can read the information (because only they know their private key).



Trust Level for Public Key

- PGP computes a *trust level* for each public key in key ring.
- Users interpret trust level for themselves.
- Trust levels for public keys dependent on:
  - Number of signatures on the key;
  - Trust level assigned to each of those signatures.
- Trust levels recomputed from time to time.

PGP Key Rings

- PGP Key Rings
  - PGP supports multiple public/private keys pairs per sender/recipient.
  - Keys stored locally in a *PGP Key Ring* – essentially a database of keys.
  - Private keys stored in encrypted form; decryption key determined by user-entered pass-phrase.

PGP Session Keys

- Need a session key for each message
  - Varying sizes: 56-bit DES, 128-bit CAST or IDEA, 168-bit Triple-DES, 128, 192, 256 bits AES
- Uses random inputs taken from

  -actual keys hit

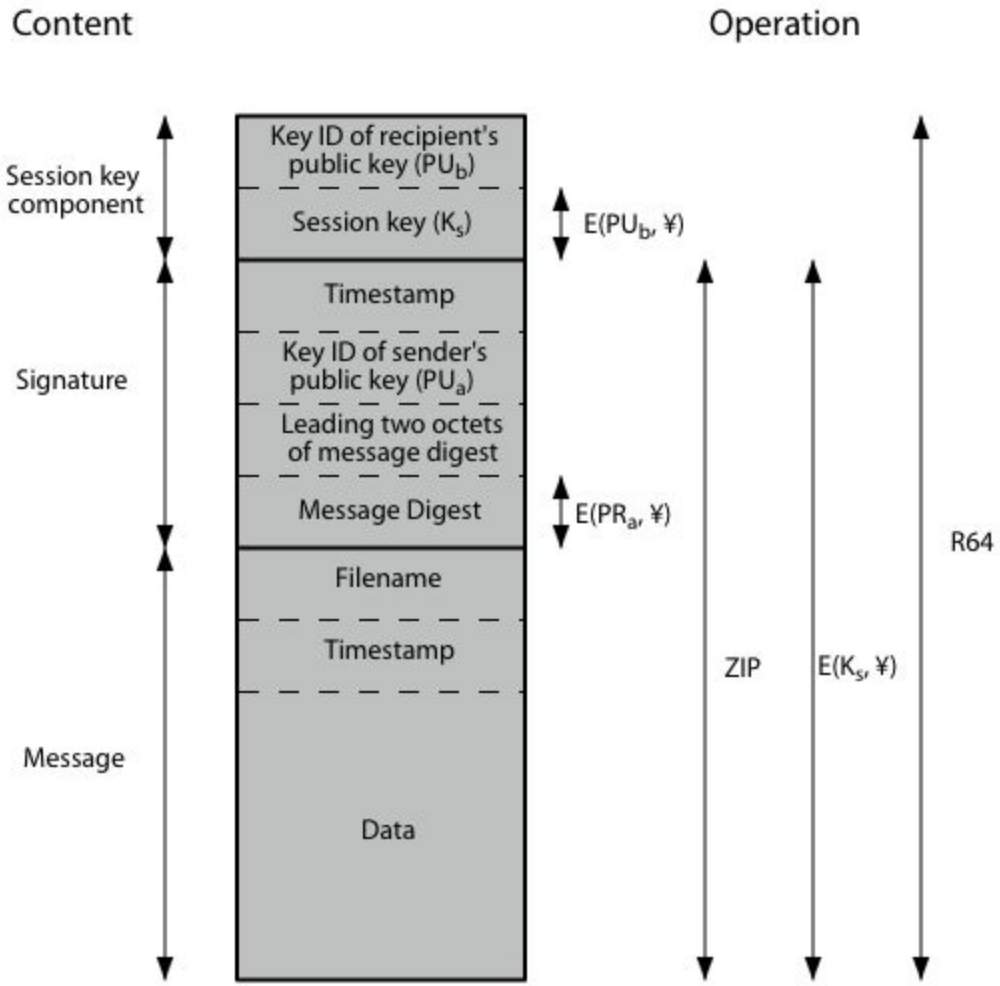
  -keystroke timing of a user

PGP makes use of four types of keys: one-time session symmetric keys, public keys, private keys, and passphrase-based symmetric keys.
Each session key is associated with a single message and is used only for the purpose of encrypting and decrypting that message. Random numbers are generated using the ANSI X12.17 generator, with inputs based on keystroke input from the user, where both the keystroke timing and the actual keys struck are used to generate a randomized stream of numbers. Stallings Appendix 15C discusses PGP random number generation techniques in more detail.

