MCA : MCA23203

CRYPTOGRAPHY
&
NETWORK SECURITY

UNIT - 4

Unit-4 : Integrity checks and Authentication Algorithms

Secure Hash Algorithm (SHA) Digital signatures: Digital signatures- authentication protocol – digital signature standards (DSS)- Authentication Applications: X.509 – directory authentication service – electronic mail security – pretty good privacy (PGP)- S/MIME

Digital Signatures

- have looked at message authentication
  - but does not address issues of lack of trust
- digital signatures provide the ability to:
  - verify author, date & time of signature
  - authenticate message contents
  - be verified by third parties to resolve disputes
- hence include authentication function with additional capabilities

Digital Signature Properties

- must depend on the message signed
- must use information unique to sender
  - to prevent both forgery and denial
- must be relatively easy to produce
- must be relatively easy to recognize & verify
- be computationally infeasible to forge
  - with new message for existing digital signature
  - with fraudulent digital signature for given message
- be practical save digital signature in storage

Direct Digital Signatures

- involve only sender & receiver
- assumed receiver has sender’s public-key
- digital signature made by sender signing entire message or hash with private-key
- can encrypt using receivers public-key
- important that sign first then encrypt message & signature
- security depends on sender’s private-key

Arbitrated Digital Signatures

- involves use of arbiter A
  - validates any signed message
  - then dated and sent to recipient
- requires suitable level of trust in arbiter
- can be implemented with either private or public-key algorithms
- arbiter may or may not see message

Authentication Protocols

- used to convince parties of each others identity and to exchange session keys
- may be one-way or mutual
- key issues are
  - confidentiality – to protect session keys
  - timeliness – to prevent replay attacks

Replay Attacks

- where a valid signed message is copied and later resent
  - simple replay
  - repetition that can be logged
  - repetition that cannot be detected
  - backward replay without modification
- countermeasures include
  - use of sequence numbers (generally impractical)
  - timestamps (needs synchronized clocks)
  - challenge/response (using unique nonce)

Using Symmetric Encryption

- as discussed previously can use a two-level hierarchy of keys
- usually with a trusted Key Distribution Center (KDC)
  - each party shares own master key with KDC
  - KDC generates session keys used for connections between parties
  - master keys used to distribute these to them

Needham-Schroeder Protocol

- original third-party key distribution protocol
- for session between A B mediated by KDC
- protocol overview is:

1. A→KDC: $ID_A \| ID_B \| N_1$

2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]]$

3. A→B: $E_{Kb}[Ks \| ID_A]$

4. B→A: $E_{Ks}[N_2]$

5. A→B: $E_{Ks}[f(N_2)]$

Needham-Schroeder Protocol

- used to securely distribute a new session key for communications between A & B
- but is vulnerable to a replay attack if an old session key has been compromised
  - then message 3 can be resent convincing B that is communicating with A
- modifications to address this require:
  - timestamps (Denning 81)
  - using an extra nonce (Neuman 93)

Using Public-Key Encryption

- have a range of approaches based on the use of public-key encryption
- need to ensure have correct public keys for other parties
- using a central Authentication Server (AS)
- various protocols exist using timestamps or nonces

Denning AS Protocol

- Denning 81 presented the following:
  1. $A \rightarrow AS: ID_A \,||\, ID_B$
  2. $AS \rightarrow A: E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T]$
  3. $A \rightarrow B: E_{KRas}[ID_A||KU_a||T] \,||\, E_{KRas}[ID_B||KU_b||T] \,||\, E_{KUb}[E_{KRas}[K_s||T]]$
- note session key is chosen by A, hence AS need not be trusted to protect it
- timestamps prevent replay but require synchronized clocks

One-Way Authentication

- required when sender & receiver are not in communications at same time (eg. email)
- have header in clear so can be delivered by email system
- may want contents of body protected & sender authenticated

Using Symmetric Encryption

- can refine use of KDC but can't have final exchange of nonces, vis:
  1. A→KDC: $ID_A \| ID_B \| N_1$
  2. KDC→A: $E_{Ka}[Ks \| ID_B \| N_1 \| E_{Kb}[Ks \| ID_A]]$
  3. A→B: $E_{Kb}[Ks \| ID_A] \| E_{Ks}[M]$
- does not protect against replays
  – could rely on timestamp in message, though email delays make this problematic

Public-Key Approaches

- have seen some public-key approaches
- if confidentiality is major concern, can use:

