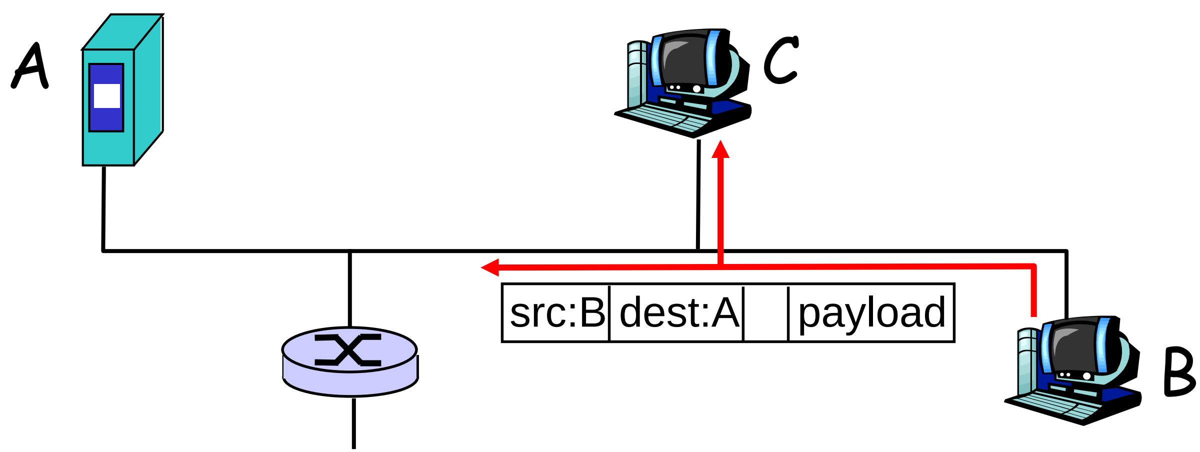
Chapter 8
Network Security

Internet security threats

Packet sniffing:

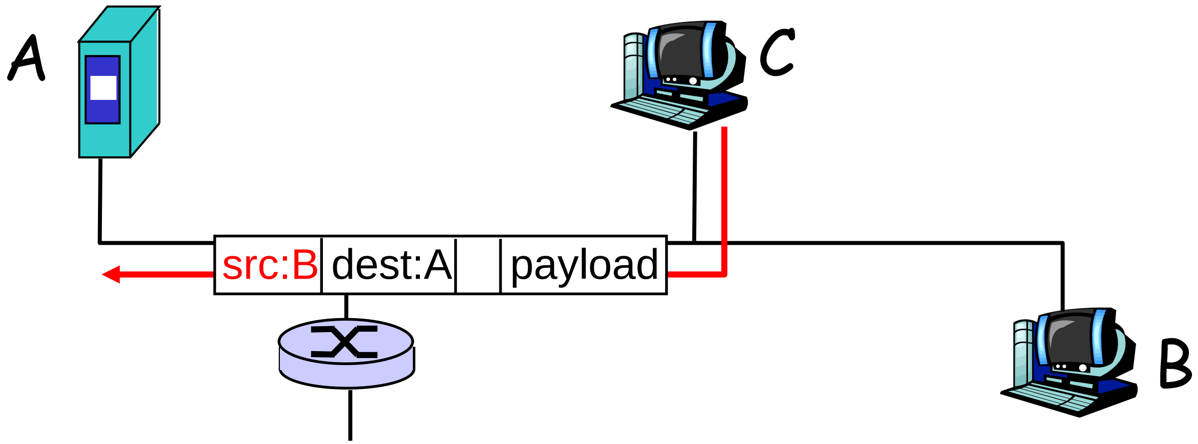
- broadcast media
- promiscuous NIC reads all packets passing by
- can read all unencrypted data (e.g. passwords)
- e.g.: C sniffs B's packets



Internet security threats

IP Spoofing:

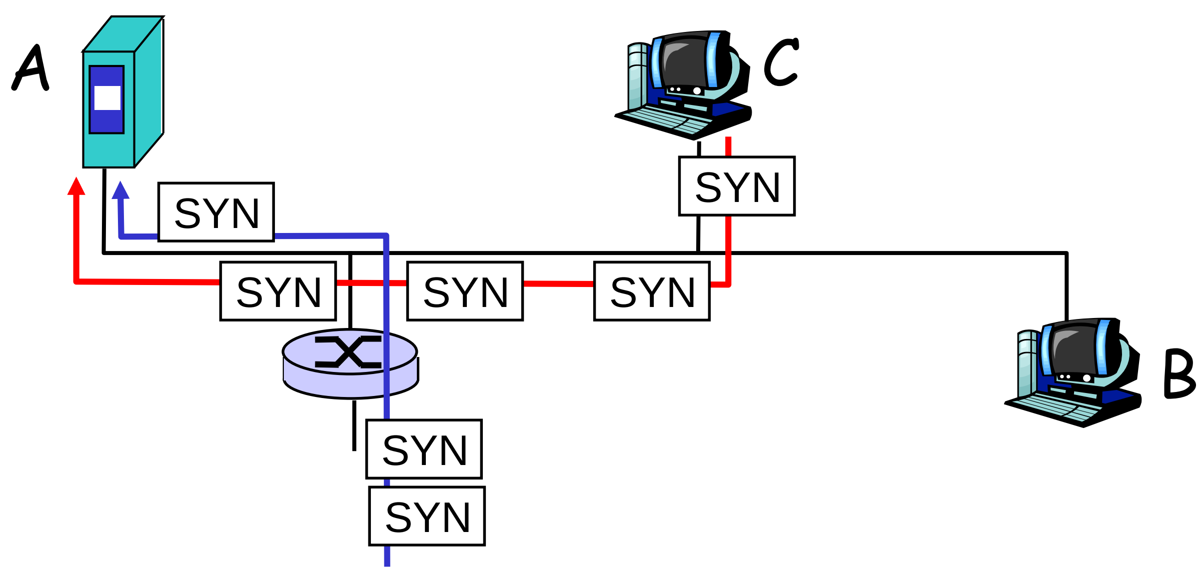
- can generate "raw" IP packets directly from application, putting any value into IP source address field
- receiver can't tell if source is spoofed
- e.g.: C pretends to be B



Internet security threats

Denial of service (DOS):

- flood of maliciously generated packets "swamp" receiver
- Distributed DOS (DDOS): multiple coordinated sources swamp receiver
- e.g., C and remote host SYN-attack A



What is network security?