PGP Public & Private Keys

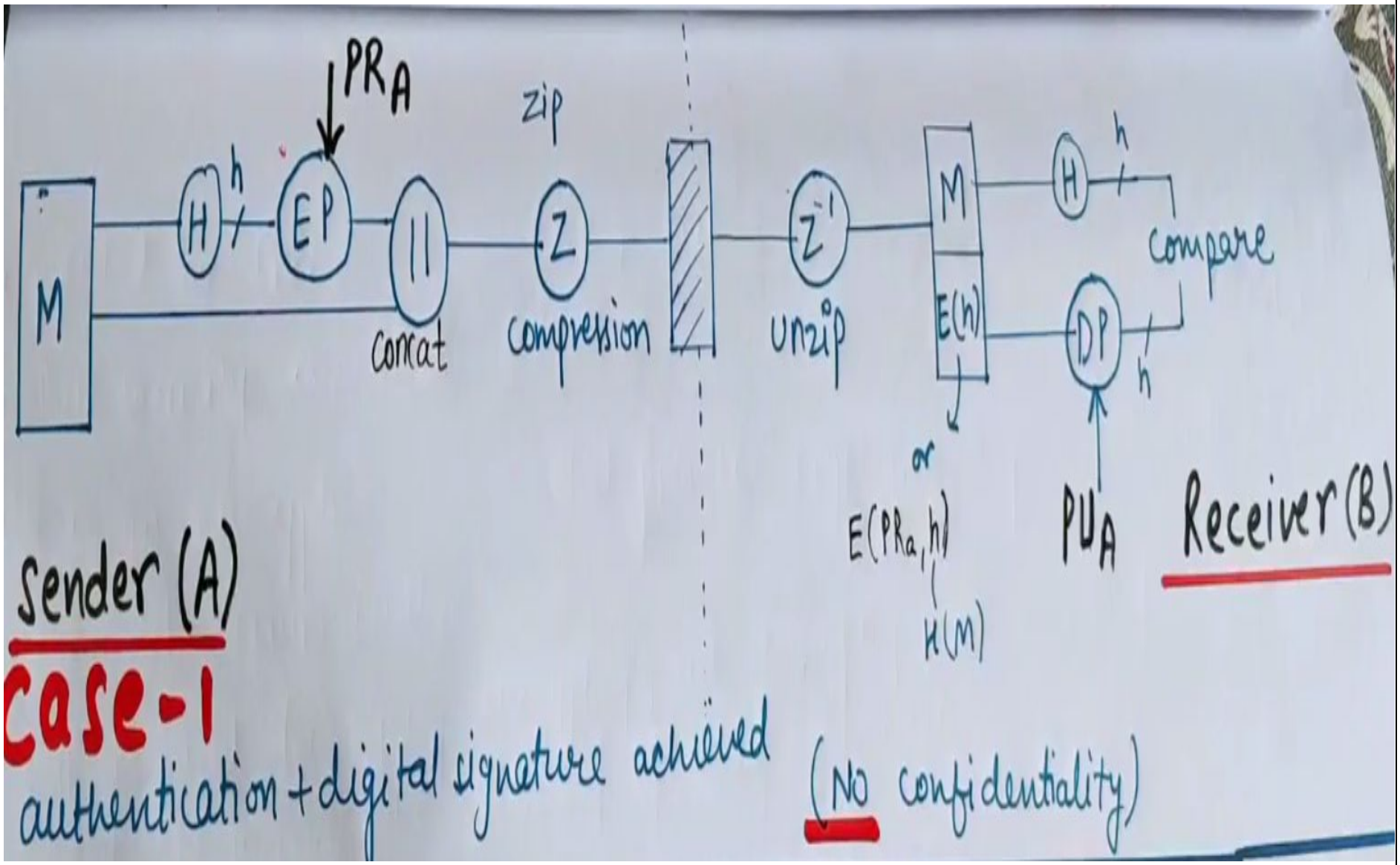
- Since many public/private keys may be in use, need to identify which is actually used to encrypt session key in a message
  - Could send full public-key with every message
  - but this is inefficient
- Rather use a key identifier based on key
  - is least significant 64-bits of the key
  - will very likely be unique
- Also use key ID in signatures