  $A \rightarrow B: E_{KUb}[Ks] \,||\, E_{Ks}[M]$

  – has encrypted session key, encrypted message
- if authentication needed use a digital signature with a digital certificate:

  $A \rightarrow B: M \,||\, E_{KRa}[H(M)] \,||\, E_{KRas}[T||ID_A||KU_a]$

  – with message, signature, certificate

Digital Signature Standard (DSS)

- US Govt approved signature scheme FIPS 186
- uses the SHA hash algorithm
- designed by NIST & NSA in early 90's
- DSS is the standard, DSA is the algorithm
- a variant on ElGamal and Schnorr schemes
- creates a 320 bit signature, but with 512-1024 bit security
- security depends on difficulty of computing discrete logarithms

DSA Key Generation

- have shared global public key values (p,q,g):
  - a large prime $p = 2^L$
    - where L= 512 to 1024 bits and is a multiple of 64
  - choose q, a 160 bit prime factor of p-1
  - choose $g = h^{(p-1)/q}$
    - where $h<p-1$, $h^{(p-1)/q} \pmod p > 1$
- users choose private & compute public key:
  - choose $x<q$
  - compute $y = g^x \pmod p$

DSA Signature Creation

- to **sign** a message M the sender:
  - generates a random signature key k, $k<q$
  - nb. k must be random, be destroyed after use, and never be reused
- then computes signature pair:

$$r = (g^k \pmod p) \pmod q$$

$$s = (k^{-1}.SHA(M) + x.r) \pmod q$$

- sends signature (r,s) with message M

DSA Signature Verification

- having received M & signature (r,s)
- to **verify** a signature, recipient computes:

$$w = s^{-1} \pmod q$$

$$u1 = (SHA(M).w) \pmod q$$

$$u2 = (r.w) \pmod q$$

$$v = (g^{u1}.y^{u2} \pmod p) \pmod q$$

- if $v=r$ then signature is verified
- see book web site for details of proof why

Authentication Applications

X.509 Authentication Service

Introduction„

ITU-T X.509:

- Part of X.500 Directory Services
- Issued in 1988; revised in 1993 and 1995
- Defines a framework for authentication service using the X.500 directory
- Repository of public-key certificates„
- Based on use of public-key cryptography and digital signatures
- Recommends use of RSA

Public-key Certificates„

- Associated with user
- Created by trusted third party
- Certificate authority (CA)
- Placed in directory by CA or by the user
- Directory server
- ļocation for certificate access
- does not create the certificates

X.509 Certificate Format

The general format for a certificate is:

- Version V
- Serial number SN
- Signature algorithm identifier AI
- Issuer Name CA
- Period of Validity T_A

- Subject Name A
- Subject's Public-key Information A_p
- Issuer Unique Identifier (added in Version 2)
- Subject Unique Identifier (added in Version 2)
- Extensions (added in Version 3)
- Signature

X.509 Standard Notation

- User certificates generated by a CA use the following standard notation:
- $CA\langle\langle A\rangle\rangle = CA\ \{V, SN, AI, CA, T_A, A, A_p\}$

where

$Y\langle\langle X\rangle\rangle$ = the certificate of user X issued by the certification authority Y

$Y\ \{I\}$ = the signing of I by Y consisting of I with an encrypted hash code appended.

X.509: Obtaining A User Certificate

user certificates generated by a CA have the following characteristics:

- Any user with access to the public key of the CA can recover the user public key that was certified.
- No party other than the CA can modify the certificate without being detected.
- Since they are unforgeable,they can be placed in a directory without the need for the directory to make special efforts to protect them.

X.509: CA Trust Issues

- If all users subscribe to the same CA, then there is a common trust of that CA.
- All user certificates can be placed in the directory for access by all users.
- Any user can transmit his/her certificate directly to other users.
- Once B is in possession of A's certificate, B has confidence that:
- Messages it encrypts will be secure.
- Messages signed with A's private key are unforgivable.

X.509: Multiple CAs

- Large User Community
- Not Practical to Support All Users
- More Practical to Have Multiple CAs
- Each CA Provides Its Public Key to A Smaller User Group

X.509: Authentication Procedures

Three alternative authentication procedures for X.509 Directory Authentication Service

- Each use public-key signatures
- Each assumes that two parties know each other's public key.
- either obtained from Directory
- or obtained in an initial message

Email Security

- Email security is dealing with issues of unauthorized access and inspection of electronic mail. This unauthorized access can happen while an email is in transit, as well as when it is stored on email servers.

- Email has to go from many untrusted servers to reach to its destination and one can intercept or modify it to harm the sender or to make some profit.