Confidentiality (Secrecy, Privacy) : only sender, intended receiver should "understand" msg contents

- sender encrypts msg
- receiver decrypts msg

Authentication: sender, receiver want to confirm identity of each other

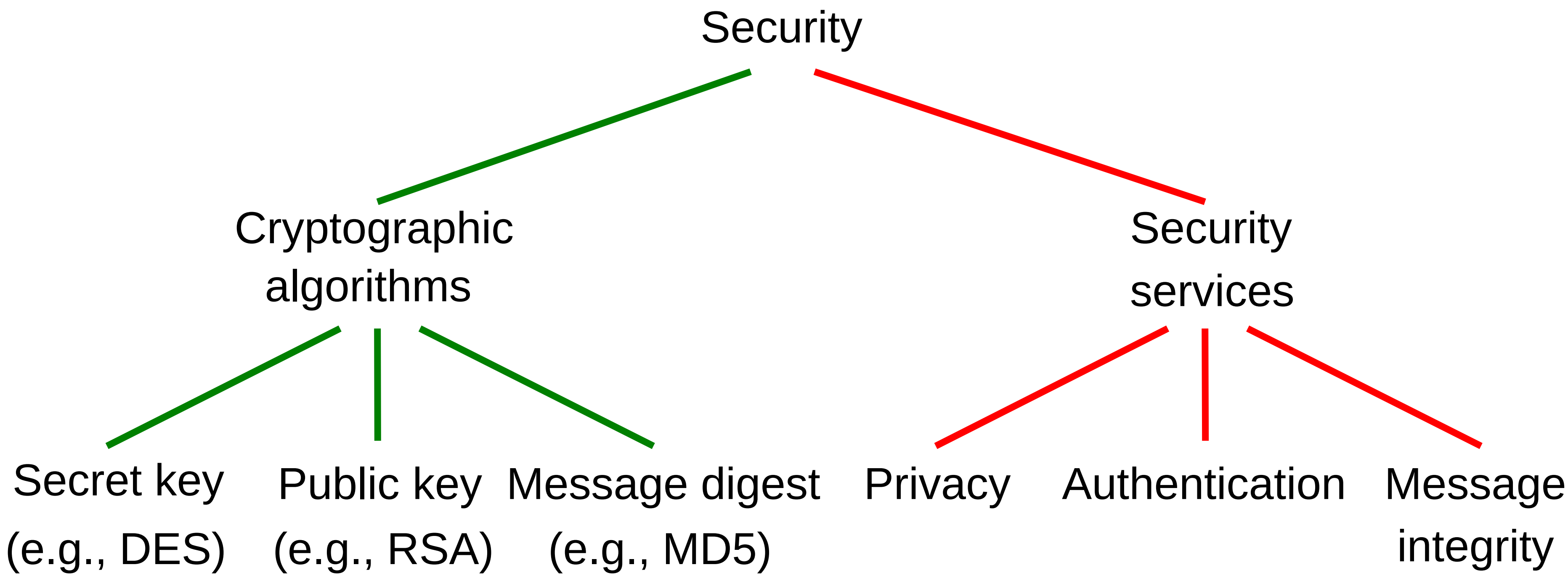
Message Integrity: sender, receiver want to ensure message not altered (in transit, or afterwards) without detection

Network Security

- Sometimes the data transmitted between application processes is **confidential**, such as **credit card numbers**
- The idea of **encryption** is that
  - The sender applies an encryption function to the original **plaintext message**
  - The resulting **ciphertext message** is sent over the network
  - The receiver applies a reverse function (**decryption**) to recover the original plaintext
- The **encryption/decryption process** generally depends on a **secret key** shared between the sender and the receiver
- It can also be used to support **authentication** (verifying the identity of the remote participant) and **integrity** (making sure that the message has not been altered)

Security Issues

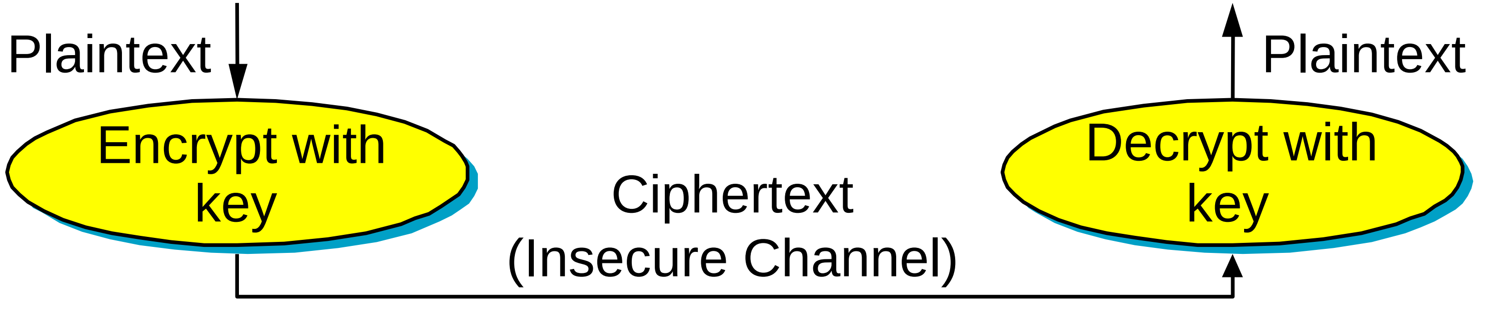
- The security issue can be divided into two parts
  - **Cryptographic algorithms**
  - **Security services**



Cryptographic Tools

Principles of Ciphers

- The **sender** applies an **encryption function** to the original **plaintext** message, resulting in a **ciphertext** message that is sent over the network
- The receiver applies a **decryption function** – the **inverse** of the encryption function – to recover the original plaintext
- Cryptographers have been led to the principle that encryption and decryption functions should be parameterized by a **key**
  - The functions should be considered **public knowledge**
  - Only the key need be **secret**