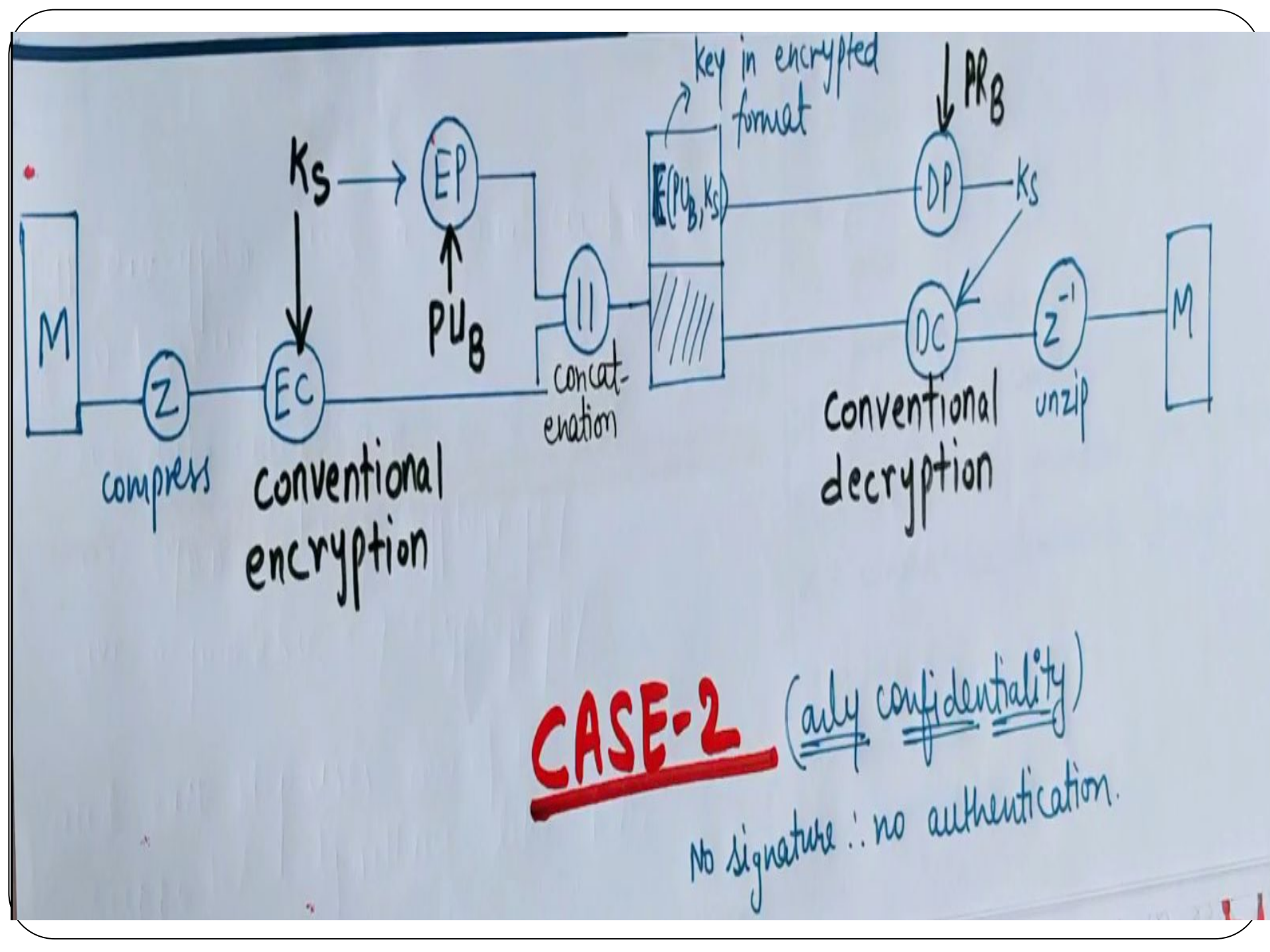
PGP Message Format

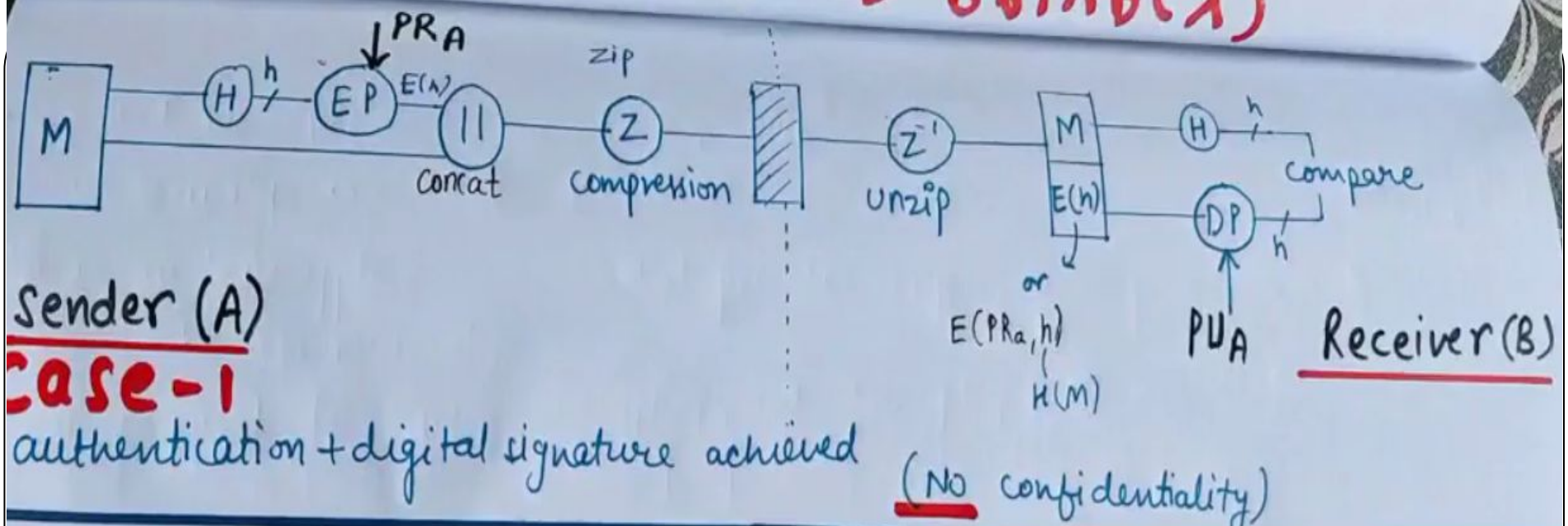


PGP services

- **Messages**
  - authentication
  - confidentiality
  - compression
  - e-mail compatibility
  - segmentation and reassembly
- **Key Management**
  - generation, distribution, and revocation of public/private keys
  - generation and transport of session keys and IVs


PRA
zip
M
H
h
EP
||
Z
Z⁻¹
M
H
h
compare
E(h)
