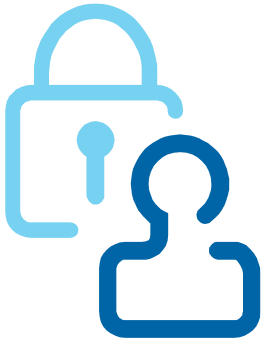


EBU714U

Security and Authentication

December 2020

Dr Na Yao

na.yao@qmul.ac.uk

School of Electronic Engineering & Computer Science,

Queen Mary University of London

Review

- **Services: Authentication Applications**
  - Kerberos
- **Certificates (Key Authentication)**
  - X.509
- **IP Security (IPSec)**
  - IPSec proper (authentication and encryption)
  - IPSec key management
- **Firewalls**

This week..

- Web Security (TLS/SSL)
- Email Security (PGP, S/MIME,..)
- Threats to Security
- Course Summary & Revision

Web Security

TLS/SSL

The Web

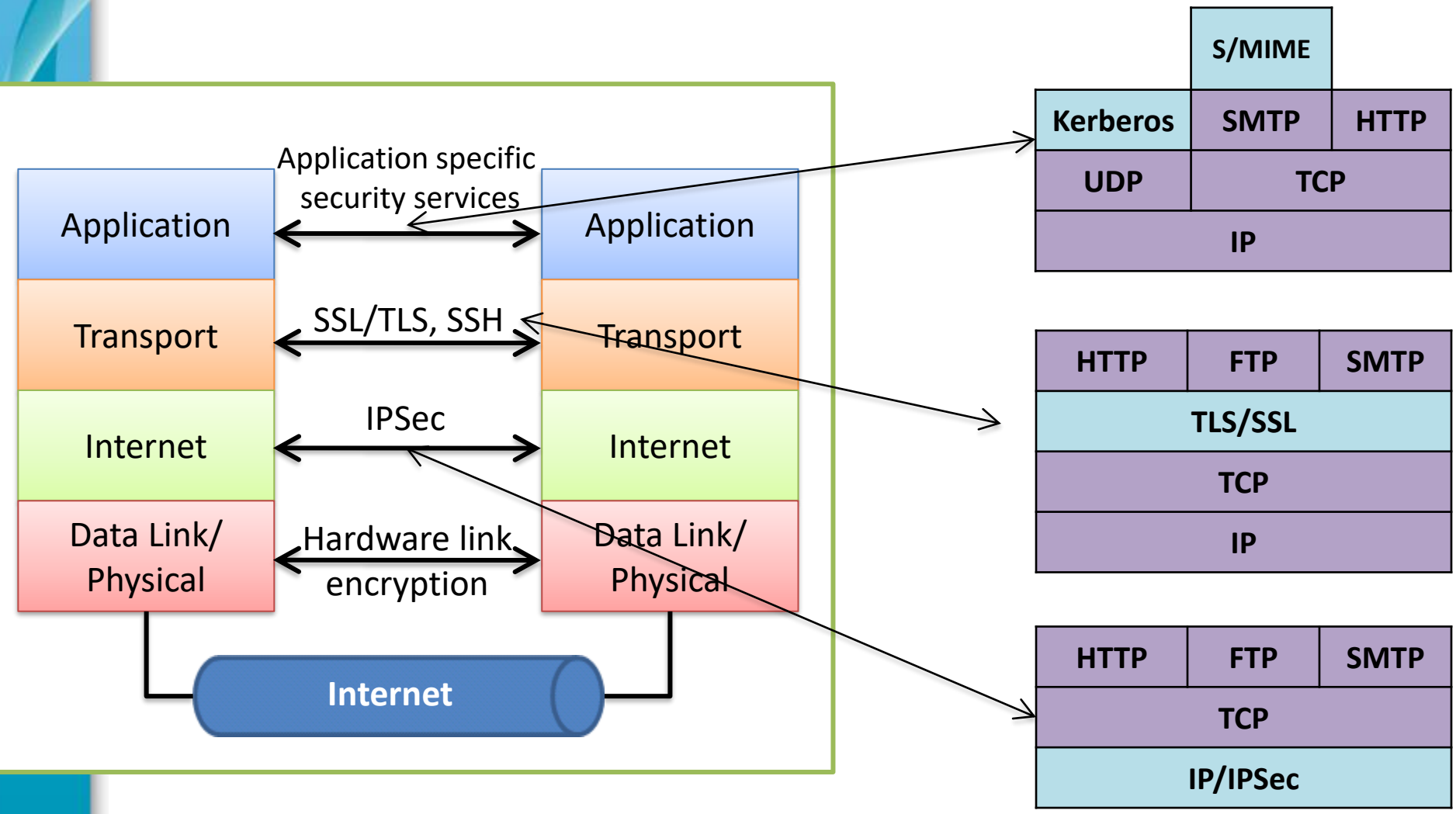
Did we trust early web developments for E-commerce applications?

- **Is a client/server application running over the Internet using TCP/IP**
  - Web browsers are easy to use and configure, but the underlying software is complex. This complexity may hide potential security flaws.
  - A Web server can be the launching pad for the LAN that contains the server.
  - In general users are not aware of security risk or do not have the know-how to take effective countermeasures.

Threats

	Threats
Integrity	Modification of user data/memory/message traffic in transit, Trojan horse browser.
Confidentiality	Eavesdropping, theft of info, data. Info about network configuration or identity of client/server.
Denial of Service	Killing user threads, flooding with bogus threats, filling up disk/memory, Isolating machine by DNS attacks
Authentication	Impersonation of legitimate users, data forgery

Web Traffic: Security Services

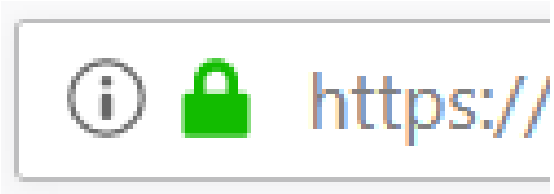


Transport Layer Security (TLS)

derived from Secure Sockets Layer (SSL)

History of SSL & TLS

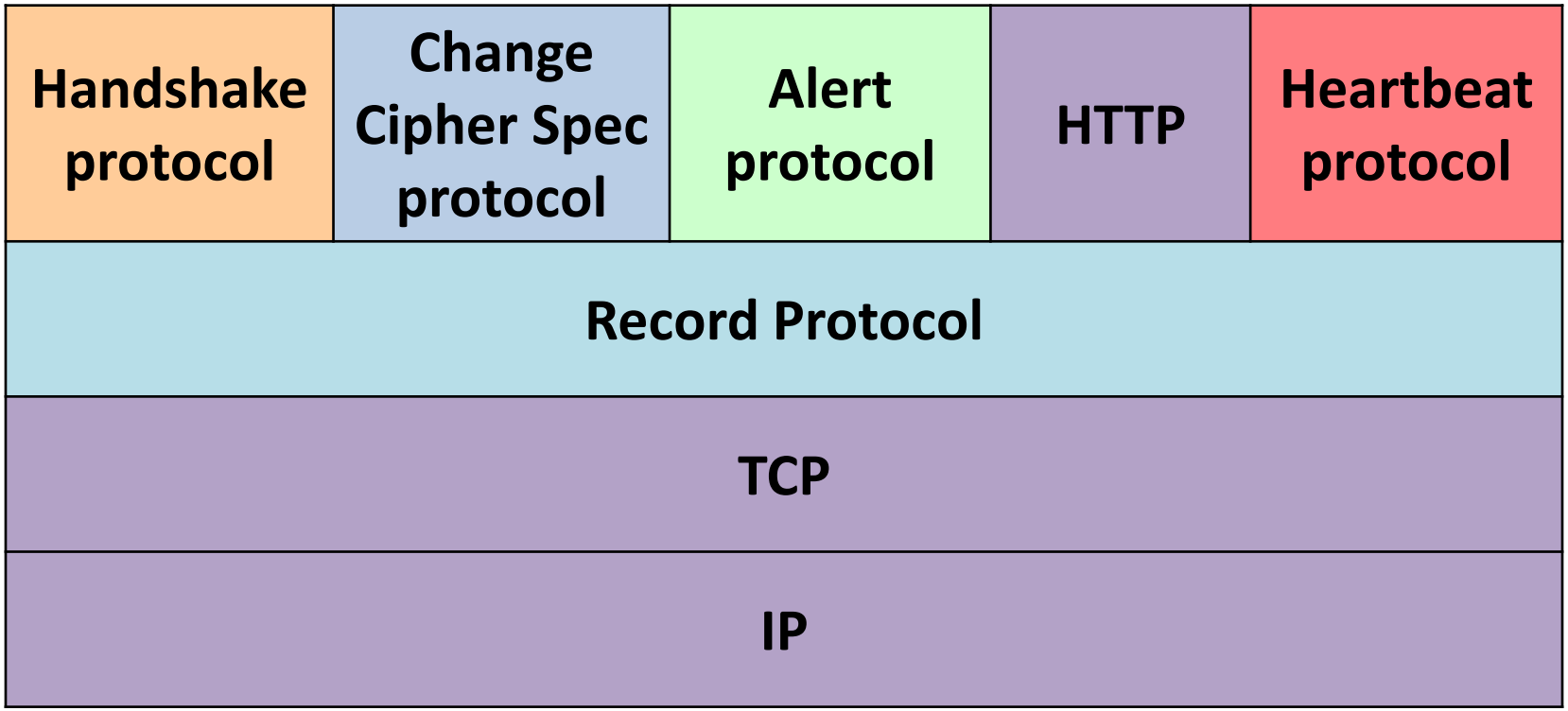
- SSL was first developed by Netscape the mid-1990s.
- The IETF released protocol versions *TLS 1.0* in 1999, *TLS 1.1* in 2006, and *TLS 1.2* in 2008. The latest version, which is significantly different, is *TLS 1.3*.
- TLS runs over the *Transmission Control Protocol* (TCP).



- All SSL versions are *deprecated* now (SSL 3.0 was prohibited in June 2015), however SSL may still appear in documents.

TLS Architecture

- Two layers of protocols



TLS Architecture

- Handshake Protocol
- Record Protocol
- Change Cipher Spec Protocol
- Alert Protocol
- Heartbeat protocol

TLS Connection

- A **connection** in TLS is a transport that provides suitable type of service (they are peer-to-peer and transient). Every connection is associated with one session.
- A connection state is defined by the following parameters:
  - Server and client random (i.e. nonce)
  - Server write MAC secret
  - Client write MAC secret
  - Server write key
  - Client write key
  - Initialisation vectors (i.e. IVs for CBC encryption)
  - Sequence number

TLS Session

- A **session** in TLS is an association between a client and a server.
- Sessions are created by the Handshake Protocol.
- Sessions define a set of cryptographic security parameters, which can be shared among multiple connections.
- Sessions are used to avoid expensive renegotiation of security parameters for each connection.

A TLS Session state is defined by:

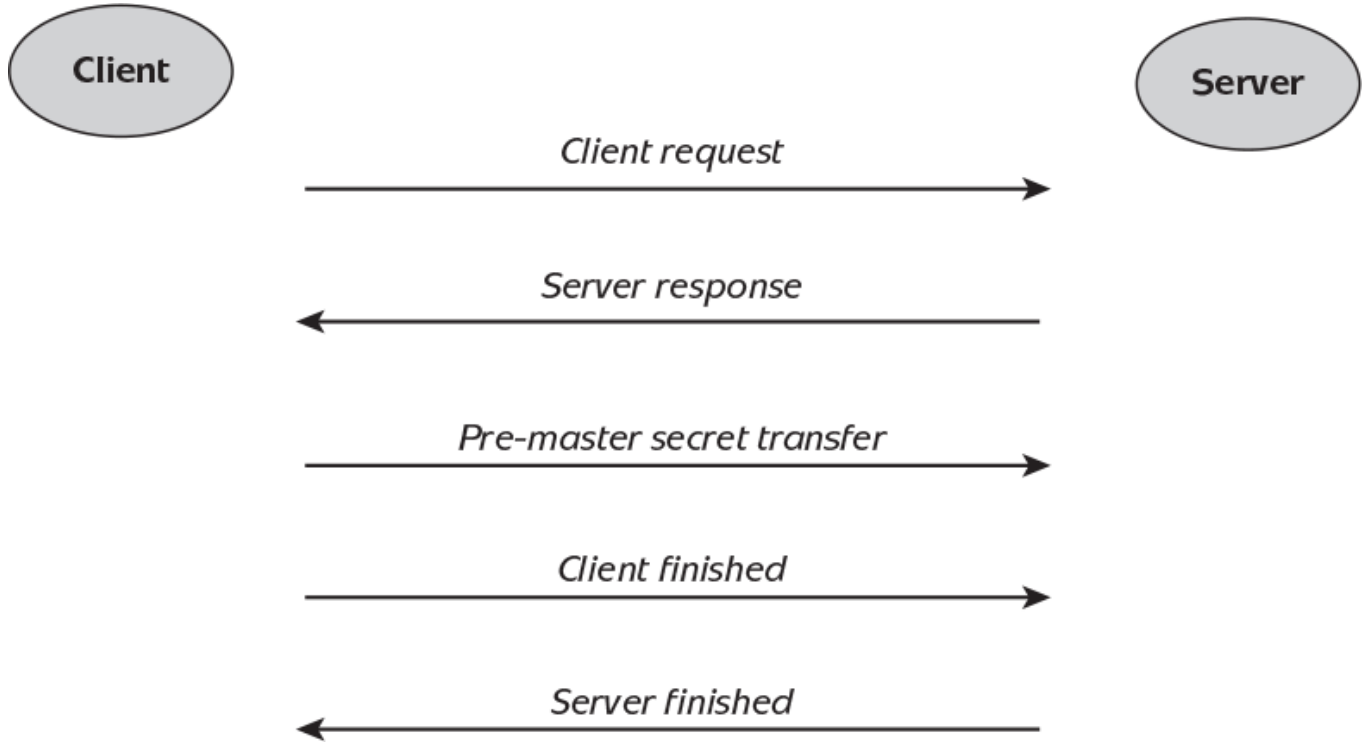
- Session identifier
- Peer certificate (X509.v3) – authentication; to create trust
- Compression method
- Cipher Spec (null, DES, MD5, SHA-1, …)
- Master secret – to authenticate (& relate) the connection to a session
- Is resumable – a flag indicating whether the session can be used to initiate new connections

Handshake protocol

- This protocol performs all the tasks requiring agreement between the two entities before they set up the secure TLS channel:
  - agree on the *cipher suite* to be used to establish the secure channel;
  - allows the server and client to authenticate each other; and
  - establish the keys needed to secure the channel.

- *Cipher suite*: a list that contains the combinations of cryptographic algorithms.

Handshake protocol (A simple version for TLS 1.2 and earlier versions)



- ***Client Request***:
  - a *session ID*: a unique identifier for the session;
  - a pseudorandom number (nonce) r_C: for the provision of freshness; and
  - a list of cipher suites the client supports (including *key exchange* method)

Supported Key exchange methods

- ***RSA***
- ***Fixed Diffie-Hellman***: Diffie-Hellman public parameters contained in server's certificate, signed by CA.
- ***Ephemeral Diffie-Hellman***:
  - Sender generates a fresh set of parameters, and sends the public values alongside a digital signature on the chosen parameters.
  - This creates ephemeral(temporary, one-time) secret keys, and considered the most secure DH option.
  - Offers perfect *forward secrecy*.
- ***Anonymous Diffie-Hellman***: Basic Diffie-Hellman used without authentication.

Handshake protocol (A simple version for TLS 1.2 and earlier versions)

Server Response:

- the session ID;
- Server's nonce r_S;
- the particular cipher suite the server has decided to use;
- a copy of the server's public-key certificate; and
- if the Ephemeral Diffie–Hellman is chosen, then the server also generates a fresh set of parameters, and sends the public values alongside a digital signature on the chosen parameters.

After receiving server response message, *client* need to

- check the server's public-key certificate is valid
- If the Ephemeral Diffie–Hellman protocol is being used, then the client should verify the digital signature on the Diffie–Hellman parameters.