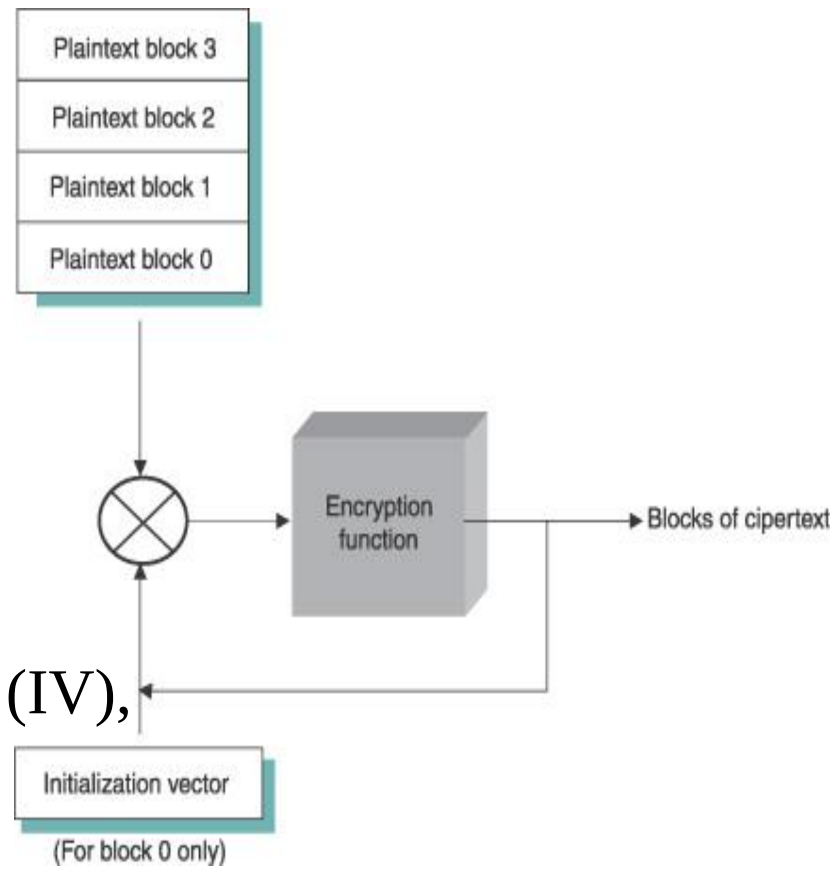
- F(plaintext, key)=ciphertext
- G(ciphertext, key)=plaintext
- Simple example:
- F: matrix multiplication
- G: matrix inversion
- Key: the matrix A
- Can be deciphered if an enough number of linearly independent vectors are known.

$$Y = AX$$

$$X = A^{-1}Y$$

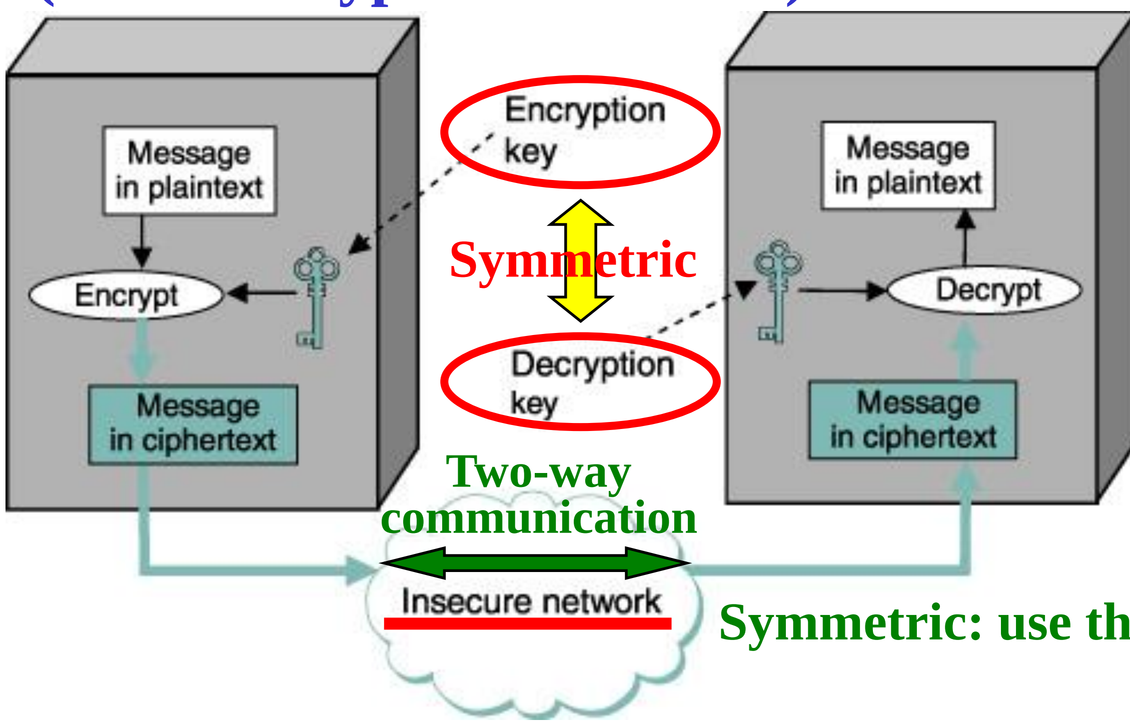
Principles of Ciphers

- Most ciphers are **block ciphers**
  - Take a plaintext block of a **fixed size**
  - The same plaintext block will always result in the same ciphtext block
- To prevent this problem
  - Each plaintext blocks is **XORed** with the previous block's ciphertext before being encrypted
  - Use an **initialization vector** (IV), which is a random number



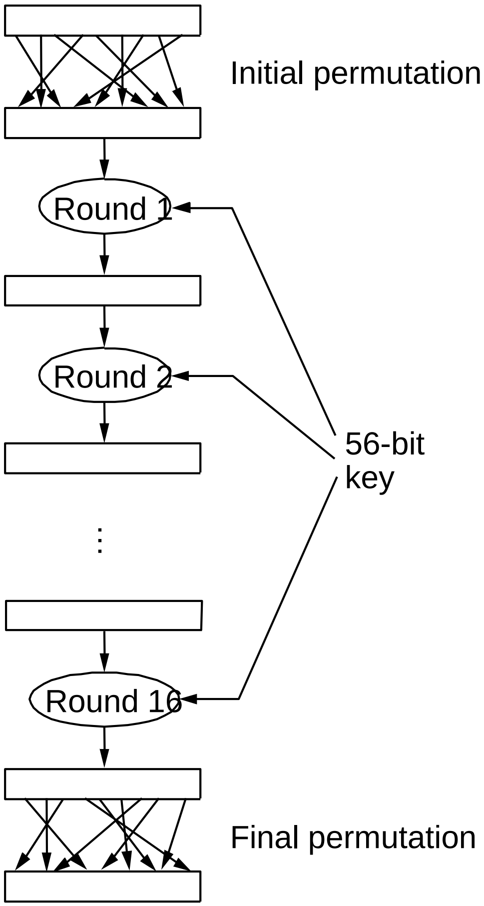
Symmetric-key Ciphers

- In a symmetric-key cipher, both participants share **the same key**
- Symmetric-key ciphers are also known as secret-key ciphers
  - The shared key must be known only to the participants
  - **DES (Data Encryption Standard)**