DP
h
concat
compression
unzip
or
E(PRa, h)
H(M)
PUA
Receiver (B)
Sender (A)
case-1
authentication + digital signature achieved
(No confidentiality)


M
Z
compress
K_S
EC
conventional encryption
EP
PU_B
||
concat-enation
key in encrypted format
$E(PU_B, K_S)$
PR_B
DP
K_S
DC
conventional decryption
Z^{-1}
unzip
M
CASE-2 (only confidentiality)
No signature ∴ no authentication.



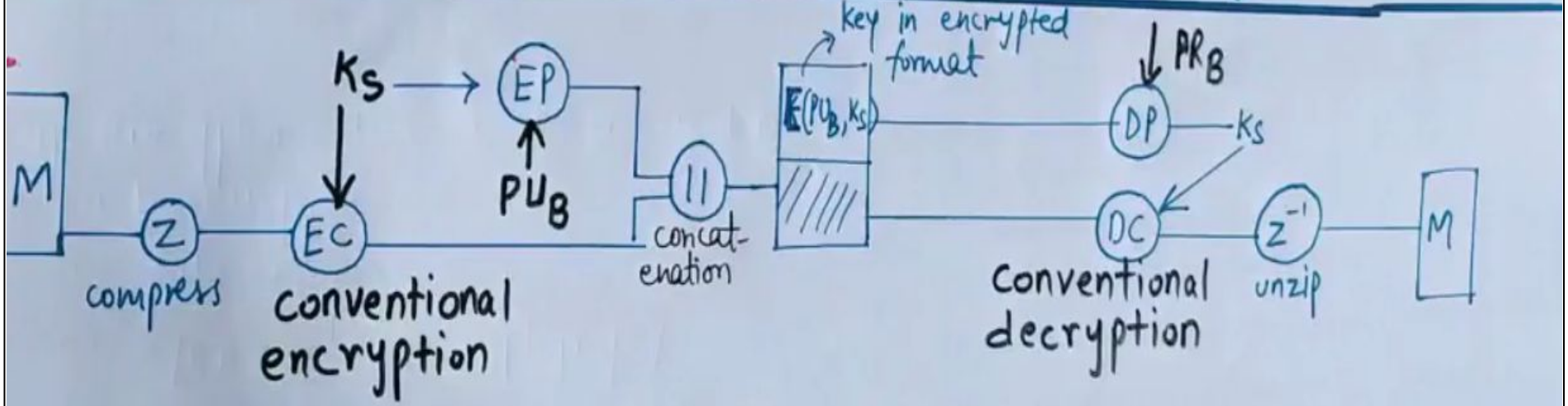
Sender (A)