Handshake protocol (A simple version for TLS 1.2 and earlier versions)

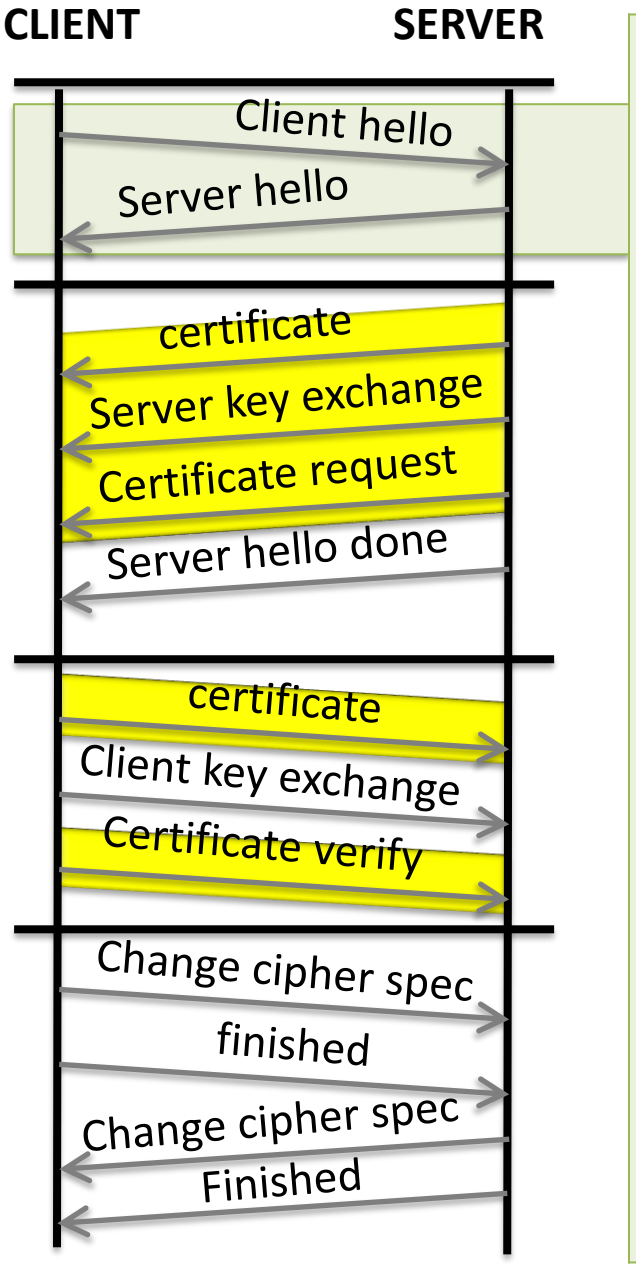
- ***Pre-master Secret Transfer***: The client and server now need to agree on a shared secret K_P (the *pre-master secret)*.
  - **RSA**: the client generates K_P, encrypted using the server's public key and sends to the server;
  - **Ephemeral Diffie–Hellman**: the client generates a fresh temporary Diffie–Hellman key pair and sends the public value to the server, after which both client and server compute the shared secret K_P.
- The client and server can now derive the keys required to secure the TLS session:
  - compute the *master secret* K_M using a key derivation function, taking K_P, r_C and r_S as part of inputs.
  - derive MAC and encryption keys from K_M. From this point on, all exchanged messages are cryptographically protected.

Handshake protocol (A simple version for TLS 1.2 and earlier versions)

- ***Client Finished***.
  - The client computes a MAC on the hash of all the messages sent thus far.
  - This MAC is then encrypted and sent to the server.

- ***Server Finished***.
  - The server checks the MAC received from the client.
  - The server then computes a MAC on the hash of all the messages sent thus far.
  - This MAC is then encrypted and sent to the client.

Handshake Protocol: Phase 1

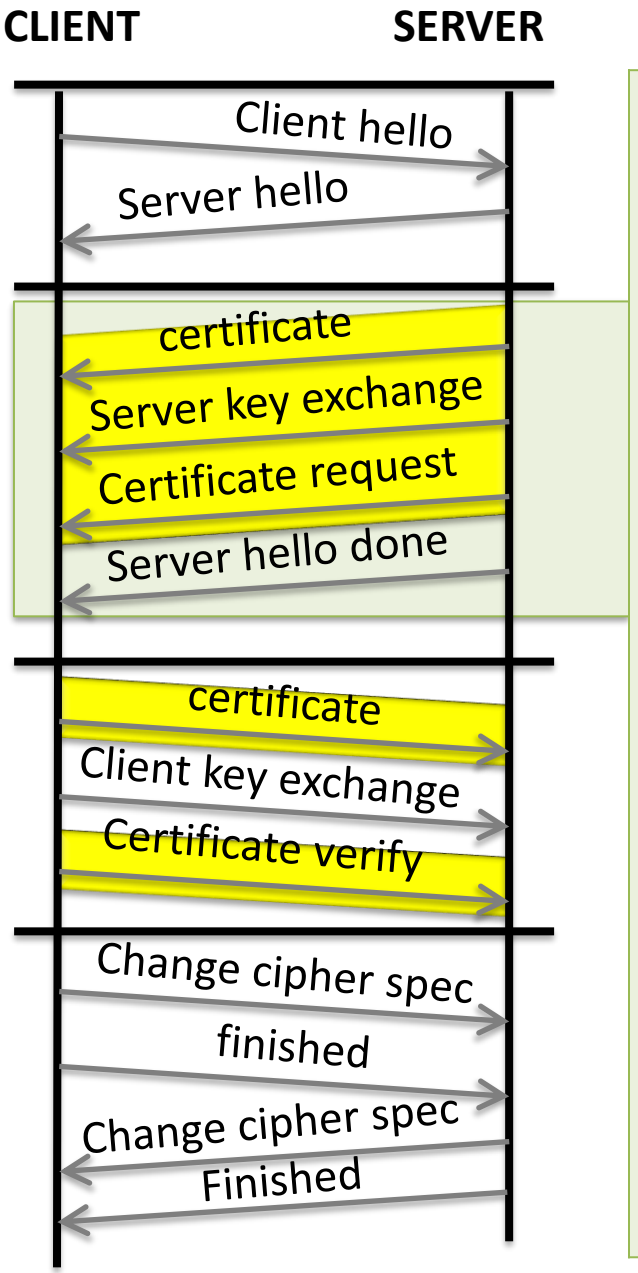


Establish Security Capabilities

Client hello = Version (The highest TLS version understood by the client),
Nonce,
Session ID,
Compression method
Cipher Suite
***Key Exchange**
RSA
Fixed Diffie-Hellman
Ephemeral Diffie-Hellman
Anonymous Diffie-Hellman
***CipherSpec**
Cipher Algorithm (RC4,3DES, AES..)
*MAC algorithm (SHA)
*Cipher Type (block, stream)
*IsExportable
*Hash size
*Key Material (data used in generating write keys)
*IV size

Server hello = replies choosing from the client list a set of algorithms and parameters.

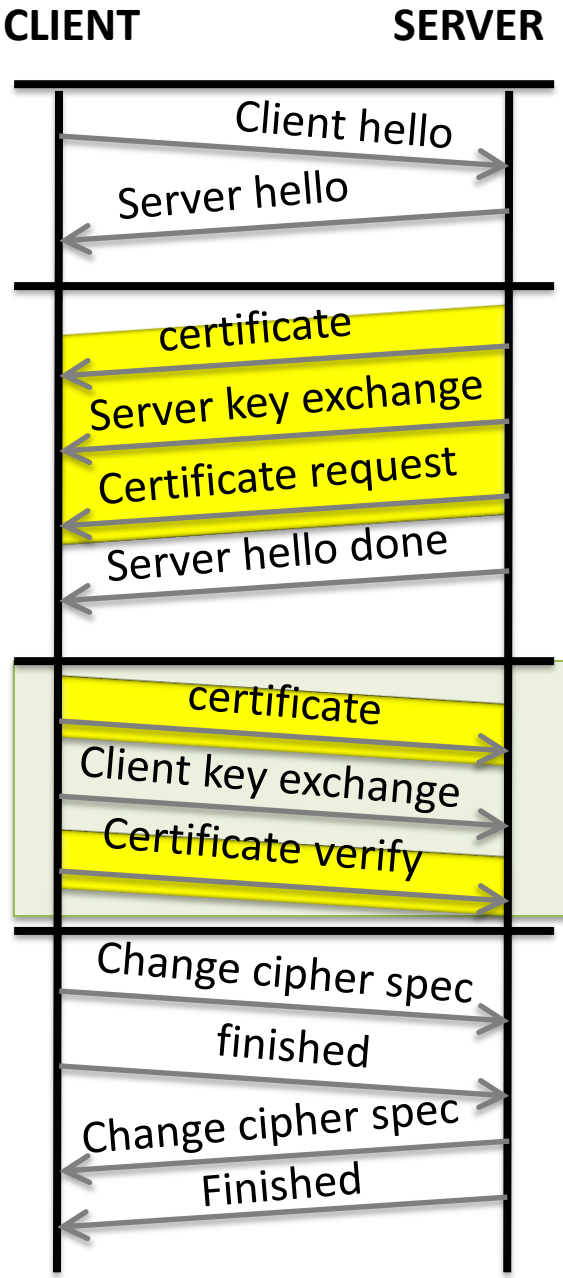
Handshake Protocol: phase 2



Authentication and Key exchange

Certificate	Contains Server's certificate (X.509)
Server key exchange	Needed if • Anonymous Diffie-Hellman • Ephemeral Diffie-Hellman • RSA key exchange (Signature only RSA key)
Certificate request	Server requests a certificate from client (optional) • Certificate type • Certificate authority
Server hello done	Indicate the end of Serve Hello and associated messages

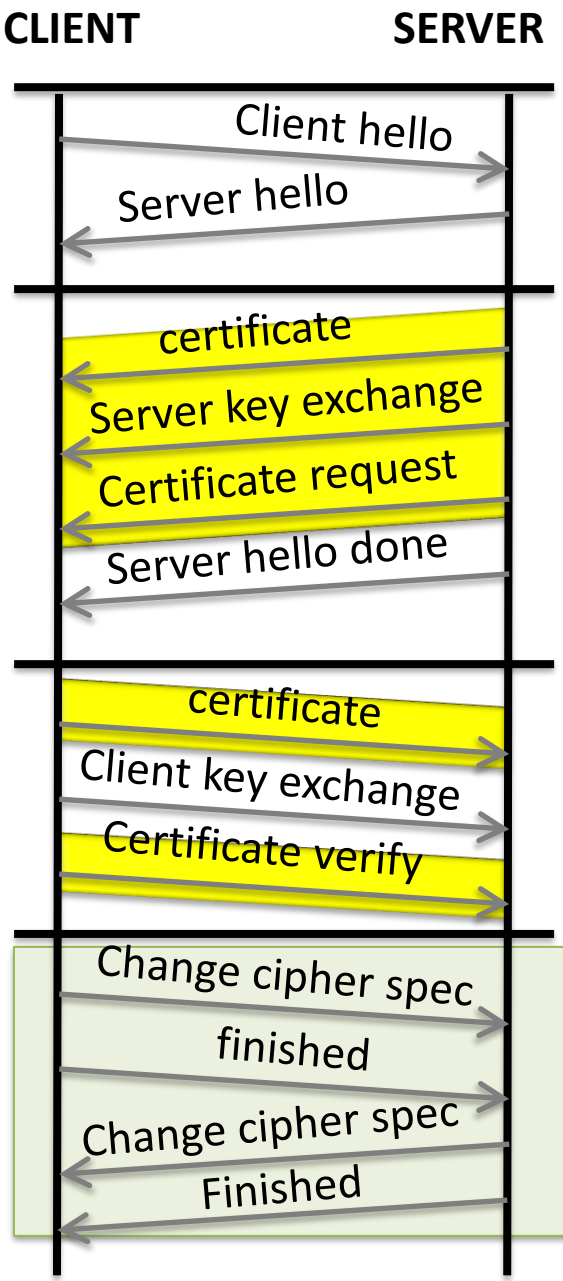
Handshake Protocol: phase 3



Client Authentication and Key exchange

Certificate message	Send the requested certificate
Client key exchange	Depending on the key exchange mechanism • RSA • Diffie-Hellman (ephemeral and Anonymous) • Fixed Diffie-Hellman
Certificate verify	Verification of clients certificate

Handshake Protocol: phase 4



Finish (Change Cipher Spec)

Change cipher spec (Client)	Copies the pending CipherSpec into the current CipherSpec
Finished	Verifies the key exchange and authentication processes to be successful
Change cipher spec (Server)	Copies the pending CipherSpec into the current CipherSpec
Finished	Verifies the key exchange and authentication processes to be successful

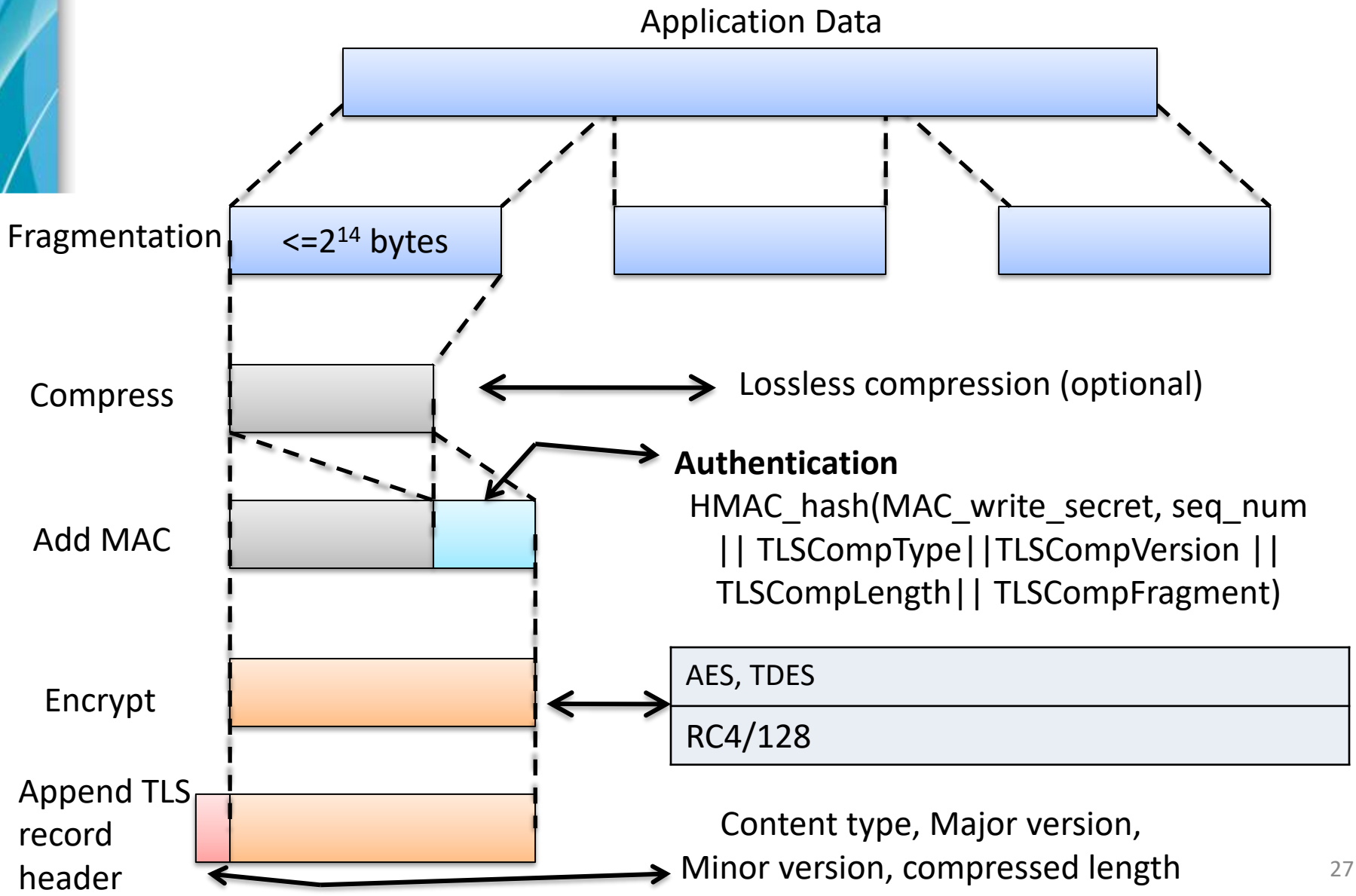
Change Cipher Spec Protocol

- This is essentially the last phase of Handshake protocol.
- Change of cipher suites
  - Sends one message which updates the cipher suite to be used on this connection. The message is a single byte with value 1.