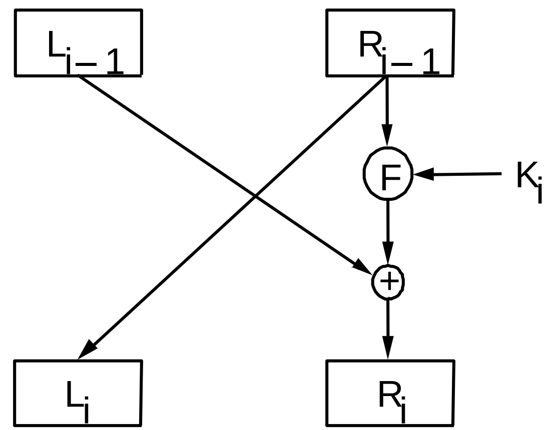


Symmetric: use the same key

- 64-bit key (56-bits + 8-bit parity)
- 16 rounds



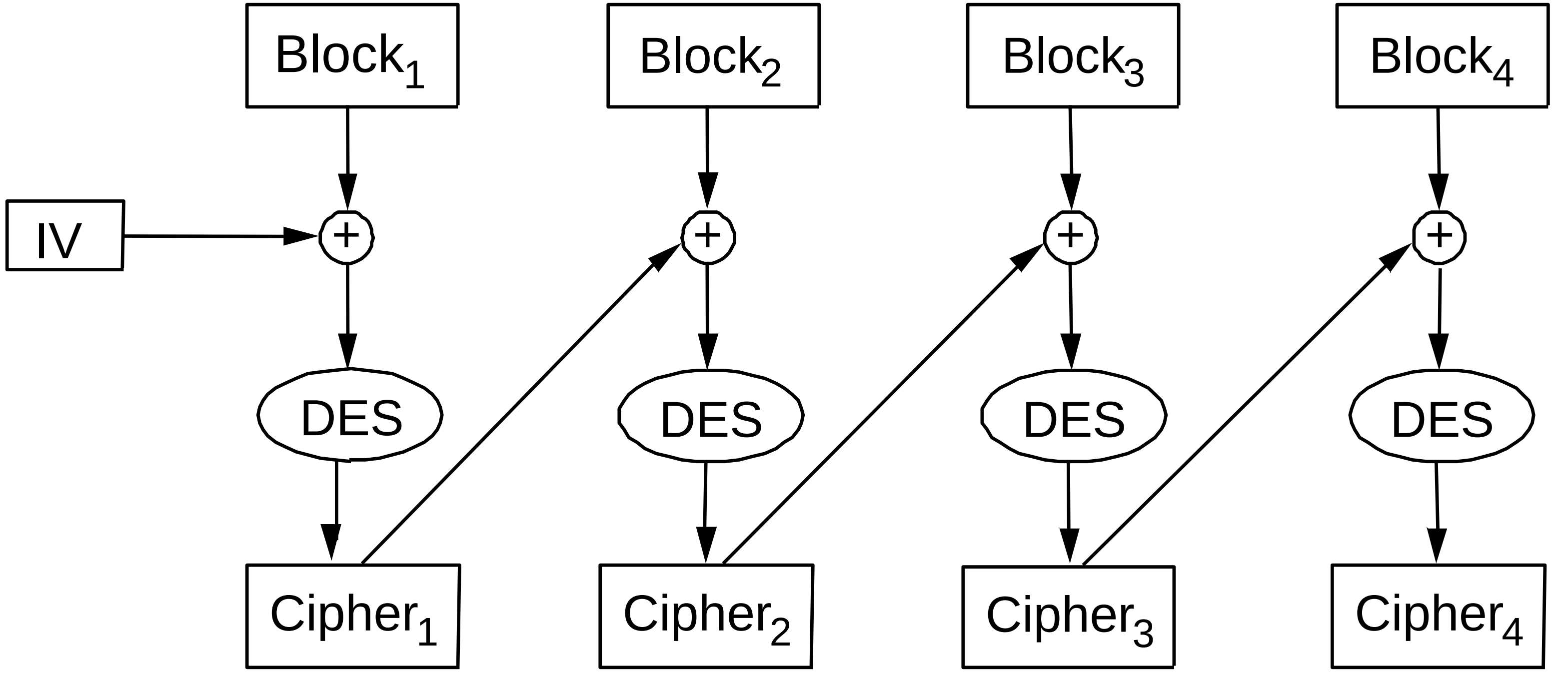
- Each Round



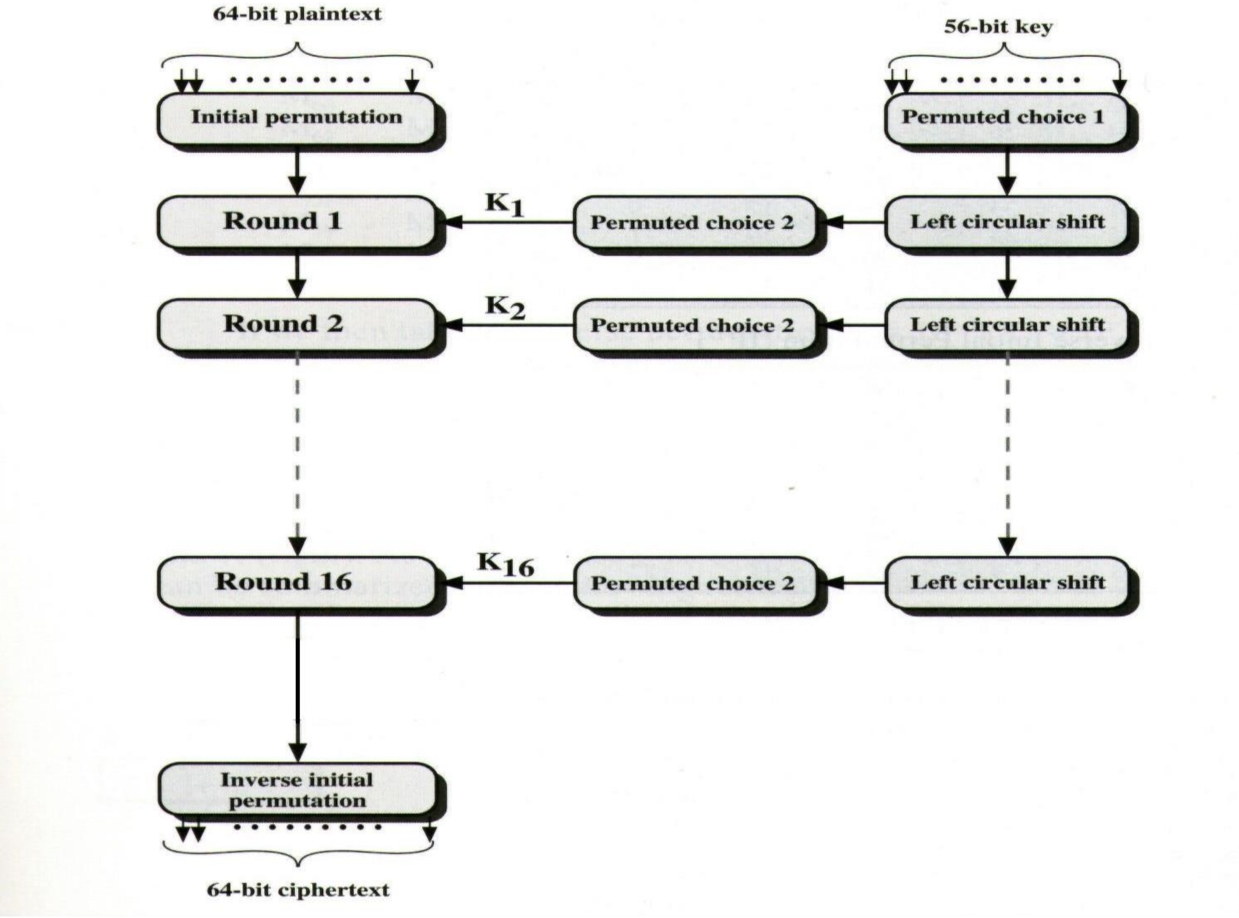
$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \oplus F\left(R_{i-1}, K_i\right)$$