Receiver (B)

Case-1

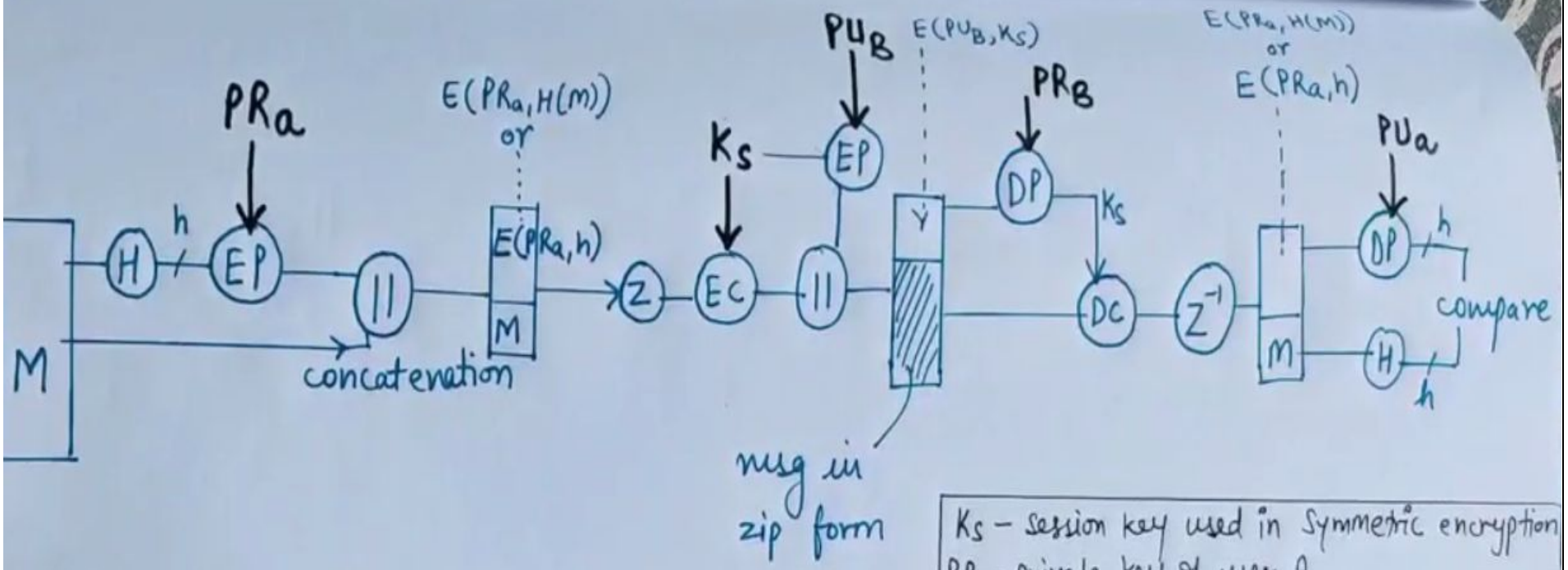
authentication + digital signature achieved

(No confidentiality)



CASE-2 (only confidentiality)