CIA for Email (Yeah! Again CIA ☺)

- Confidentiality: Email should be only viewed by the person it is intended to.
- Integrity: Original content should be received by the receiver.
- Availability: Receiver should be able to access the mail any time he requires.

Steps to secure our Emails

- Security at sender's side
- Security at Receiver's side
- Secure transmission of emails

Security at sender's side

- Can be implemented by non-technical person
- Use incognito mode while sending mails
- Avoid using public computers

Security at receiver's side

- Avoid downloading attachments from unknown sender's
- Check Email Headers to verify identity of sender

Secure Transmission of Emails

- ✓ PGP (**P**retty **G**ood **P**rivacy)
- ✓ S/MIME

(**S**ecure/**M**ultipurpose **I**nternet **M**ail **E**xtension)

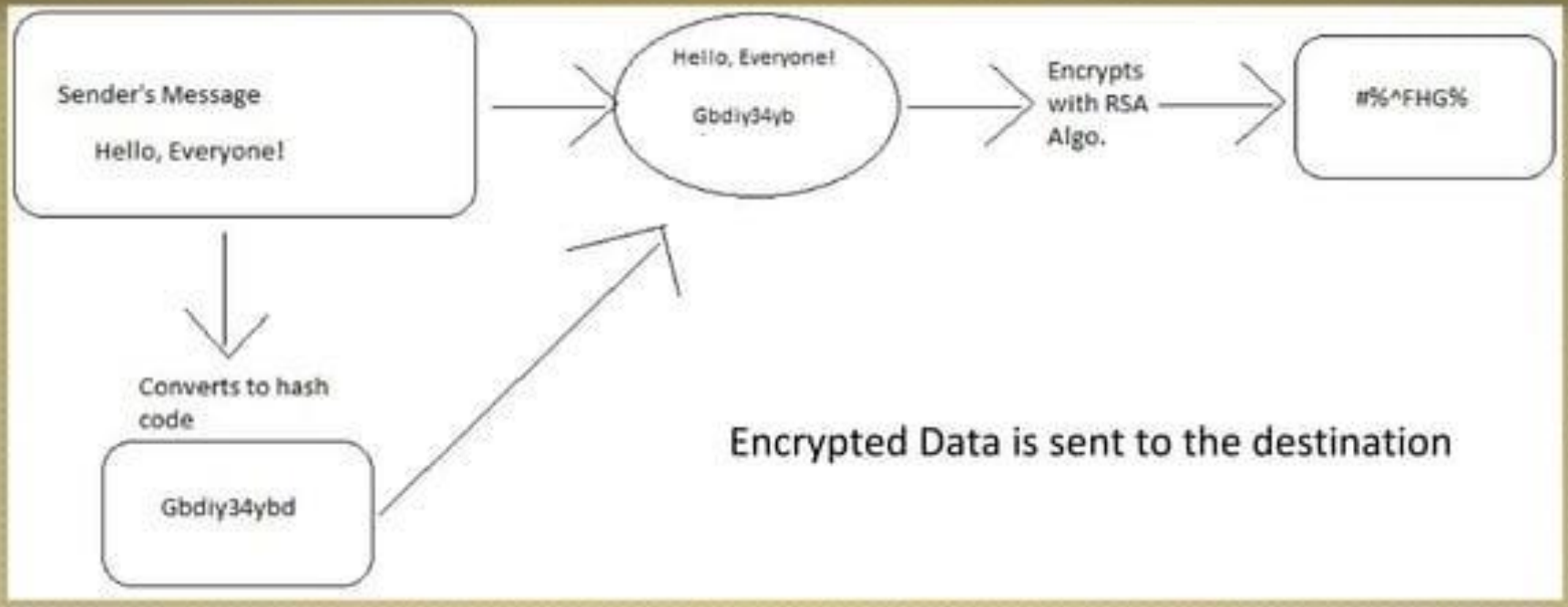
PGP

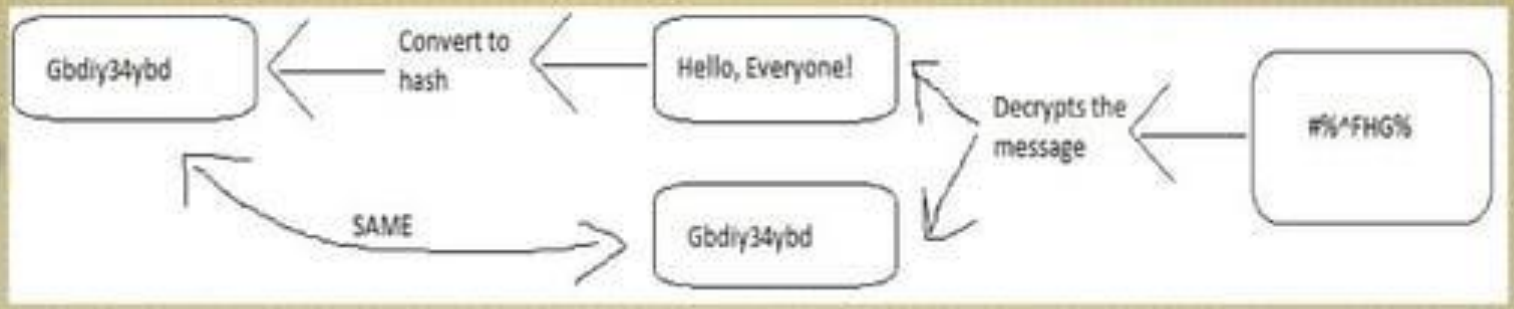
- Pretty Good Privacy
- PGP provides a confidentiality and authentication service that can be used for electronic mail and file storage applications.
- Available free worldwide
- Based on extremely secure algorithm
- Not developed by governmental organization

PGP: Services

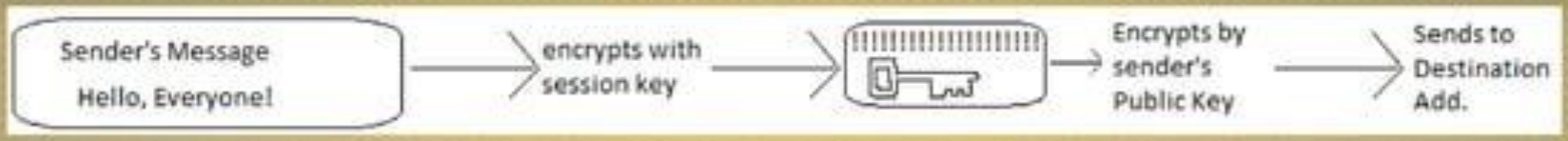
- ✓ Authentication
- ✓ Confidentiality
- ✓ Compression
- ✓ Email Compatibility
- ✓ Segmentation

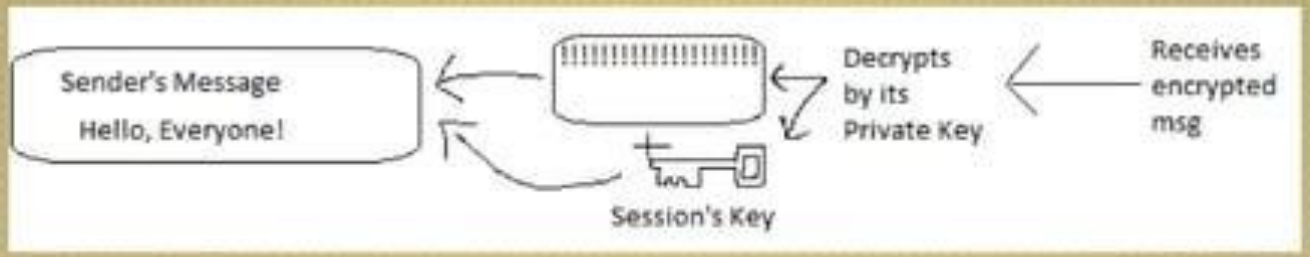
PGP: Authentication





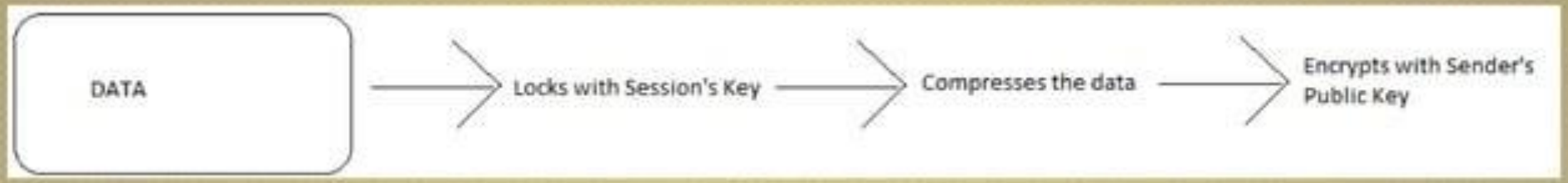
PGP: Confidentiality





PGP: Compression

- Compresses the data before encrypting
- Compression is done after signing (Locking with session key)
- Use ZIP Compression Algorithm