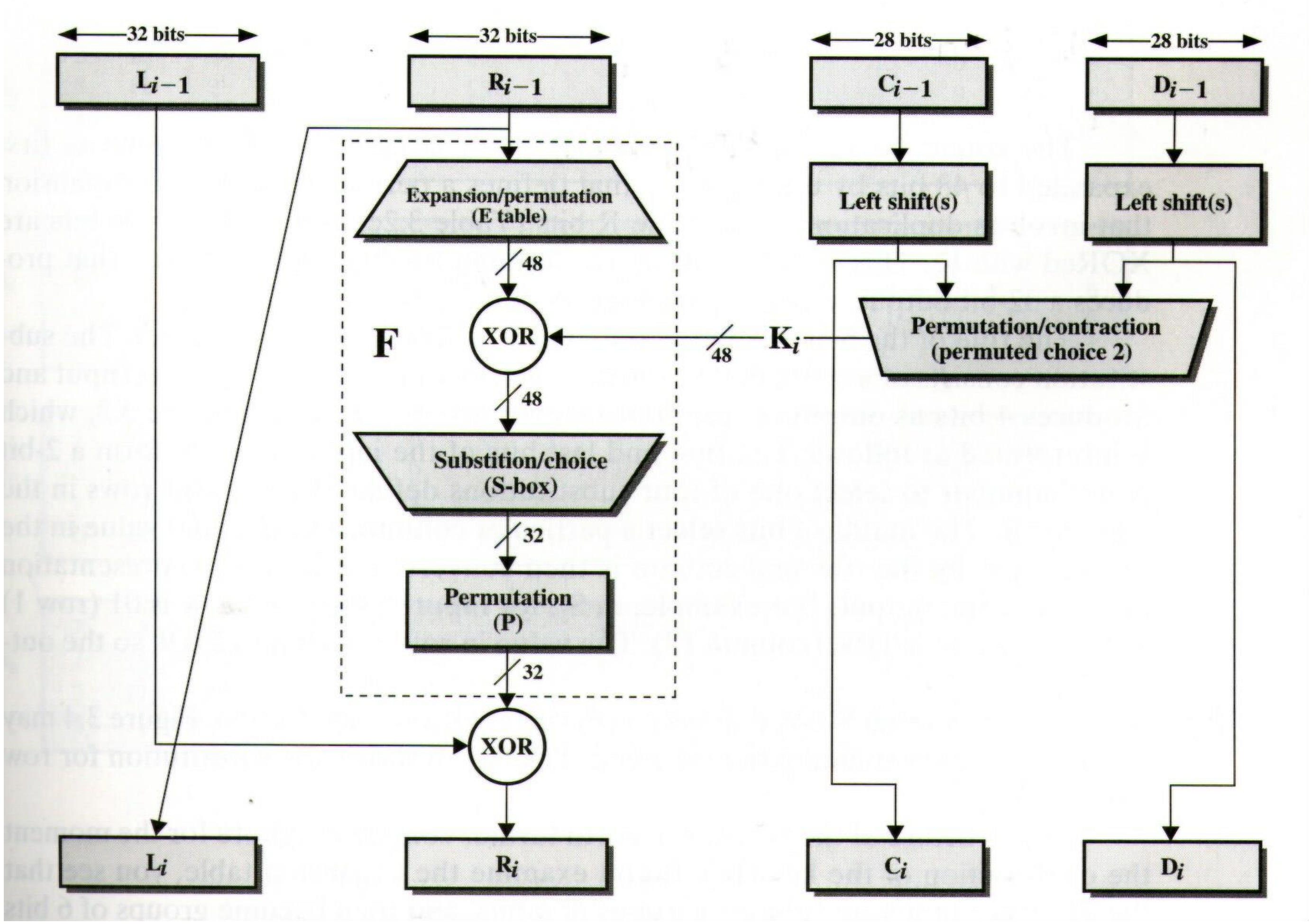
- Repeat for larger messages



General Depiction of DES Encryption Algorithm



Detail of Single Round



Detail of Single Round (cont.)

- L_i is the left half of the output text at round i.
- R_i is the right half of the output text at round i.
- C_i is the left half of the key at round i.
- D_i is the right half of the key at round i.