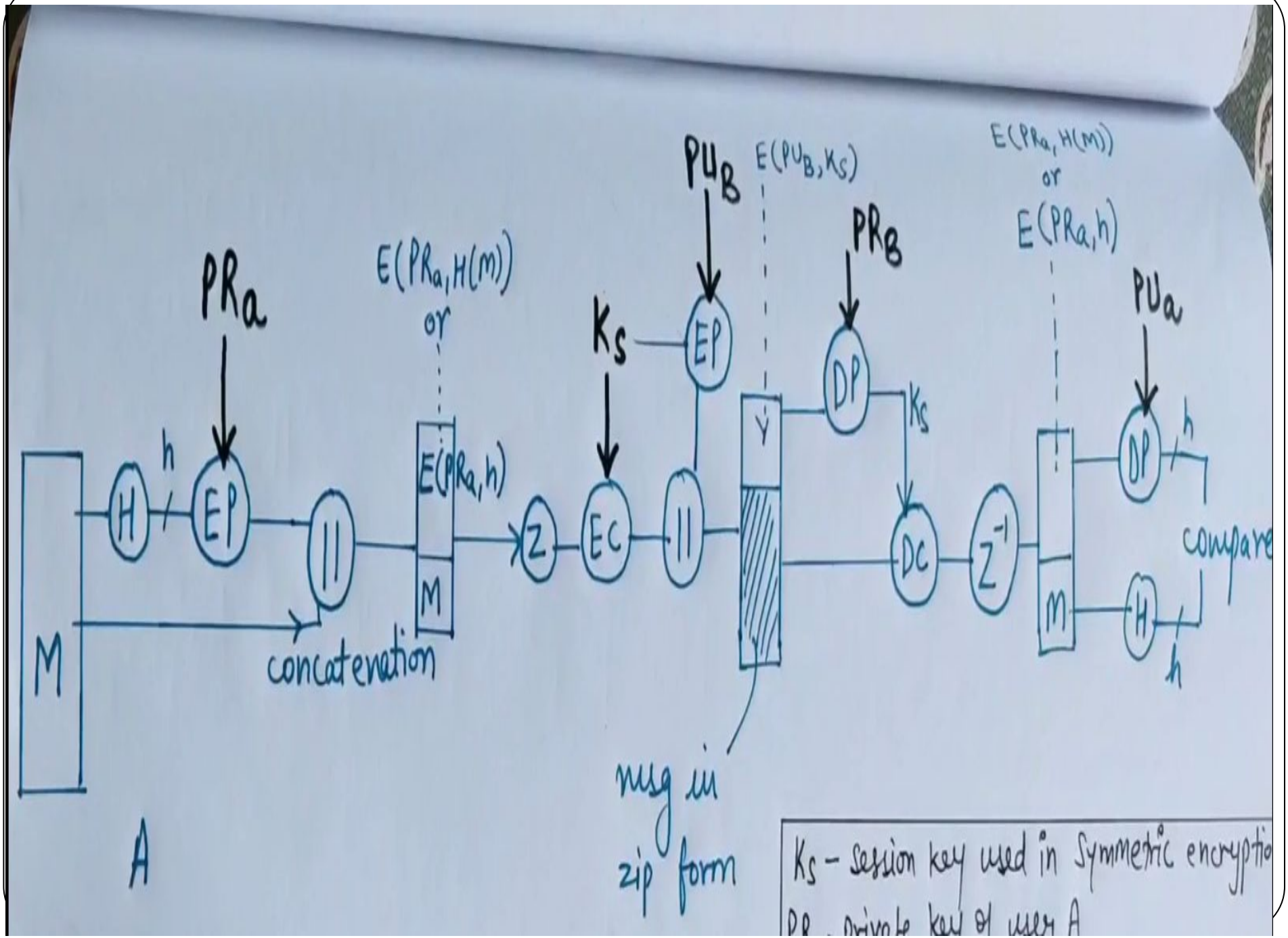
No signature ∴ no authentication.



Ks – session key used in Symmetric encryption
PRa – private key of user A
PUa – public " " " "
EP – public key encryption
DP – public key decryption
EC – conventional / symmetric encryption
DC – conventional / symmetric decryption

H → Hash fn
h – hash code
Z – compression algo

(C) CONFIDENTIALITY + AUTHENTICATION


PRa
E(PRa, H(M))
or
E(PRa, h)
Ks
PUB
E(PUB, Ks)
PRB
E(PRa, H(M))
or
E(PRa, h)
PUa
H
h
EP
||
M
concatenation
Z
EC
DP
Ks
DC
Z-1
compare
msg in
zip form
A
Ks - session key used in Symmetric encryptio
PR - private key of user A

K_s - session key used in Symmetric encryption

PR_a - private key of user A

PU_a - public " " " "