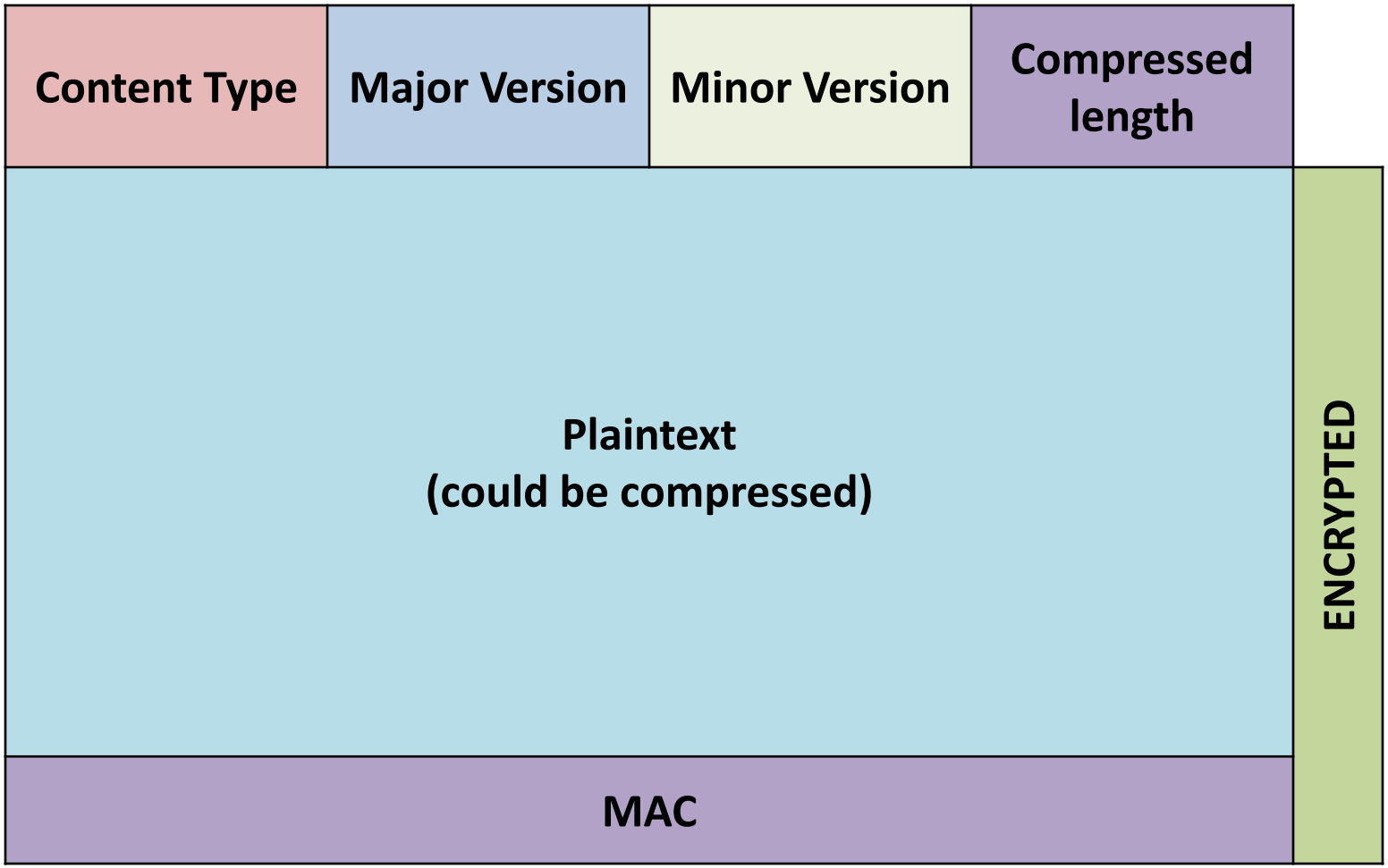
TLS Record Protocol

- TLS Record protocol provides:
  - Confidentiality
  - Message Integrity

TLS Record Protocol



TLS Record Format



Alert Protocol

- Used to convey TLS-related alerts to the peer entity.
- Consist of two bytes, the first byte flags a
  - Warning or
  - Fatal
- The second byte contains the code that indicates the specific alert, e.g.
    - bad_record_mac
    - handshake_failure
    - decryption_failed
    - bad_certificate; etc

Heartbeat protocol

- A **heartbeat** is a periodic signal generated to indicate normal operation or to synchronise.
- A heartbeat protocol is typically used to monitor the availability of a protocol entity. (In the specific case of TLS, a heartbeat protocol was defined in 2012 in RFC6250.)
- Two purposes:

  1. assures the sender that the recipient is still alive, even though there may not be any activity for a while.

  2. avoid closure by a firewall that does not tolerate idle connections.

TLS attacks

- Attacks on the handshake protocol
- Attacks on the record and application data protocols
  - chosen-plaintext attack, session hijacking
- Attacks on the PKI
- Other attacks
  - Heartbleed (buffer over-read)

TLS 1.3

- Various attacks on earlier versions of TLS resulted in a series of fixes having to be proposed, which is not desirable.
- The Handshake Protocol is somewhat inefficient.
- Major revision of TLS resulted in TLS 1.3 published in August 2018 (RFC8446).
- What is new in TLS 1.3**:**
  - **Perfect Forward Secrecy**. removing support for key establishment based on RSA and mandating the use of Ephemeral Diffie–Hellman.
  - **New Handshake Protocol**. only requiring one full round trip between client and server. More of the data exchanged in the new Handshake Protocol is encrypted.
  - **Authenticated encryption modes**. Encryption in TLS 1.3 must be conducted using an authenticated-encryption mode of a block cipher.

Email Security

Email Security

- Basic requirements
  - Confidentiality
  - Authentication
  - Integrity
- Other requirements
  - Non-repudiation
  - Proof of submission
  - Proof of delivery
  - Anonymity
  - Revocability
  - Resistance to traffic analysis

Pretty Good Privacy (PGP)

- Largely the effort of one person, Phil Zimmerman. Released in 1991 and distributed worldwide via Usenet post.
  - Uses the best available cryptographic algorithms.
  - Easy-to-use and platform independent.
  - PGP is free (even the source), easy to get and it is documented.
  - Wide range of applicability, from corporations to individuals.
  - Is not controlled by any governmental or standards organisation.

PGP

Actual operations of PGP consist of **five services**:

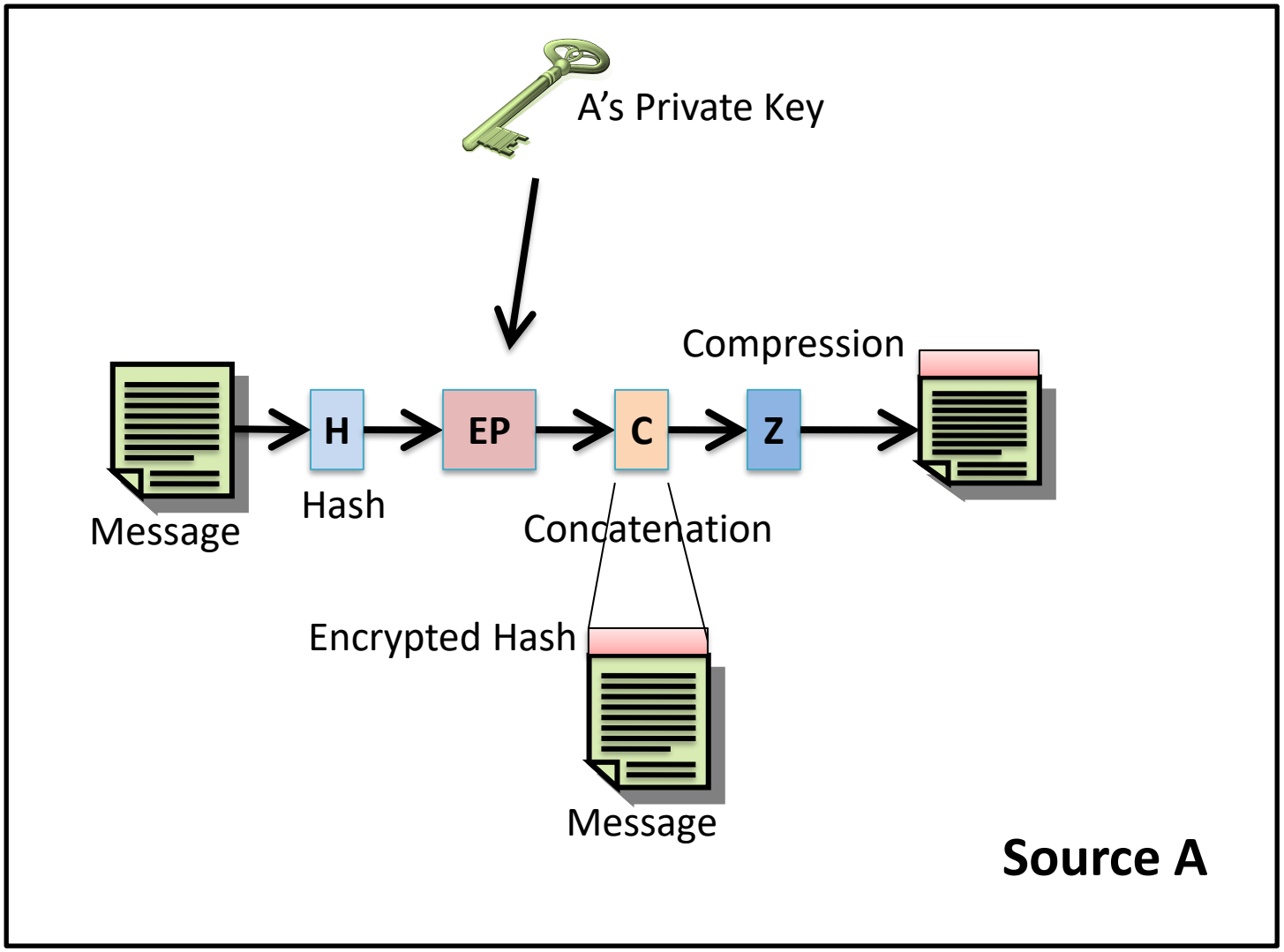
- **Authentication** - DSS/SHA or RSA/SHA
- **Confidentiality** - CAST or IDEA or RSA or 3DES
- **Compression**: A message may be compressed, for storage or transmission using ZIP
- **E-mail compatibility**: To provide transparency for e-mail applications, an encrypted message may be converted to an ASCII using Radix-64
- **Segmentation**: To accommodate maximum message size limitations, PGP performs segmentation and reassembly.

PGP

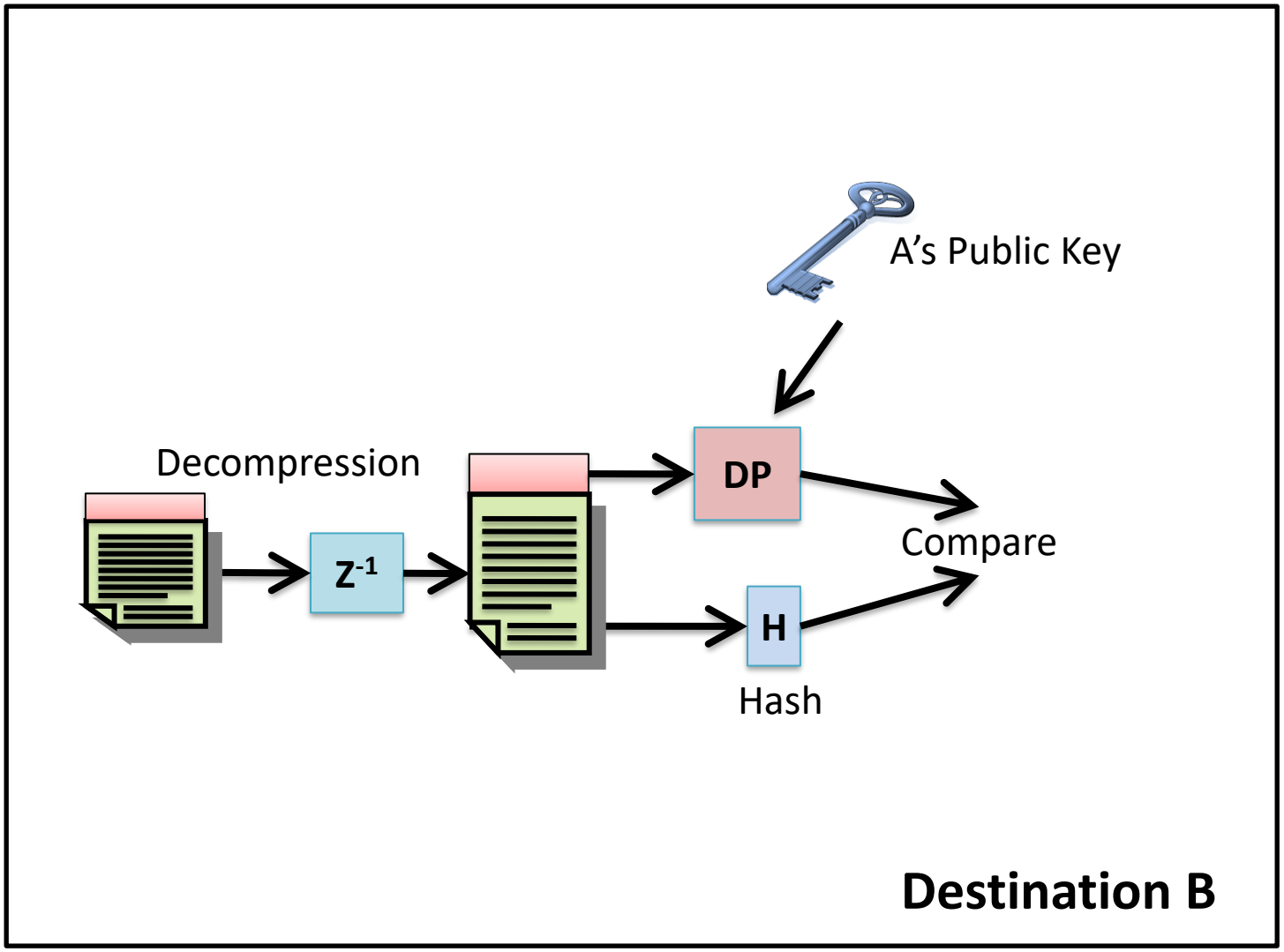
- The latest PGP services

Function	Algorithm
Digital Signature	DSS/SHA or RSA/SHA
Message Encryption	CAST or IDEA Triple DES. With Diffie-Hellman or RSA for key exchange
Compression	ZIP
e-mail compatibility	Radix-64
Segmentation	-