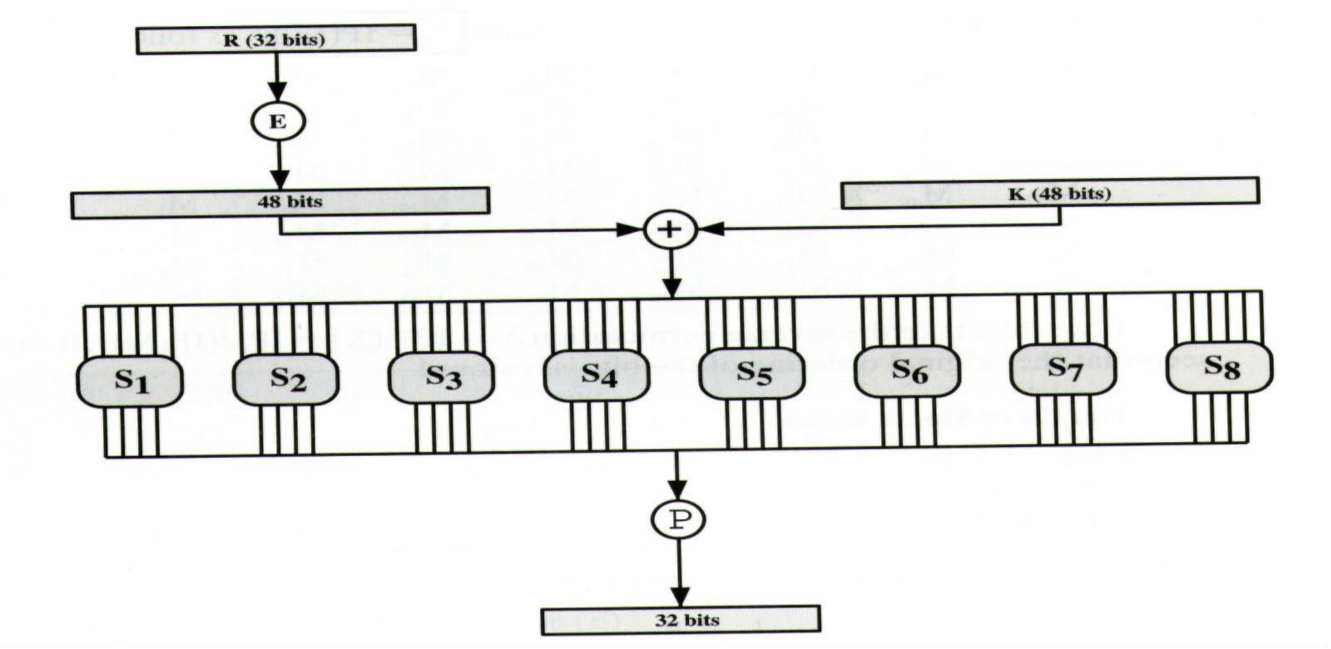
Expansion/Permutation (E table)

- We must expands R from 32 bits into 48 bits, so that it can be combined with the 48 bits K.

- Break R into eight 4-bits chunks and expand each chunk into 6 bits by stealing the rightmost and leftmost bit from the left and right adjacent 4-bits chunks (consult E table).

S-box

- You can think of S-box as just performing a many-to-one mapping from 6-bits numbers to 4-bits numbers.



S-box (cont.)

Middle 4 bits of input

Outer bits

S_1	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13
S_2	15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
	0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
	13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9
S_3	10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
	13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
	1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12
S_4	7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
	10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
	3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14
S_5	2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
	4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
	11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3
S_6	12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
	9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
	4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13
S_7	4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
	1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
	6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12
S_8	13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
	7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
	2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Permutation Tables for DES

(a) Initial Permutation (IP)

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

(b) Inverse Initial Permutation (IP^{-1})

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26
33	1	41	9	49	17	57	25

(c) Expansion Permutation (E)

32bit-48bit expansion

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

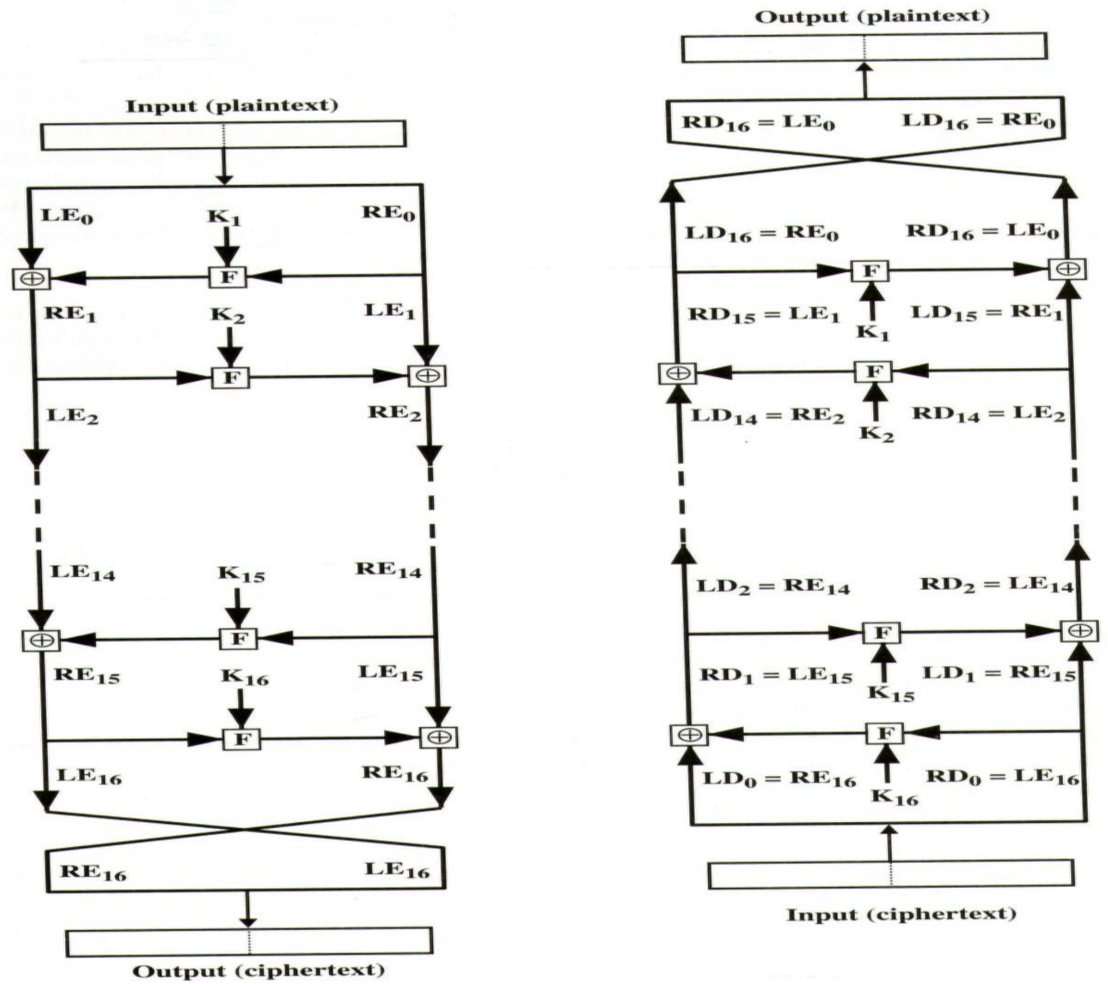
(d) Permutation Function (P)

16	7	20	21	29	12	28	17
1	15	23	26	5	18	31	10
2	8	24	14	32	27	3	9
19	13	30	6	22	11	4	25

How to Decrypt?