EP - public key encryption

DP - public key decryption

EC - conventional / symmetric encryption

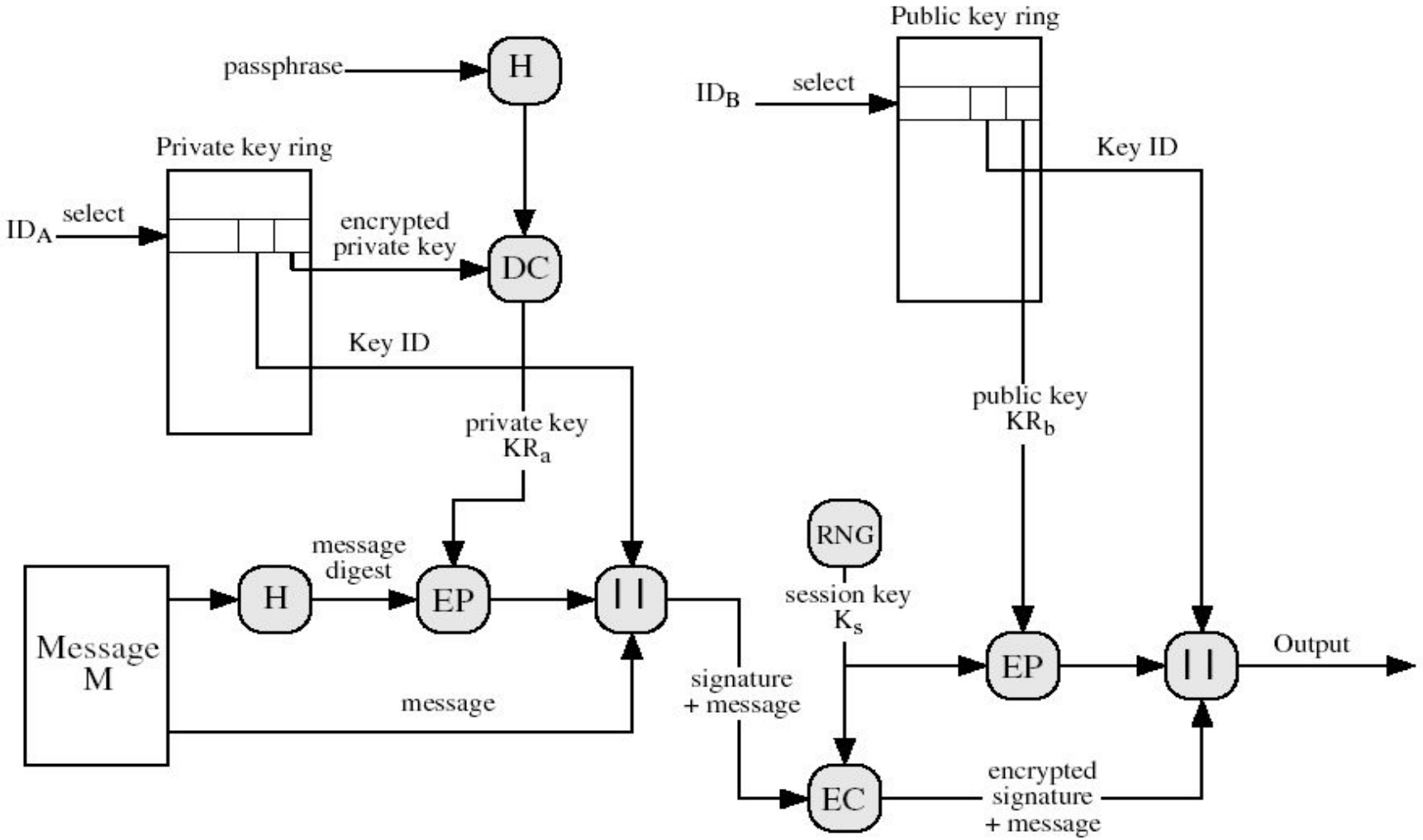
DC - conventional / symmetric decryption