Authentication



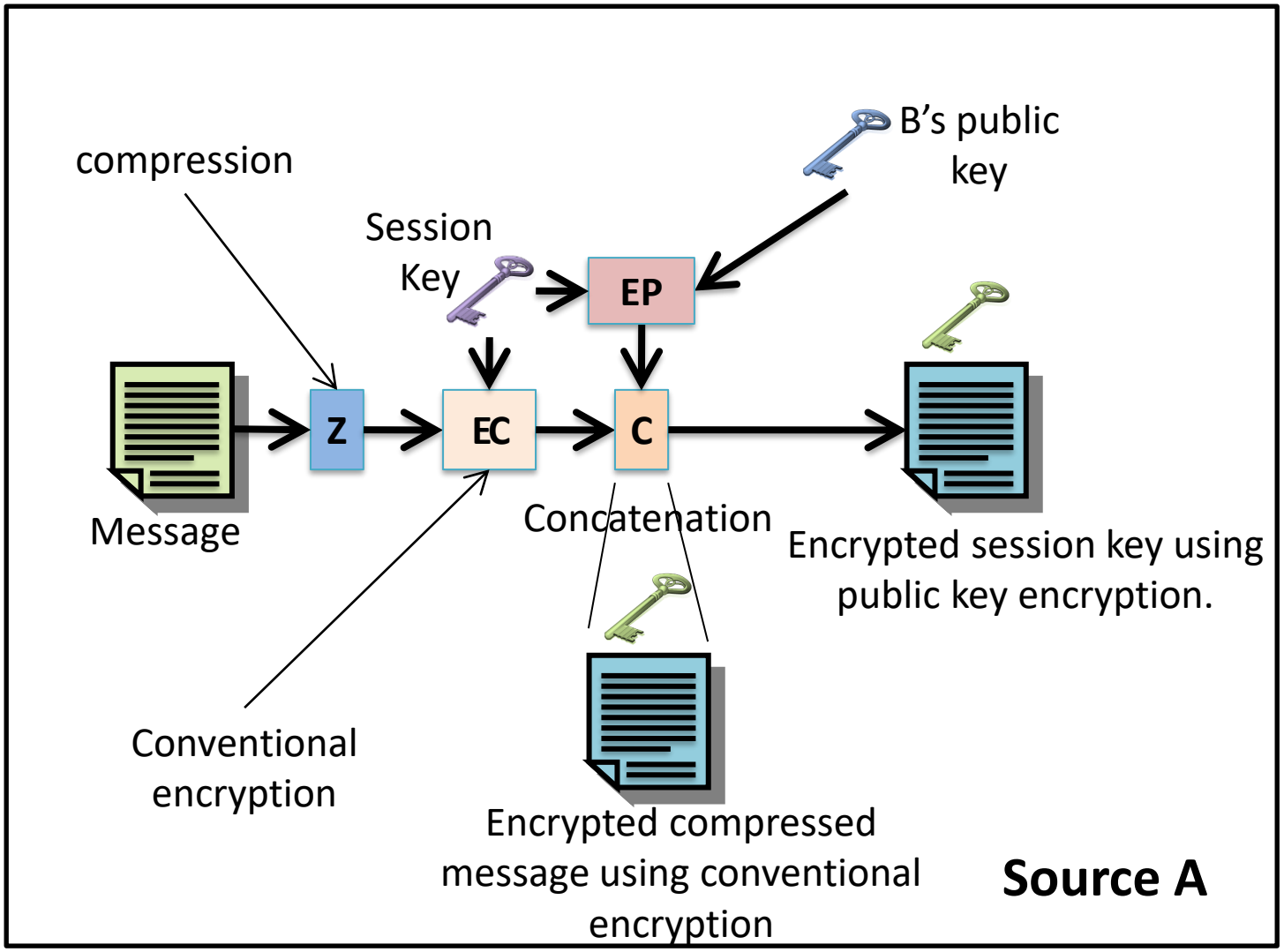
Authentication



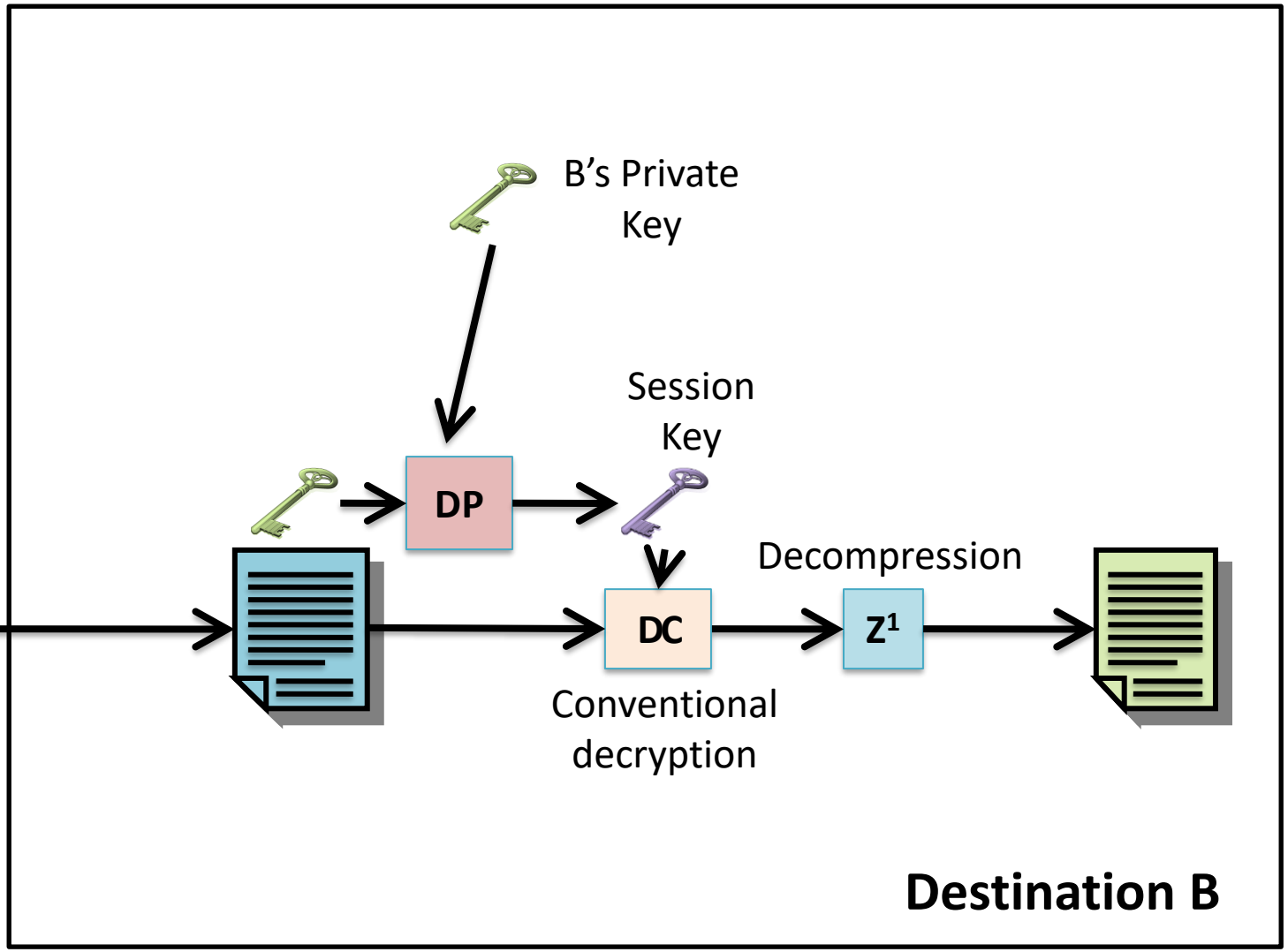
Authentication

- Create a message.
- SHA used to generate 160-bit hash code.
- Hash encrypted with RSA using sender's private key, the result is prepended to the message.
- The receiver uses RSA with sender's public key.
- The receiver decrypts the hash with the sender's public key.
- The receiver generates a new hash code from the received message and compares it with the decrypted hash to prove authenticity.

Confidentiality



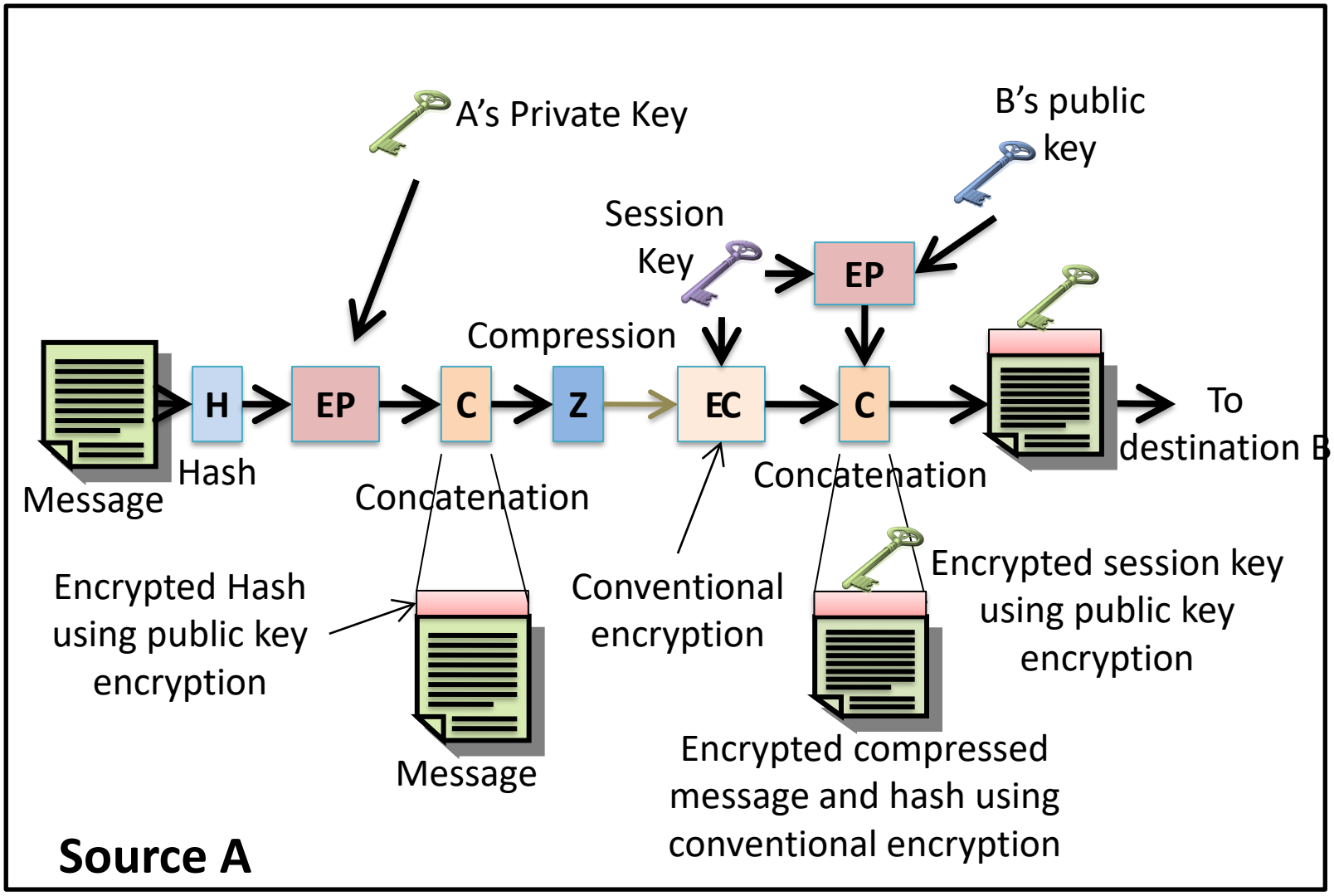
Confidentiality



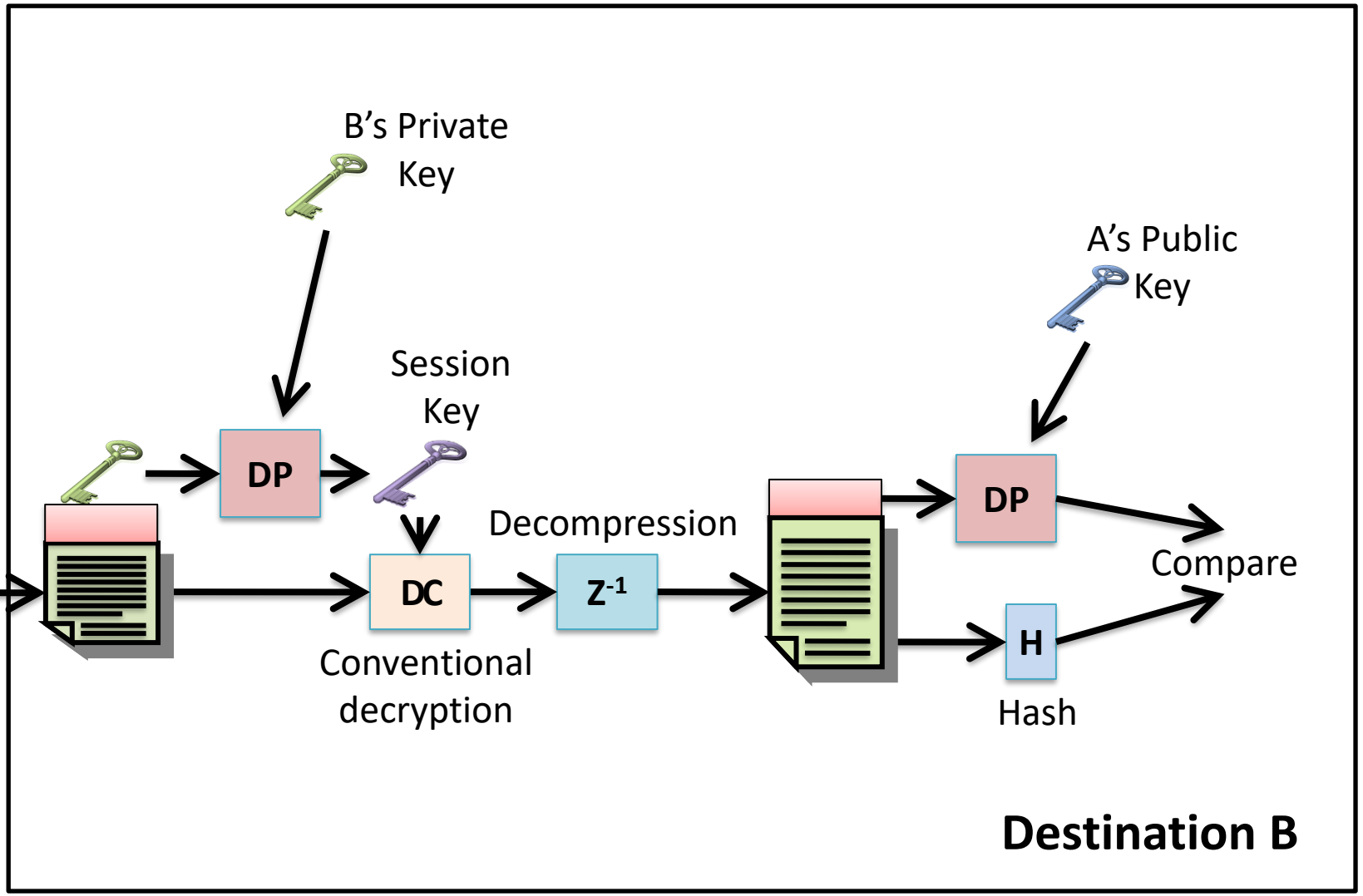
Confidentiality

- Sender generates a message.
- Sender generates a session key (128-bits long).
- Message is encrypted using CAST-128 (or IDEA, TDEA) using the session key.
- The session key is encrypted using RSA and the recipient's public key. The encrypted session key is prepend to the message.
- The receiver uses RSA and its' private key to recover the session key.
- Using the session key, the receiver decrypts the message.

Authentication & Confidentiality



Confidentiality & Authentication



Compression

To save space both for e-mail and storage as a default PGP compresses the message **after** applying the signature but **before** encryption.

- If the message was first compressed and then signed then for future verification
  - A compressed version of the document has to be stored, or
  - Re-compress the message when verification is required.

Compression

- But more relevantly, a compression algorithm is **not** deterministic.
  - That is the same message when compressed can produce different compressed forms (this depends on running speed vs compression ratio).
  - If sender and receiver use different settings for the compression algorithm they obtain different forms. This makes authentication difficult.

- The compressed message has less redundancy so is more difficult to cryptanalysis.

E-mail compatibility

- When PGP transmits a message at least part of the message is encrypted. The encrypted part (can be the whole document) consist of a stream of 8-bit octets.