- One nice feature of DES is that decryption uses the same algorithm as encryption, except that the application of the subkeys (K_i) is reversed.
- Why?
- Next, we consider a general combiner function F.

How to Decrypt? (cont.)



How to Decrypt? (cont.)

- First, consider the encryption process :

$$LE_{16} = RE_{15}$$
$$RE_{16} = LE_{15} \oplus F\left(RE_{15}, K_{16}\right)$$

- On the decryption side :

$$LD_1 = RD_0 = LE_{16} = RE_{15}$$
$$LD_0 = RE_{16}$$
$$\begin{aligned} RD_1 &= LD_0 \oplus F\left(RD_0, K_{16}\right) \\ &= RE_{16} \oplus F\left(RE_{15}, K_{16}\right) \\ &= \left[LE_{15} \oplus F\left(RE_{15}, K_{16}\right)\right] \oplus F\left(RE_{15}, K_{16}\right) \end{aligned}$$

How to Decrypt? (cont.)

- The XOR has the following properties :

$$[A \oplus B] \oplus C = A \oplus [B \oplus C]$$

$$D \oplus D = 0$$

$$E \oplus 0 = E$$

- Thus, we have :

$$LD_1 = RE_{15}$$

$$RD_1 = LE_{15}$$

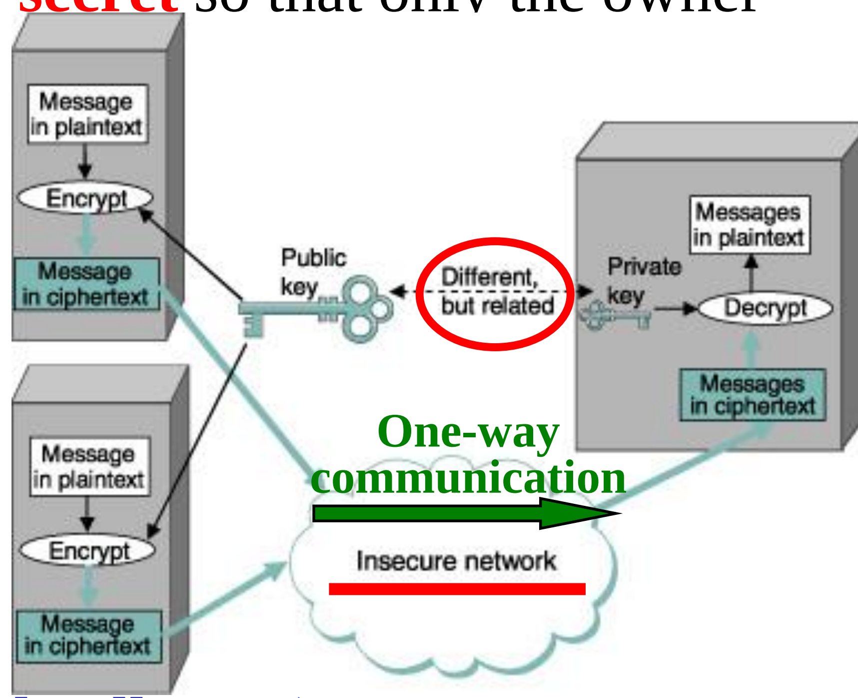
- In general case :

$$LD_i = RE_{16-i}$$

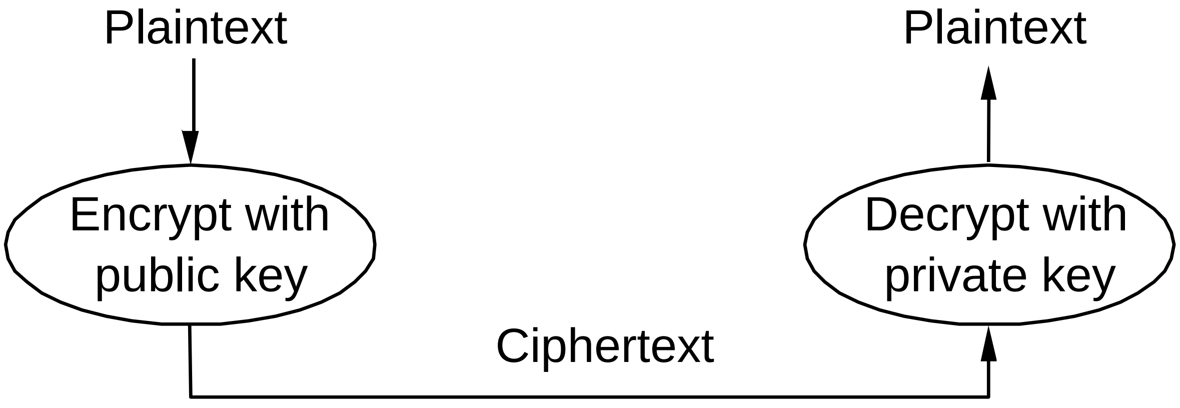
$$RD_i = LE_{16-i}$$

Public-key Ciphers

- A public-key cipher uses a pair of related keys, one for encryption and a different one for decryption
- The pair of keys is "owned" by just one participant
  - Keeps the **decryption key** **secret** so that only the owner can decrypt messages; this key is called the **private key**
  - Make the **encryption key** **public** so that anyone can encrypt messages for the owner; this key is called the **public key**
  - **RSA (Rivest, Shamir, and Adleman)**



Public Key (RSA)



- Encryption & Decryption

$$c = m^e \bmod n$$
$$m = c^d \bmod n$$

RSA (cont.)

- Choose two large prime numbers p and q (each 256 bits)
- Multiply p and q together to get n
- Choose the encryption key e, such that e and $(p - 1) \times (q - 1)$ are relatively prime.
- Two numbers are relatively prime if they have no common factor greater than one
- Compute decryption key d such that

$$d = e^{-1} mod \; ((p - 1) \times (q - 1))$$

- Construct public key as (e, n)
- Construct private key as (d, n)
- Discard (do not disclose) original primes p and q

The Mathematic Theory for RSA

- Theorem:

If $C = M^e \bmod n$

then $M = C^d \bmod n$ (The parameters are defined above.)

Properties of Modular Arithmetic

Lemma 1 :

If a is relatively prime to n and $(a \times b) \bmod n = (a \times c) \bmod n$

then $b \bmod n = c \bmod n$.

pf :

$b \bmod n = c \bmod n$ (同餘), iff $\exists\ p \in Z \ni (b-c) = pn$.

Let $(a \times b) \bmod n = (a \times c) \bmod n = r$

Then $\exists\ p_1, p_2 \in Z \ni \begin{cases} a \times b = p_1 \times n + r \ldots (1) \\ a \times c = p_2 \times n + r \ldots (2) \end{cases}$

Properties of Modular Arithmetic (cont.)