H → Hash fn

h - hash code

Z - compression algo

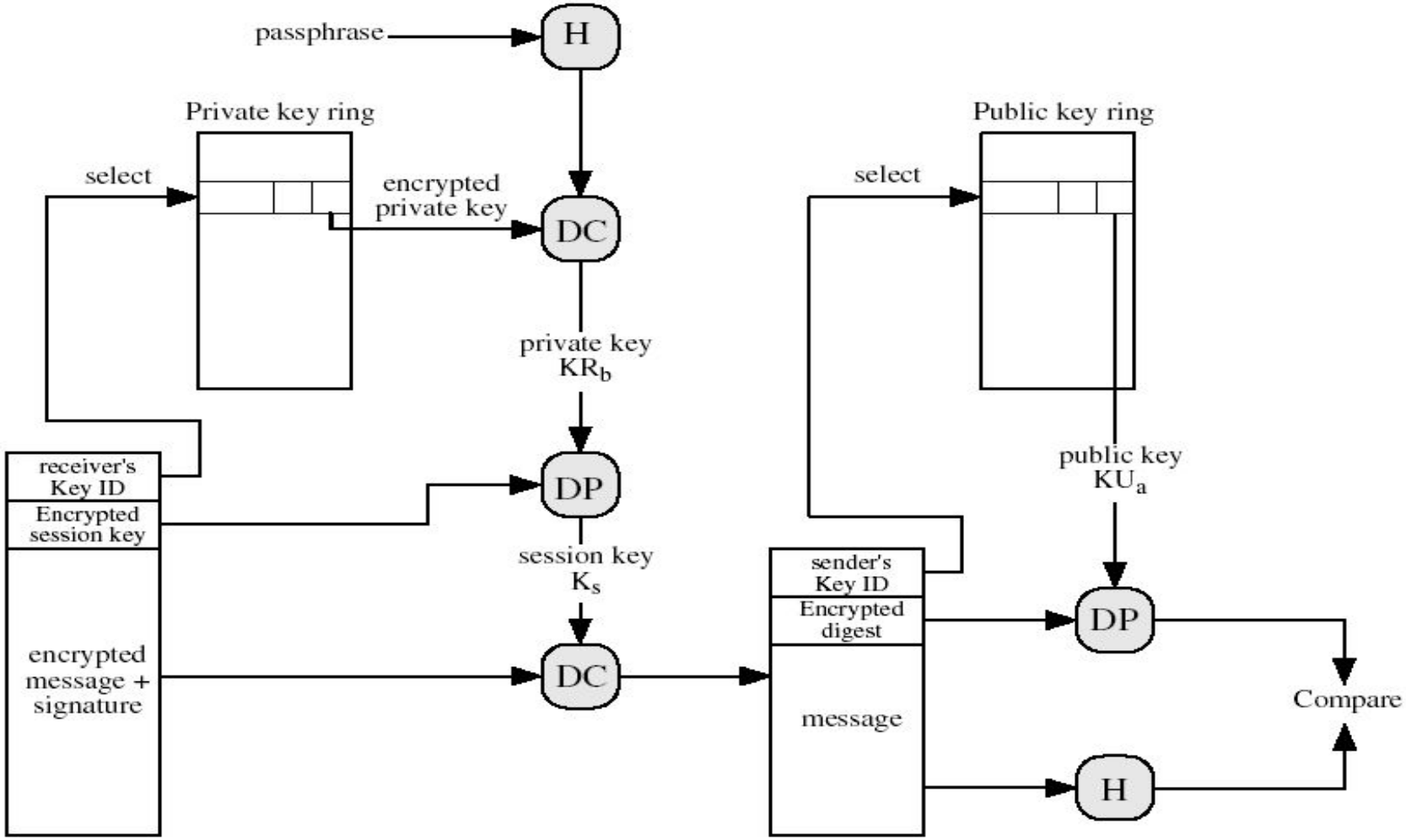
PGP Message Generation



PGP Message Generation

- The sending PGP entity performs the following steps:
  - Signs the message:
    - PGP gets sender's private key from key ring using its user id as an index.
    - PGP prompts user for passphrase to decrypt private key.
    - PGP constructs the signature component of the message.
  - Encrypts the message:
    - PGP generates a session key and encrypts the message.
    - PGP retrieves the receiver public key from the key ring using its user id as an index.
    - PGP constructs session component of message

PGP Message Reception



PGP Message Reception

- The receiving PGP entity performs the following steps:
  - Decrypting the message:
    - PGP get private key from private-key ring using Key ID field in session key component of message as an index.
    - PGP prompts user for passphrase to decrypt private key.
    - PGP recovers the session key and decrypts the message.
  - Authenticating the message:
    - PGP retrieves the sender's public key from the public-key ring using the Key ID field in the signature key component as index.
    - PGP recovers the transmitted message digest.
    - PGP computes the message for the received message and compares it to the transmitted version for authentication.

Resources

- http://www.pgpi.org/doc/faq/
- www.gnupg.org
- William Stallings," **Cryptography and Network Security Principles and Practices",** Fourth Edition " Prentice Hall , 2005
- GITA" **Encryption Technologies**", Standard P800-S850 V2.0, April 5, 2004
- Sieuwert van Otterloo" **A security analysis of Pretty Good Privacy**", September 7, 2001
- Amr el-kadi" what is computer security"2005

Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Hussain Awad and Dr. Lo'ai Tawalbeh
- Lawrie Brown (University of Kentucky)
- M. Singhal (University of Kentucky)
- Addam Schroll (Purdue University)