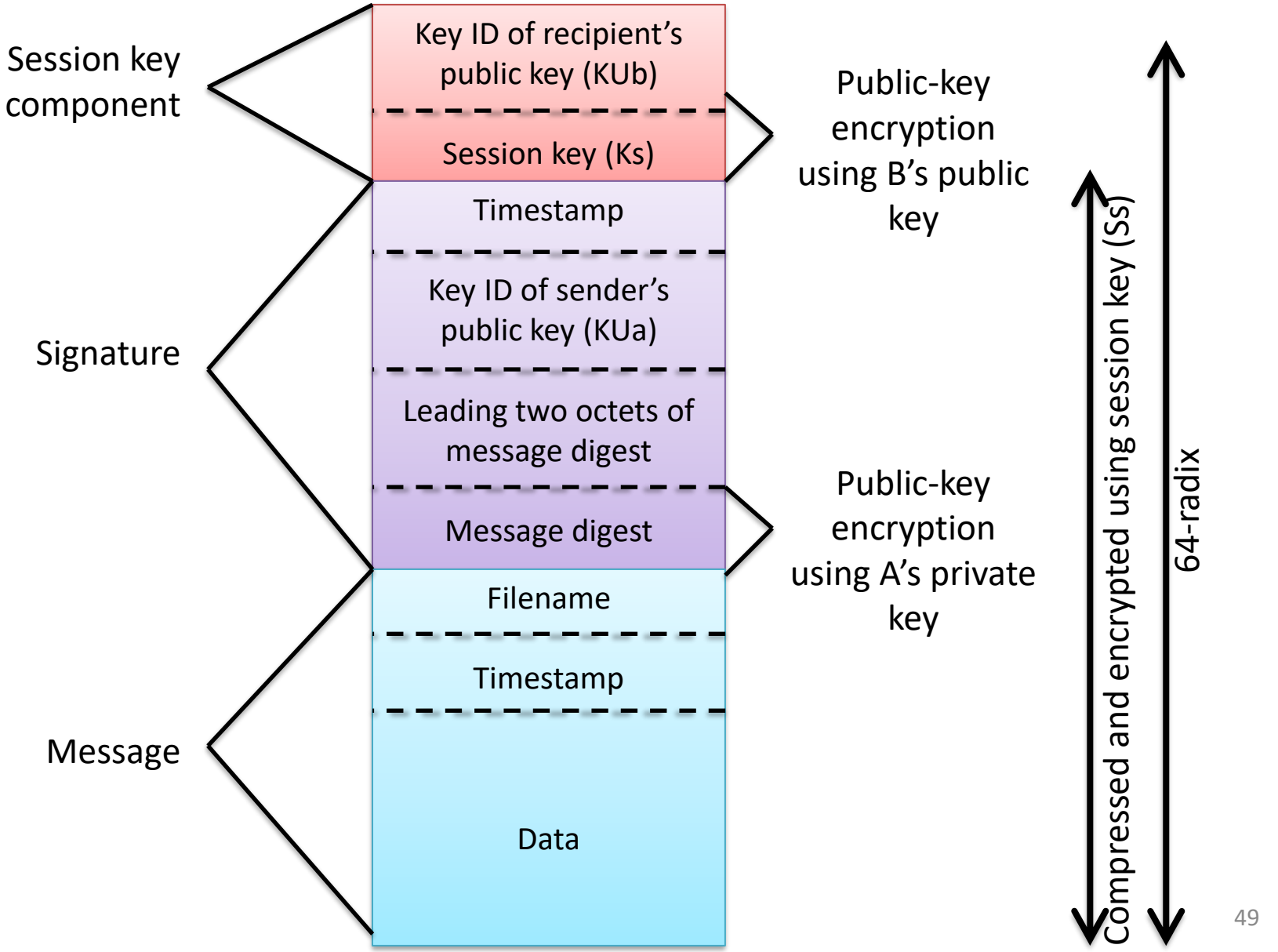
- Many electronic mail systems only permit the use of blocks consisting of ASCII text.

- PGP converts the raw 8-bit octets stream to a stream of printable ASCII characters using radix-64 expansion.

  Radix-64 algorithm
  - Maps 3 bytes to 4 printable chars
  - Also appends a CRC to detect transmission errors

- PGP also segments messages if too big.

General Format



PGP keys

- There are four types of keys
  - Pass-phrase key
  - Session keys (random keys generated)
  - Public-key
  - Private-key

- PGP allows to have more than one set of private-public keys per user.

Key management

- Because of the possibility of multiple public–private keys per user, the recipient of the message needs to know which of his/hers public key was used for encryption.
- One possibility is that the sender of the message includes the public key of the recipient, but it is unnecessarily wasteful of space.
- PGP send an identifier of the recipients public key.
- PGP assigns an ID to each public–key by using the least 64 significant bits of the key. (ID → KU_A mod(2^64))

Key rings

- One or more keys stored together constitute a key-ring.

- There are two classes:
  - **Private-key ring:** Stores the private/public key pairs owned at this node.
  - **Public-key ring:** Stores the public keys of other users known at this node.

Key rings

Using Passphrase key

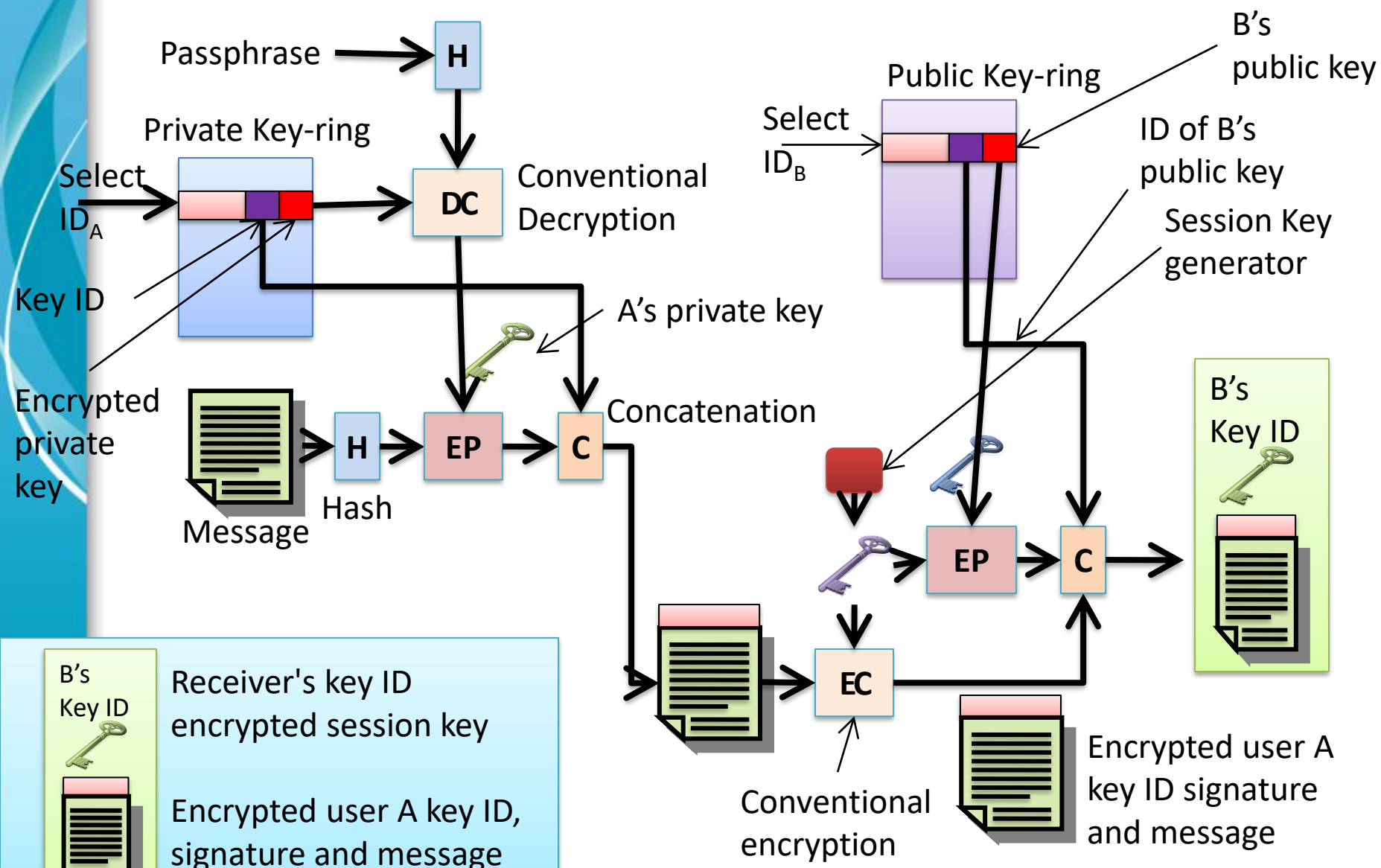
Private Key ring

Timestamp	Key ID	Public Key	Encrypted private key	User ID

Public Key ring

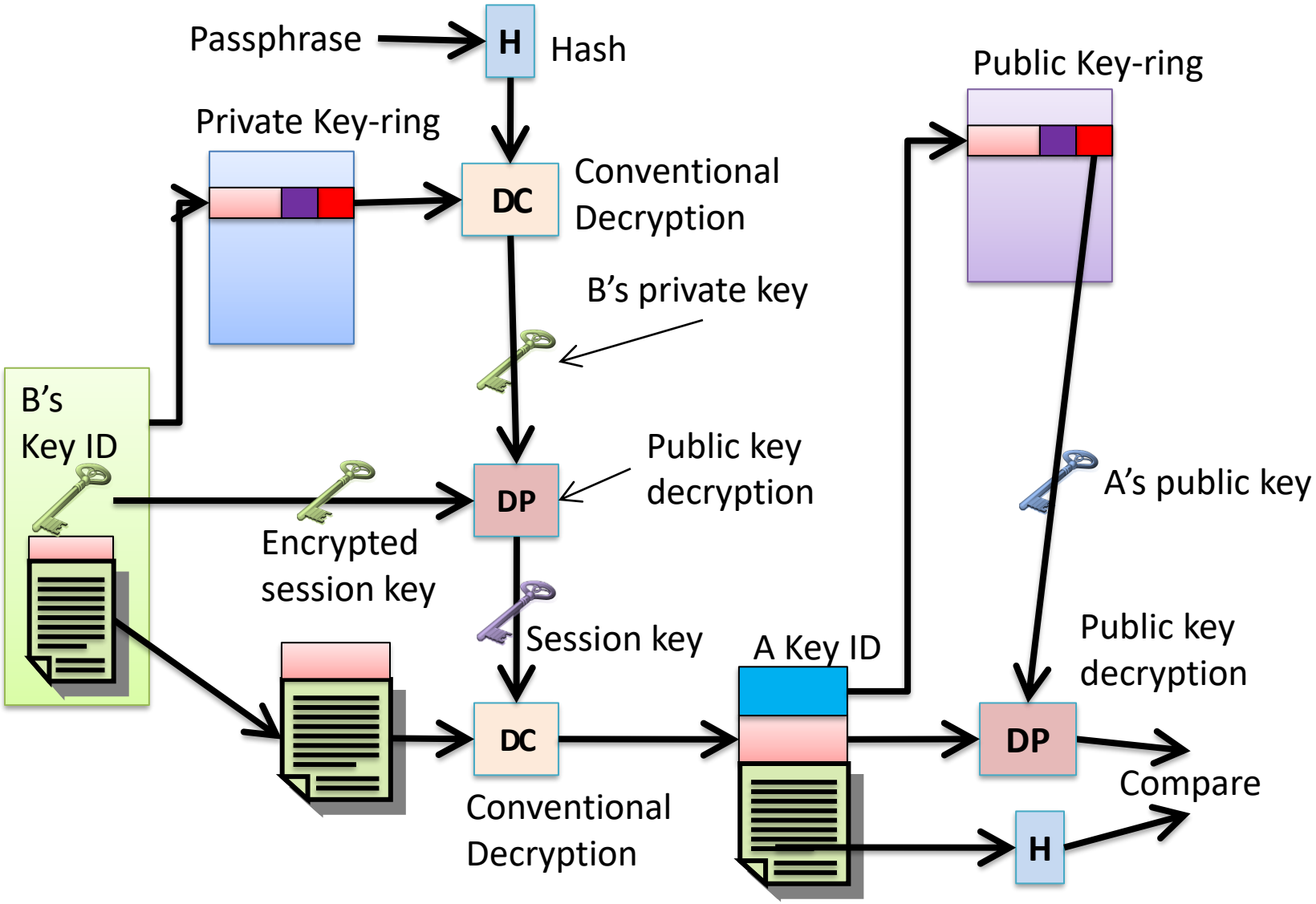
Timestamp	Key ID	Public Key	Owner Trust	User ID	Key legitimacy	Signature	Signature Trust

PGP message generation



PGP message generation from A to B

PGP message reception



PGP message reception from A to B

PGP Public-key management

- The problem: A's public-key ring contains a public key attributed to B, but how can A be sure the public key is from B, not someone else?
- The solution:
  - PGP does not include any specification for establishing certifying authorities
  - adopts a different trust model – the "web of trust"
  - Individuals sign one another's public keys (the "signature field" in the public-key ring) and create an interconnected community of public-key users.
  - PGP computes a *trust level* for each public key in key ring

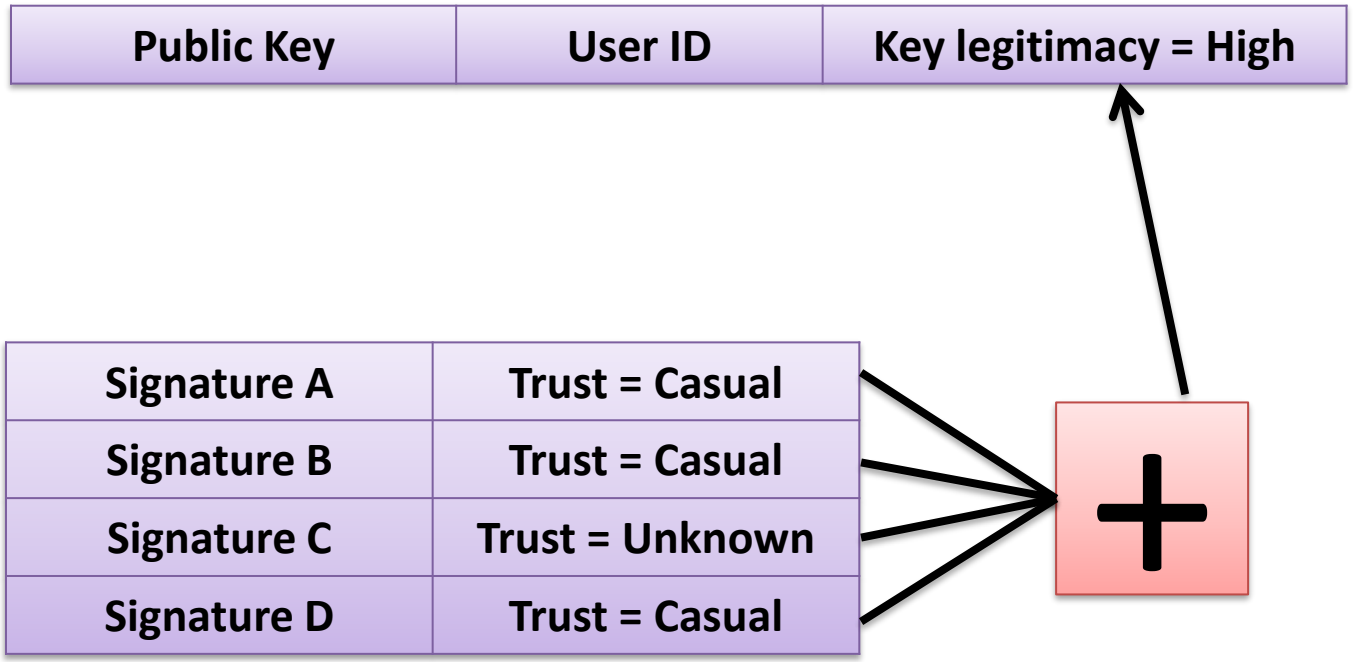
Trust

Timestamp	Key ID	Public Key	Owner Trust	User ID	Key legitimacy	Signature	Signature Trust