PGP: Email Compatibility

- Binary Data is obtained after applying PGP
- Converted to ASCII to able to send it over mail
- Uses Radix64 Algorithm for conversion

NOTE: PGP divides big emails in smaller sizes just before sending. (Segmentation)

S/MIME

- Secure / Multipurpose Internet Mail Extensions
- S/MIME is standard for exchanging secure mails with the help of encryption
- Previously, Mails were supposed to carry text only
- S/MIME provides support for varying content
- Supported by major email programs like Outlook, Netscape

S/MIME: Signed Mail

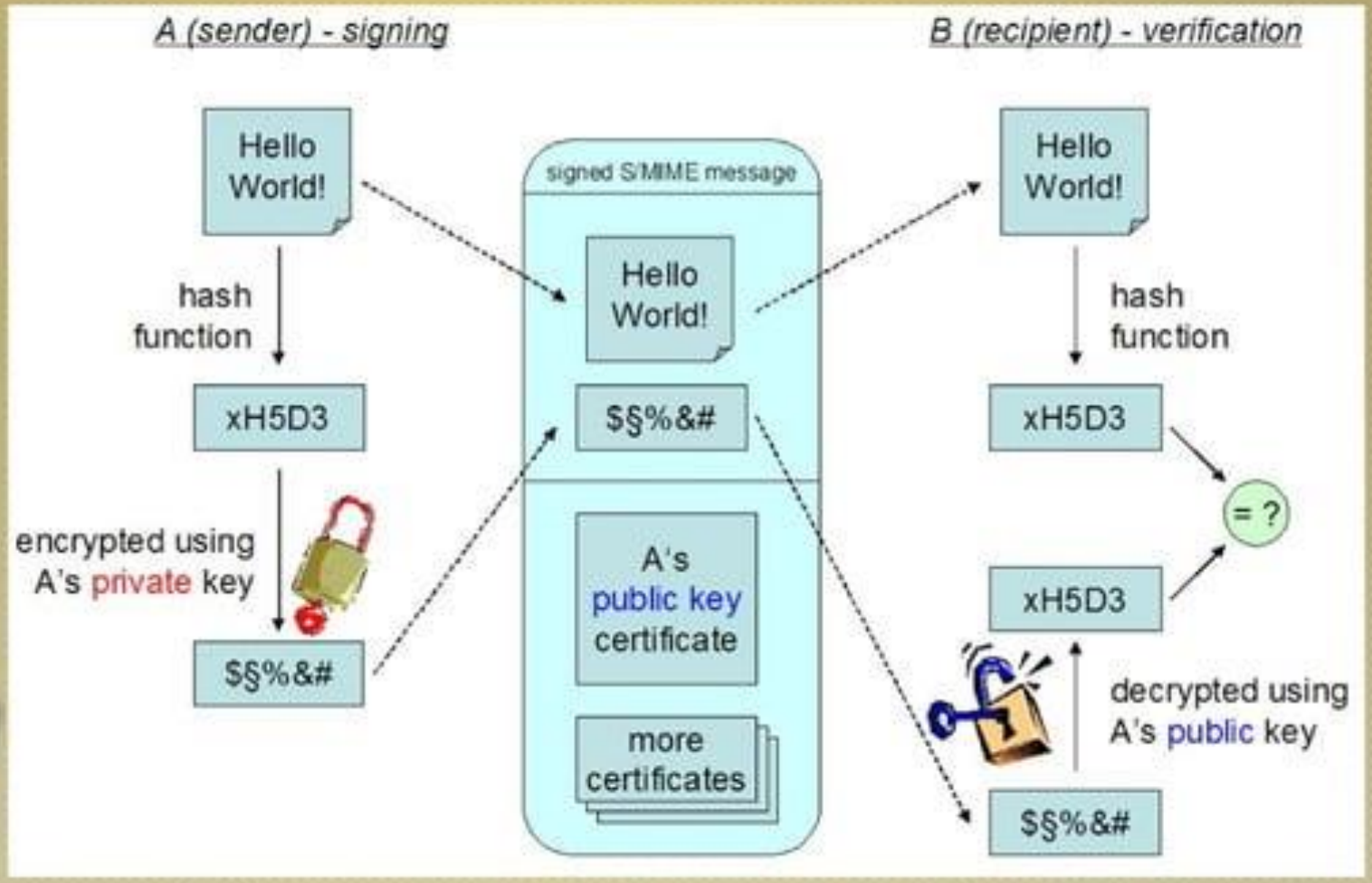


Image copied from internet (Not the complete PPT :P)