- **Key legitimacy** → indicates the extent to which PGP will trust that this is a valid public key for this user; the higher the level of trust, the stronger is the binding of this user ID to this key. This field is computed by PGP.
- **Owner Trust** → indicates the degree to which this public key is trusted to sign other public-key certificates; this level of trust is assigned by the user
- **Signature Trust** → indicates the degree to which this PGP user trusts the signer to certify public keys. The key legitimacy field is derived from the collection of signature trust fields in the entry.
- Example

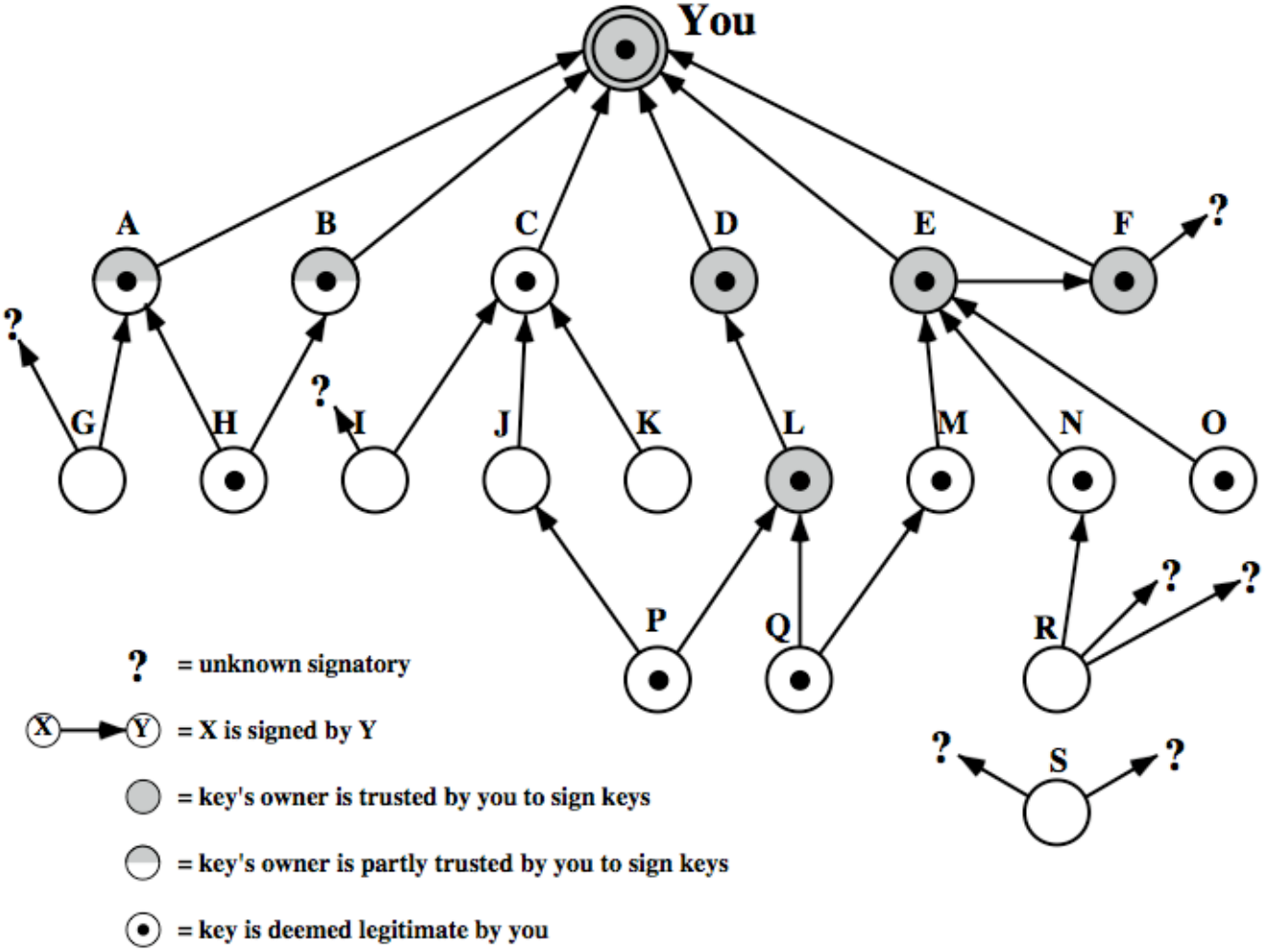
  UserID = politician
  - Owner Trust = low
  - Key legitimacy = high

Example of how PGP compute key legitimacy



PGP Trust Model Example

The structure of a public-key ring where the user has acquired a number of public keys, some directly from their owners and some from a third party such as a key server.



S/MIME Secure/Multipurpose Internet Mail Extension

- Security enhancement for the MIME (*Internet e–mail format standard*).
- MIME provides a convenient mechanism for transferring composite data.
- Binary data handled via base64 encoding.
- S/MIME: security related information sent as sections of a multipart message
  - multipart/signed
  - multipart/encrypted

RFC5322

- Defines a format for **text** messages that are sent using electronic mail
- A message consists of header lines (the header) followed by unrestricted text (the body).
- This is an example message:

```
Date: October 8, 2009 2:15:49 PM EDT
From: "William Stallings" <ws@shore.net>
Subject: The Syntax in RFC 5322
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.
```

Multipurpose Internet Mail Extensions (MIME)

- MIME is an extension to the RFC5322 framework
- MIME addresses some problems and limitations of the use of SMTP(Simple Mail Transfer Protocol) and RFC5322, e.g., cannot transmit executable files or other binary objects.
- The MIME specification includes the following elements:
  - Five new message header fields are defined (The **Content-Type** header field is used for secure communications)
  - A number of content formats are defined, which support multimedia electronic mail

MIME formats

- This is an example of **multipart type** email. The headers of a simple email in MIME looks like this (Taken from RFC 2046):

```
Date: Wed 09 Dec 2009 10:37:17 (GMT)
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
   handy place for email composers to include an
   explanatory note to non-MIME conformant readers.
-simple boundary
This is implicitly typed plan ASCII text. It does NOT end
   with a line break.
-simple boundary
Content-type: text/plain; charset=us-ascii
This is explicitly typed plain ASCII text. It DOES end
   with line break.
-simple boundary-
This is the epilogue. It is also to be ignored.
```

Secure/Multipurpose Internet Mail Extension (S/MIME)

- A security enhancement to the MIME Internet e-mail format standard based on technology from RSA Data Security

Summary of S/MIME services

Function	Typical Algorithm
Digital signature	RSA/SHA-256
Message encryption	AES-128 with CBC
Compression	unspecified
E-mail compatibility	Radix-64 conversion

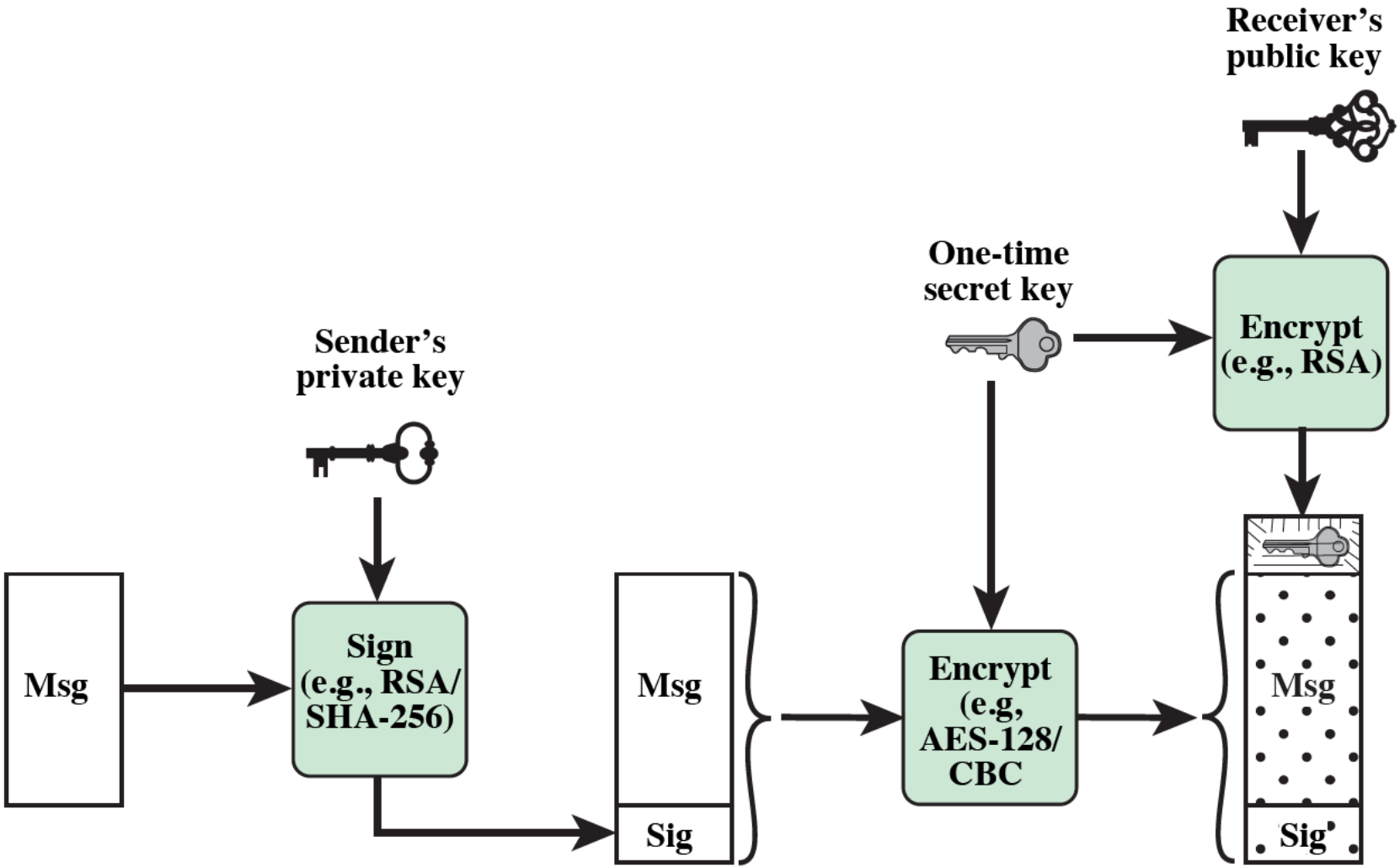
S/MIME Authentication

1. The sender creates a message
2. SHA-256 is used to generate a 256-bit message digest of the message
3. The message digest is encrypted with RSA using the sender's private key, and the result is appended to the message. Also appended is identifying information for the signer, which will enable the receiver to retrieve the signer's public key
4. The receiver then verifies the message digest.

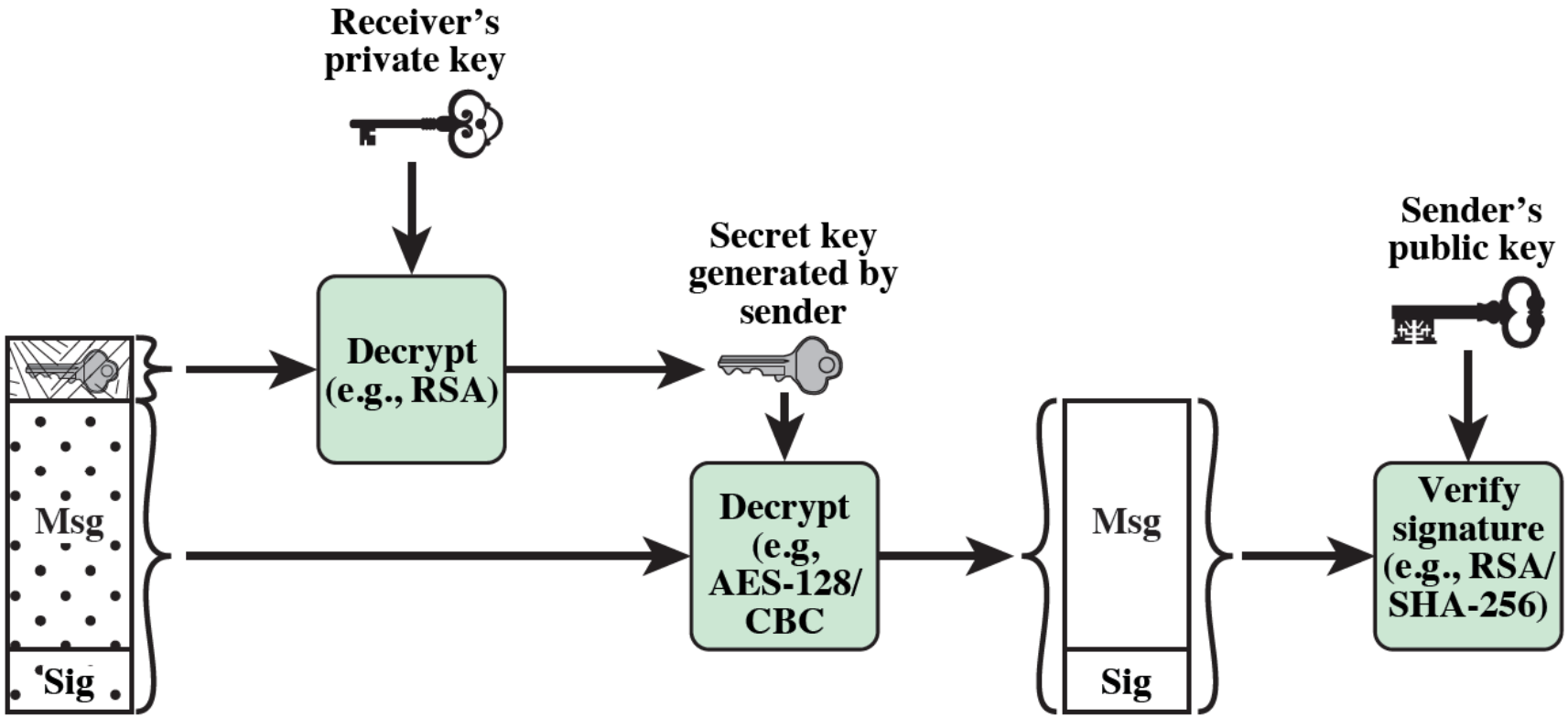
S/MIME Confidentiality

1. The sender generates a message and a random 128-bit number to be used as a content-encryption key for this message only.
2. The message is encrypted using the content-encryption key.
3. The content-encryption key is encrypted with RSA using the recipient's public key and is attached to the message.
4. The receiver uses RSA with its private key to decrypt and recover the content-encryption key.
5. The content-encryption key is used to decrypt the message.

Confidentiality and Authentication



Confidentiality and Authentication



S/MIME Cryptographic Algorithms

Function	Requirement
Create a message digest to be used in formatting a digital signature	• **MUST** support SHA-256. •**ShOULD** support SHA-1 • Receiver **SHOULD** support MD5 for backward compatibility.
Encrypt message digest to form digital signature	•**MUST** support RSA with SHA-256. •**SHOULD** support: DSA with SHA-256, RSASSA-PSS with SHA256, RSA with SHA-1, DSA with SHA-1, RSA with MD5
Encrypt session key for transmission with message	•**MUST** support RSA encryption (key size 512-1024 bits). •**SHOULD** support RSAES-OAEP, Diffie-Hellman ephemeral-static mode.
Encrypt message for transmission with a one-time session key	•**MUST** support AES-128 with CBC. •**SHOULD** support AES-192 CBC and AES-256 CBC, Triple DES CBC.

MIME-based Security

- The general format is

...
Content-Type: multipart/type; boundary ="Boundary"
Content-Transfer-Encoding: base64
--Boundary
’’ Encryption info ’’
--Boundary
’’ Message ’’
--Boundary
’’Signature’’
--Boundary--

S/MIME messages

- S/MIME secures a MIME entity with a signature, encryption, or both.
- A MIME entity may be an entire message (except for the
- RFC 5322 headers), or if the MIME content type is multipart, then a MIME entity is one or more of the subparts of the message.

- The MIME entity is prepared according to the normal rules for MIME message preparation
  - The MIME entity plus some security-related data, such as algorithm identifiers and certificates, are processed by S/MIME to produce what is known as a PKCS object
  - A PKCS object is then treated as message content and wrapped in MIME

(PKCS is a set of public-key cryptography specification issued by RSA Lab.)

S/MIME content types

- S/MIME also provides the following content types:
  - Enveloped data (Encrypted content and associated keys)
  - Signed data (Encoded message + signed digest)
  - Clear–signed data (Cleartext message + encoded signed digest)
  - Signed & Enveloped data (Nesting of signed & encrypted entities)

SignedData

Content-Type: application/pkcs-7; smime-type=signed-data;
    name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m
ruqweoirlJASkjqwpie8038aksdjf

SignedData

- Select algorithm (SHA or MD5).
- Compute the message digest (Hash) to be signed.
- Encrypt the digest using signer’s private key.

SignedData

- Prepare a block known as SignerInfo that contains
  - Signer's public key certificate
  - ID of the message digest algorithm
  - ID of the algorithm used to encrypt the digest
  - Encrypted digest

Clear Signed

- **The message is sent “in the clear”**

Content-Type: multipart/signed;
    protocol=’’application/pkcs7-signature’’;
    micalg=sha1; boundary=boundary42
--boundary42
Content-Type: text/plain
Hello there
--boundary42
Content-Type: application/pkcs7-signature; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7s
ruqweoirlJASkjqwpie8038aksdjf
--boundary42--

EnvelopedData

Content-Type: application/pkcs7-mime;
    smime-type=enveloped-data; name=smime.p7m
Content-Transfer-Encoding: base64
Content-Disposition: attachment; filename=smime.p7m

ruqweoirlJASkjqwpie8038aksdjf

EnvelopedData

- Generate the session key for the conventional encryption (RC2/40, TDES).
- Encrypt the session key with the recipient's public key.

EnvelopedData

- Prepare a block known as RecipientInfo that contains:
  - Sender's public certificate
  - ID of the encryption algorithm
  - Encrypted session key
- Encrypt the message with the session key.

S/MIME Certificates processing

- Uses public-key certificates that conform to X.509 v3.
- Key management is hybrid between X.509 and PGP's web of trust.
- Each client has a list of trusted CA's certificates and own public/private key pairs & certificates.
- Certificates must be signed by trusted CAs (e.g. VeriSign)

S/MIME Problems

- Earlier versions used mostly crippled crypto.
- S/MIME cracking screen saver released 1997.
- Original S/MIME based on patented RSA and proprietary RC2 (rejected as a standard).
- S/MIME v3 uses strong crypto and non-patented, non-proprietary technology.

Summary

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

Threats to Security:

Intruders
&
Malicious Software

Intruders

- Hostile or unwanted trespass by users or software.

- Intruder types:

  - **Masquerader:** An unauthorised user who logs in, by exploiting a legitimate user's account.

  - **Misfeasor:** An authorised user who misuse their privileges (to perform actions beyond their limits).

  - **Clandestine user:** A user who seizes supervisory control of the system and uses this to suppress audit collection.

Examples of Intrusion

- Remote root compromise of email server
- Web server defacement
- Guessing / cracking passwords
- Copying viewing sensitive data / databases
- Running a packet sniffer
- Distributing pirated software
- Impersonating a user to reset password
- Using an unattended workstation without permission

Intrusion techniques

- **Secure access using passwords:**
  - **One-way function (not reversible):** Stores only the value of a function based on the password. The system transforms the entered password and compares it with the stored one.
  - **Access control:** Access to password file is limited to one or a very few accounts.

- **Typical attacks on the above approach include:**
  - Default system passwords.
  - Trial of all short passwords.
  - Trial of system's online dictionary.
  - Collection of info (full names, family, pets, office pictures, etc).
  - Trial of legitimate license plate numbers in the city.
  - Use Trojan horse to bypass restrictions on access.
  - Tap the line between the user and the remote host.
  - Other (e.g. Phone numbers, ID numbers, addresses, etc).

Intrusion detection

- **Audit records**
  - **Native audit records:** Almost all multiuser operating systems include accounting software that collect information on user activity.
    - Advantages: No additional collection software is needed.
    - Disadvantages: Records may not contain the needed info.
  - **Detection-specific records:** A collection facility that generates audit records containing only the information required by the intrusion detection system.
    - Advantages: Vendor independent (reported to variety of systems)
    - Disadvantages: Extra overhead in having two accounting packages.

Intrusion detection

- **Audit records example**
  - Subject
  - Action
  - Object
  - Exception-Condition
  - Resource-Usage
  - Time-Stamp

Smith	Execute	<Library> COPY.EXE	0	CPU = 00002	11058721678
Smith	Read	<Smith> GAME.EXE	0	RECORDS = 0	11058721679
Smith	Execute	<Library> COPY.EXE	Write-viol	RECORDS = 0	11058721680

Intrusion detection

- **Statistical anomaly detection**
  - Uses statistical tests to observe and determine high level of confidence (tests are applied on the collected data relating to the behaviour of legitimate user over a period of time).
  - **Threshold detection:** Defines the thresholds for the frequency of events occurrences, independent of the user.
  - **Profile based detection:** A profile of the behaviour of users is built and then used to detect changes in the behaviour of the account activity.

Intrusion detection

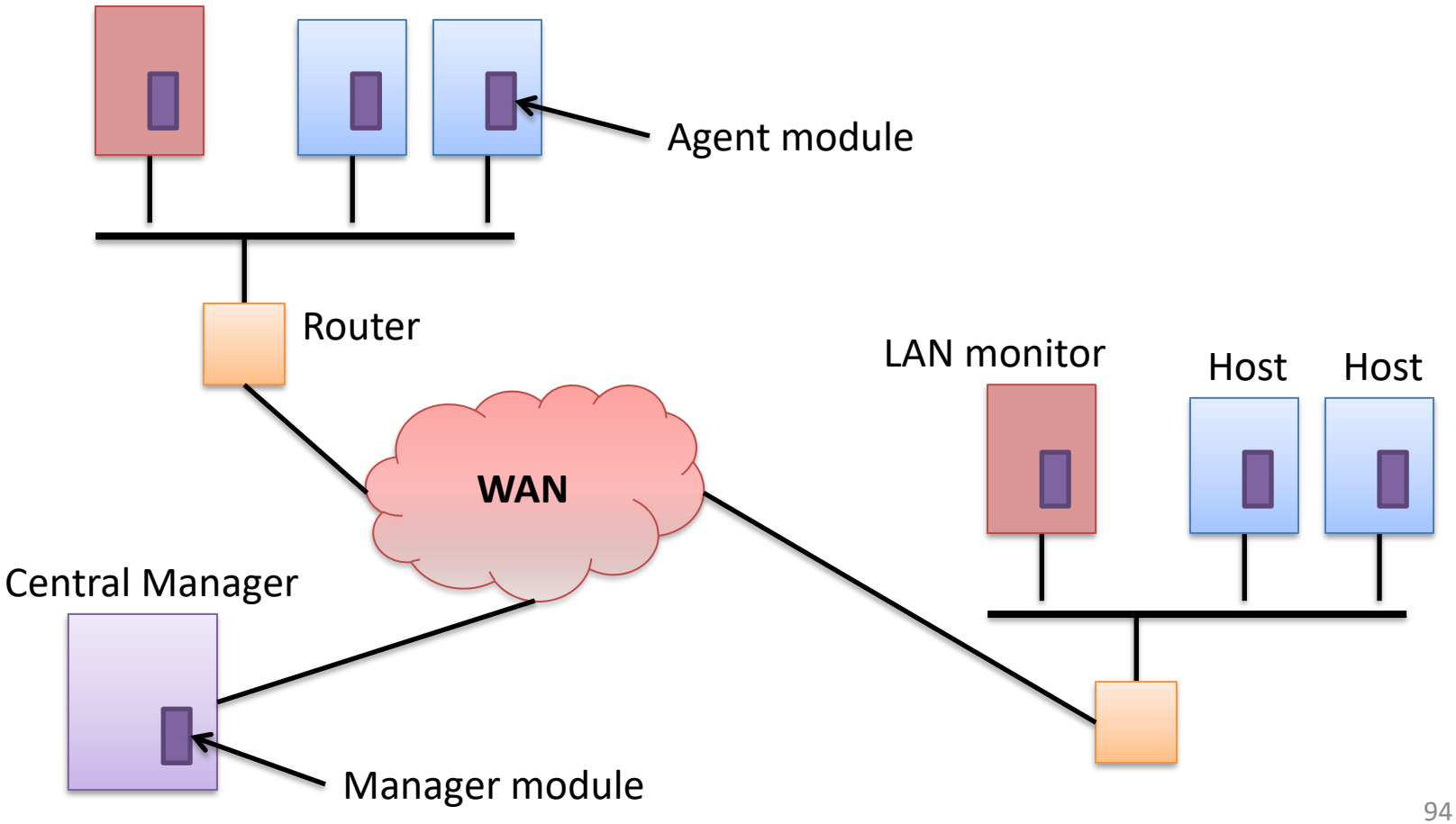
- **Rule-based detection**
  - Defines a set of rules that can be used to decide that a given behaviour is that of an intruder.
  - **Anomaly detection:** Detection of deviation from previous usage patterns is derived from certain developed rules.
  - **Penetration identification:** An expert system approach that searches for suspicious behaviour.

Intrusion detection

- **Distributed intrusion detection**
  - To deal with different audit record formats.
  - On the network, one or more nodes in the network will serve as collection and analysis points for the data and the system.
  - Either centralised or decentralised architecture can be used.

Intrusion detection

- **Distributed intrusion detection**



Intrusion detection

- **Honeypots (Decoy systems to lure attackers)**
  - Divert an attacker from accessing critical systems.
  - Collect information about the attacker's activity.
  - Encourage the attacker to stay on the system long enough for administrator to respond.

Malicious Software

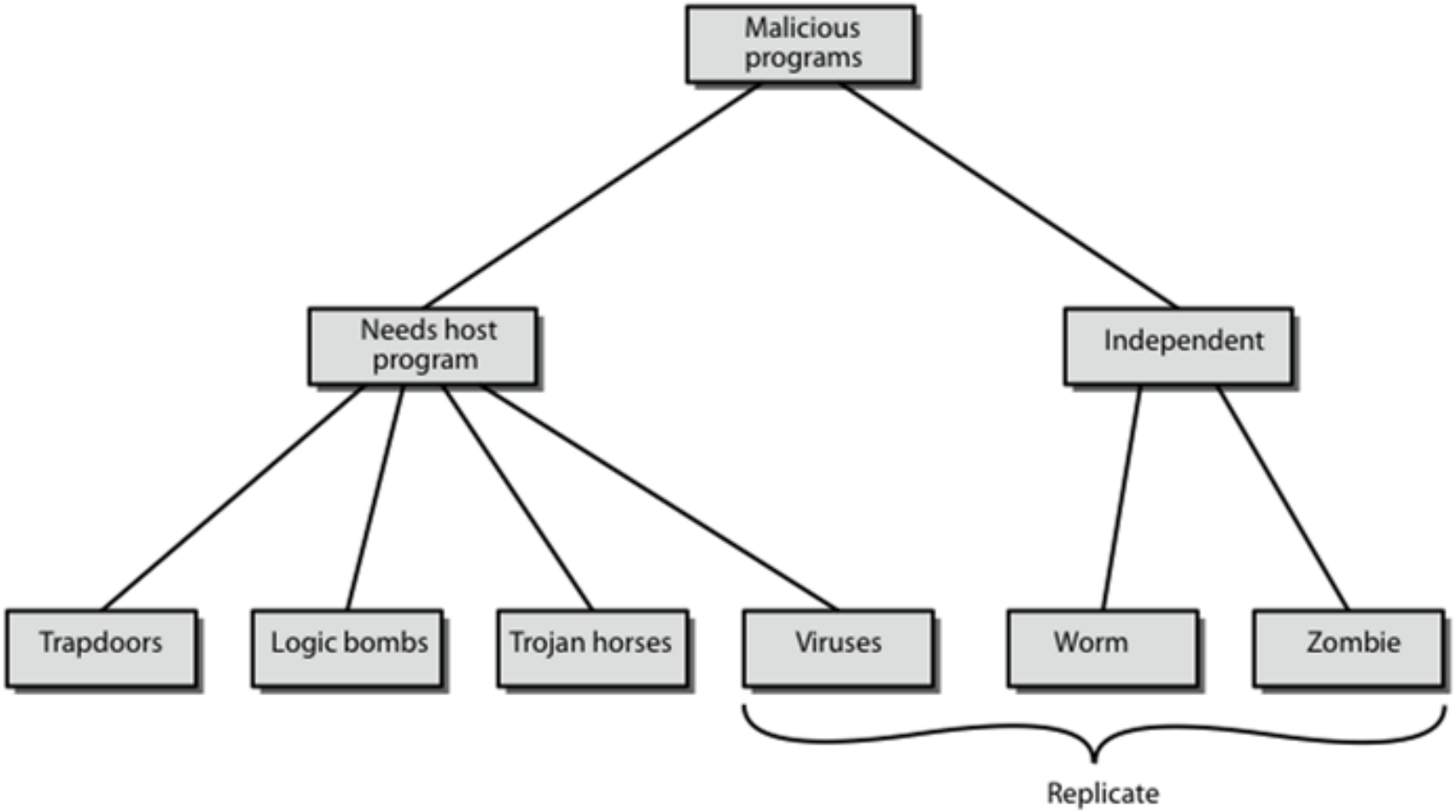
Malicious Software - Terminology

Term	Description
Virus	Malware that, when executed, tries to replicate itself into other executable code; when it succeeds the code is said to be infected. When the infected code is executed, the virus also executes.
Worm	A computer program that can run independently and can propagate a complete working version of itself onto other hosts on a network.
Trojan horse	A computer program that appears to have a useful function, but also has a hidden and potentially malicious function that evades security mechanisms, sometimes by exploiting legitimate authorizations of a system entity that invokes the Trojan horse program.
Backdoor (trapdoor)	Any mechanism that bypasses a normal security check; it may allow unauthorized access to functionality.

Malicious Software - Terminology

Term	Description
Downloaders	Program that installs other items on a machine that is under attack. Usually, a downloader is sent in an e-mail.
Auto-rooter	Malicious tools used to remotely break into new computers.
Spammer programs	Used to send large volumes of unwanted e-mail.
Flooders	Used to attack networked computer systems with a large volume of traffic to carry out a denial-of-service (DoS) attack.
Zombie, bot	Program activated on an infected machine that is activated to launch attacks on other machines.
Spyware	Software that collects information from a computer and transmits it to another system.
Adware	Advertising that is integrated into software. It can result in pop-up ads or redirection of a browser to a commercial site.