S/MIME: Encrypted Mail

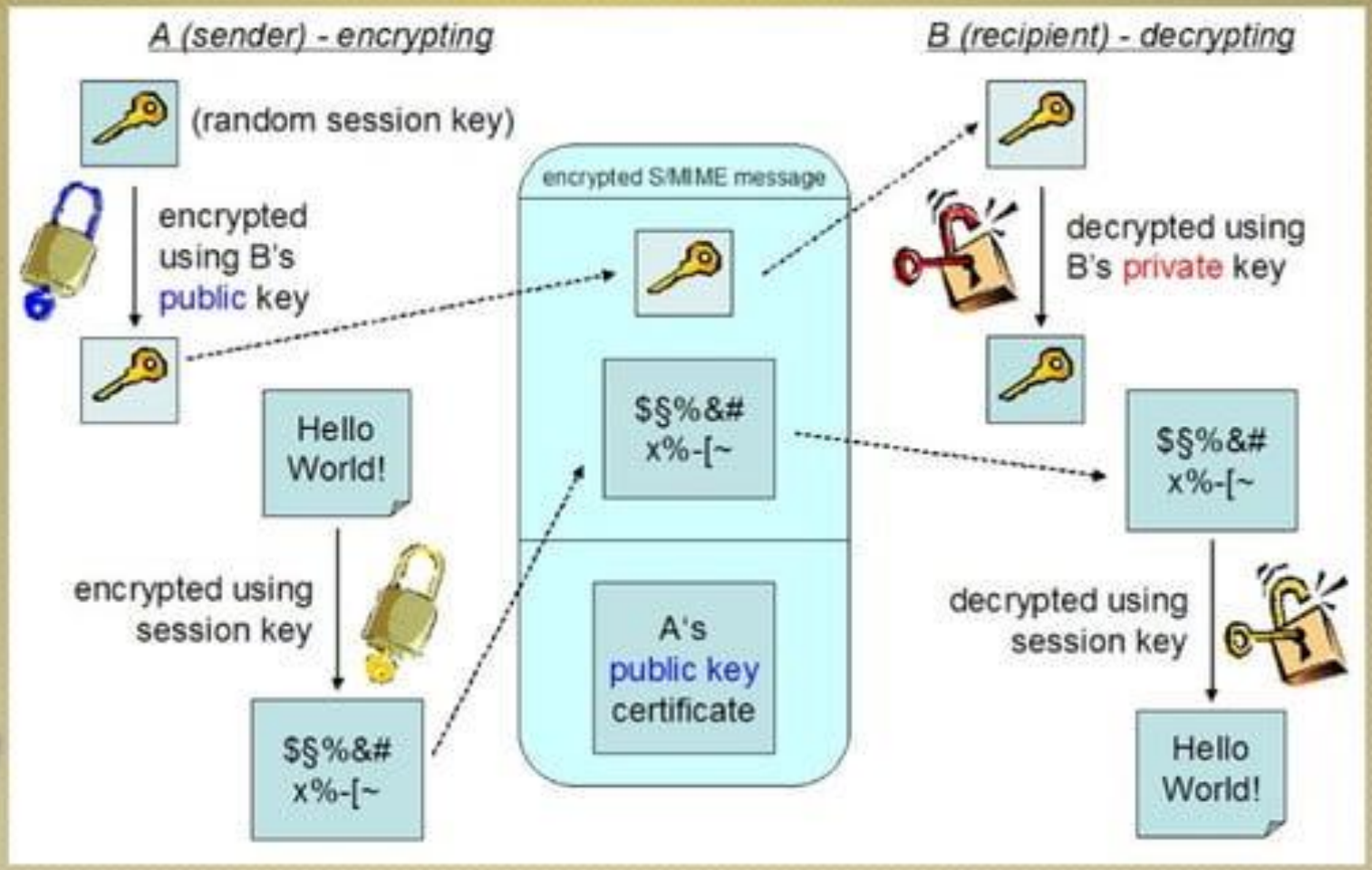


Image copied from internet (Not the complete PPT :P)

S/MIME: Functions

- Enveloped Data : Encrypted content and Associated keys
- Signed Data : Encoded message + Signed digest
- Clear-signed data : Clear text message + Encoded signed digest
- Signed & Enveloped Data : Nesting of signed & encrypted entities