Malicious Software



Distributed DoS Attacks

Distributed Denial of Service Attacks

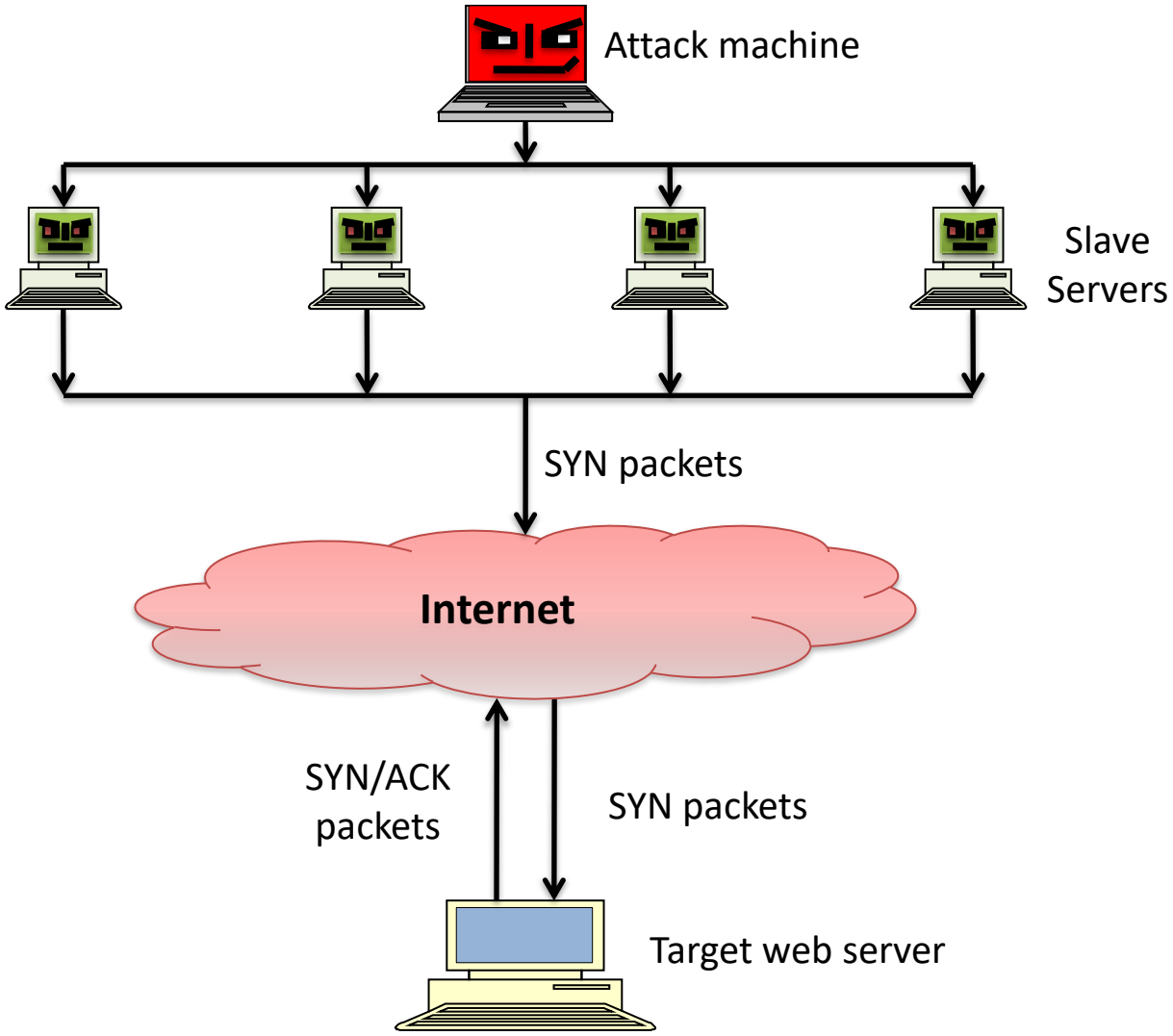
- A denial of service (**DoS**) attack is an attempt to prevent legitimate users of a service from using that service. When this attack comes from a single host or network node, then it is simply referred to as a DoS attack.

- DDoS Attack attempts to consume the target's resources so that it cannot provide services. In a DDoS attack, an attacker is able to recruit a number of hosts throughout the Internet to simultaneously or in a coordinated fashion launch an attack upon the target.

- Simple attack examples
  - Internal Resource Attack (SYN or ICMP)
  - Direct or Reflector flooding attack

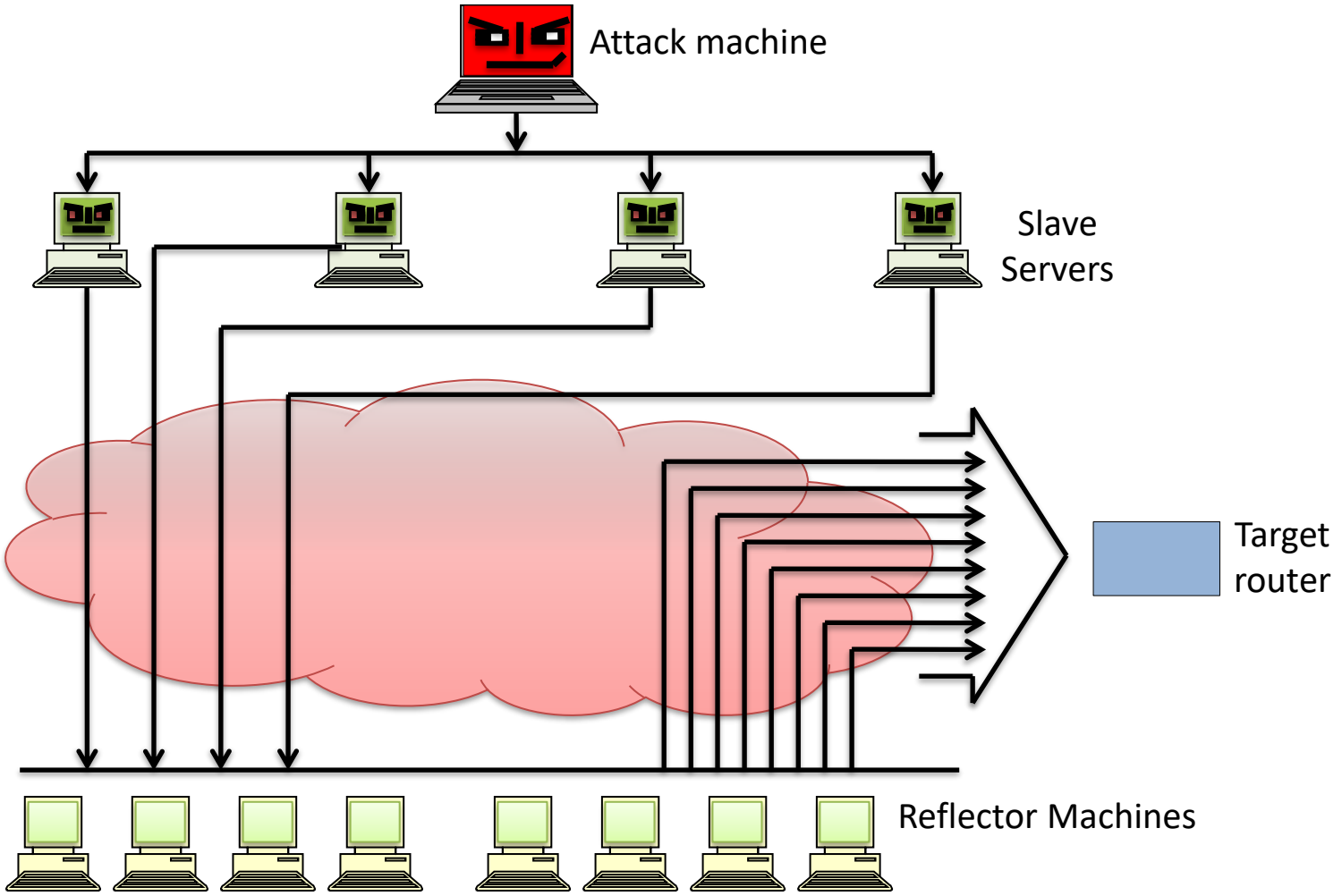
Distributed Denial of Service Attacks

- Distributed SYN flood attack



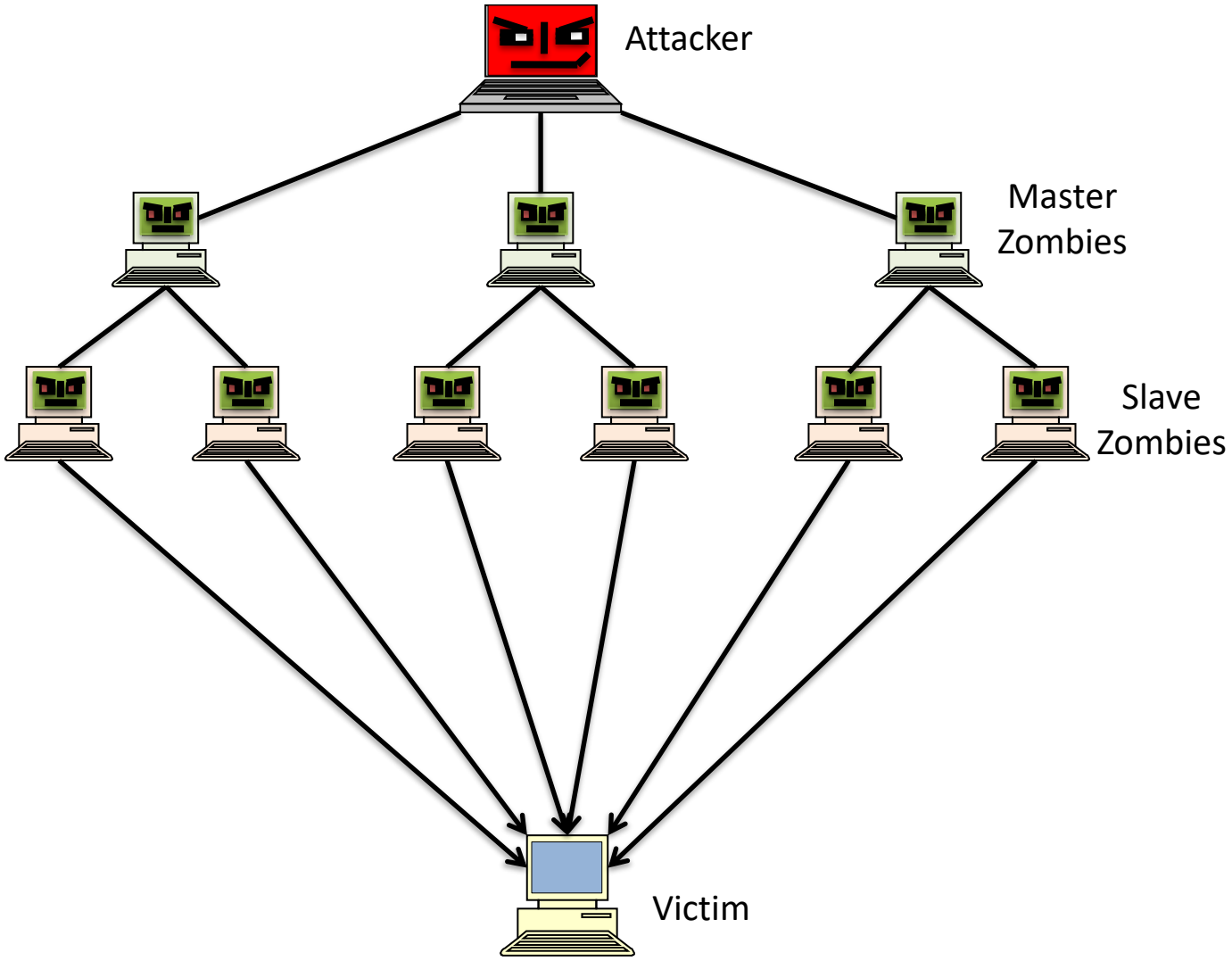
Distributed Denial of Service Attacks

- Distributed ICMP attack



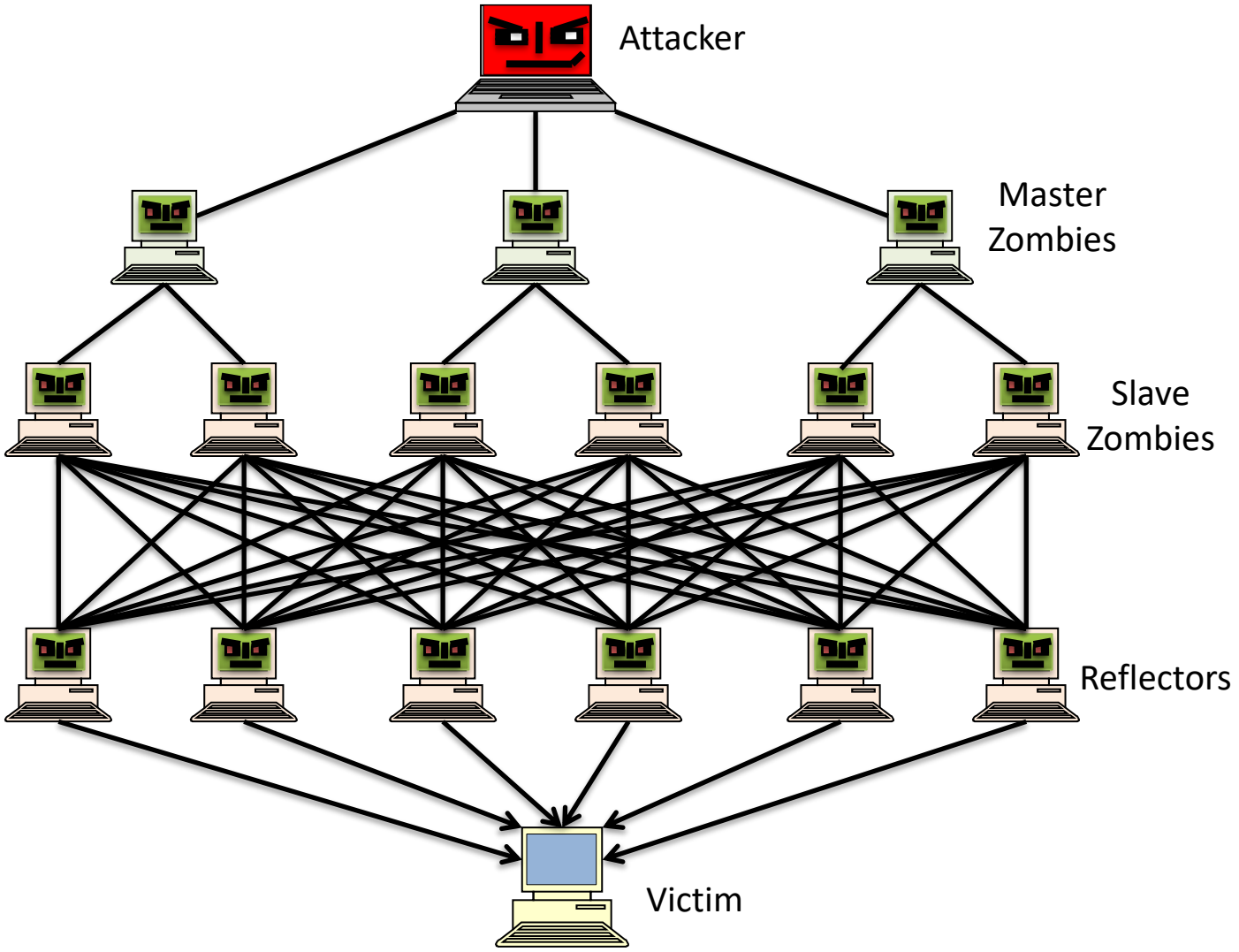
Distributed Denial of Service Attacks

- Direct DDoS attack (flooding based)



Distributed Denial of Service Attacks

- Reflector DDoS Attack (flooding based)



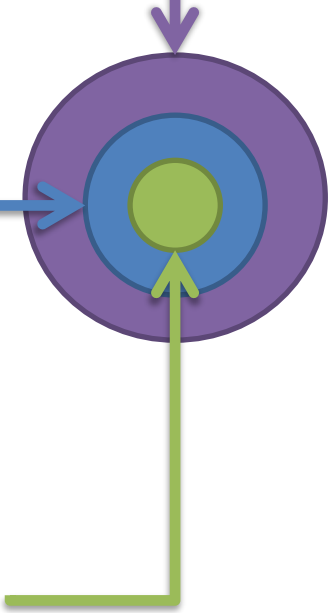
DDoS Countermeasures

- **Attack prevention and pre-emption** (before the attack)
- **Attack detection and filtering** (during the attack)
- **Attack source trace-back and identification** (during and after the attack)

The big picture..

Information Systems Security

- **Informal**
  - Educating and training the members of organisation
- **Formal**
  - Data management or security rules
  - Management of personnel
- **Technology Based (Technical):**
  - Smart security cards, Ciphers, etc.



Review..