(1)-(2) :

$$(b\text{-}c)a = (p_1\text{-}p_2)n$$

Since a is relatively prime to n, $(b\text{-}c)$ is an integer Multiple of n, *i.e.*,

$$(b\text{-}c) = kn \ \ \textit{for some } n.$$

Thus, $b \bmod n = c \bmod n.$

■

Properties of Modular Arithmetic (cont.)

Lemma 2 :

$ab \bmod n = (a \bmod n)(b \bmod n) \bmod n$

pf :

Let $a \bmod n = r_1 \Rightarrow a = np_1 + r_1$

$b \bmod n = r_2 \Rightarrow b = np_2 + r_2$ where p_1, p_2

then $ab = (np_1+r_1)(np_2+r_2) = n(np_1p_2+p_2r_1+p_1r_2)+r_1r_2$

$\Rightarrow ab \bmod n = [n(np_1p_2+p_2r_1+p_1r_2)+r_1r_2] \bmod n$

$= r_1r_2 \bmod n = (a \bmod n)(b \bmod n) \bmod n$

■

Fermat's Theorem

- Let Z_n is the set of nonnegative integers less then n, $i.e.$, $Z_n = \{0,1,\ldots,(n-1)\}$

Fermat's Theorem (cont.)

Lemma 3 :

Let $Z_p^a = \{0, (a \bmod p), (2a \bmod p), \ldots, ((p-1)a \bmod p)\}$.

If *p* is prime and *a* is a positive integer not divisible by *p*, then

$$Z_p^a = Z_p.$$

Fermat's Theorem (cont.)

Theorem (*Fermat's Theorem*) :

If p is prime and a is a positive integer not divisible by p, then

$$a^{p-1} \bmod p = 1 \bmod p$$

pf :

Since $Z_p^a = Z_p$, the products of all the elements in Z_p^a and Z_p are the same.

Fermat's Theorem (cont.)

Thus,

$$\left(a \times 2a \times \ldots \times (p-1)a\right) \bmod p$$
$$= \left[\left(a \bmod p\right) \times \left(2a \bmod p\right) \times \ldots \times \left(\left(p-1\right)a \bmod p\right)\right] \bmod p$$
$$= \left(p-1\right)! \bmod p$$

Note that

$$a \times 2a \times \ldots \times \left(\left(p-1\right)a\right) = a^{p-1} \times \left(p-1\right)!$$

Therefore,

$$\left[a^{p-1} \times (p-1)!\right] \bmod p = (p-1)! \bmod p$$

Since (p-1)! is relatively prime to p,

$$a^{p-1} \bmod p = 1 \bmod p \quad \text{(Lemma 1)}$$

Euler's Theorem

Define (*Euler's totient funtion*) :

Euler's totient function $\phi(n)$ is defined to be the number of positive integers that are less than n and relatively prime to n.

- It is clear that for a prime number p ,

$$\phi(p) = p - 1.$$

- Then, for $n = pq$ (p and q are two prime numbers)

$$\begin{aligned}\phi(n) = \phi(pq) &= pq - [(q-1) + (p-1) + 1] \\ &= pq - (p+q) + 1 = (p-1)\times(q-1) \\ &= \phi(p)\times\phi(q)\end{aligned}$$

Euler's Theorem (cont.)

Theorem (*Euler's Theorem*) :

For every a and n that are relatively prime, then

$$a^{\phi(n)} \bmod n = 1 \bmod n.$$

pf :

Let R be the set of all integers that are less than n and relatively prime to n,

$$R = \{x_1, x_2, \ldots, x_{\phi(n)}\}.$$

Now multiply each element by a, and then modulo n,

$$S = \{ax_1 \bmod n, ax_2 \bmod n, \ldots, ax_{\phi(n)} \bmod n\}.$$

Euler's Theorem (cont.)

Then $R = S$.

1. Since a is relatively prime to n and x_i is relatively prime to n, ax_i must also be relatively prime to n. Th-us, all the members of S are intergers less than n and they are relatively prime to n.
2. All the elements in S are distinct. If $ax_i \bmod n = ax_j \bmod n$, then $x_i = x_j$ (contraction to that all the element in R are distinct.)

Euler's Theorem (cont.)

$$\prod_{i=1}^{\phi(n)} \left(ax_i \bmod n\right) = \prod_{i=1}^{\phi(n)} x_i$$

$$\Rightarrow \left(a^{\phi(n)} \prod_{i=1}^{\phi(n)} x_i \right) \bmod n = \left(\prod_{i=1}^{\phi(n)} x_i \right) \bmod n$$

$$\Rightarrow a^{\phi(n)} \bmod n = 1 \bmod n \quad \text{(Lemma 1)}$$

$\blacksquare$

The Mathematical Theory for RSA (cont.)

Let's recall the definitions of all parameters.

- p, q, two prime numbers.
- $n = pq$.
- e is relatively prime to $(p-1)(q-1)$, i.e. $\gcd((p-1)(q-1), e) = 1$.
- $de \bmod [(p-1)(q-1)] = 1 \bmod [(p-1)(q-1)]$

Now, we need to prove :

$$\text{if } C = M^e \bmod n$$
$$\text{then } M = C^d \bmod n \ \forall M < n$$

The Mathematical Theory for RSA (cont.)

pf :

$de = k(p-1)(q-1)+1 = k\phi(n)+1$, where k is an integer.

If M is relatively prime to n, then

$$
\begin{aligned}
C^d \bmod n &= \left(M^e\right)^d \bmod n \\
&= M^{ed} \bmod n \\
&= M^{k\phi(n)+1} \bmod n \\
&= \left[\left(M^{\phi(n)}\right)^k \times M\right] \bmod n \\
&= \left[\left(\left(M^{\phi(n)}\right)^k \bmod n\right) \times \left(M \bmod n\right)\right] \bmod n \quad \text{(Lemma 2)} \\
&= M \bmod n = M \qquad \forall M < n
\end{aligned}
$$

The Mathematical Theory for RSA (cont.)

Suppose M is not relatively prime to n and $M < n = pq$. W.L.G., let $M = sp$ for some integer s. From Euler's theorem,

$$M^{\phi(q)} \bmod q = 1 \bmod q$$

$$\Rightarrow M^{k\phi(q)} \bmod q = 1 \bmod q$$

$$\Rightarrow \left[M^{k\phi(q)}\right]^{\phi(p)} \bmod q = 1 \bmod q$$

$$\Rightarrow M^{k\phi(n)} \bmod q = 1 \bmod q$$

$$\Rightarrow M^{k\phi(n)} = 1 + tq \qquad \text{for some integer } t$$

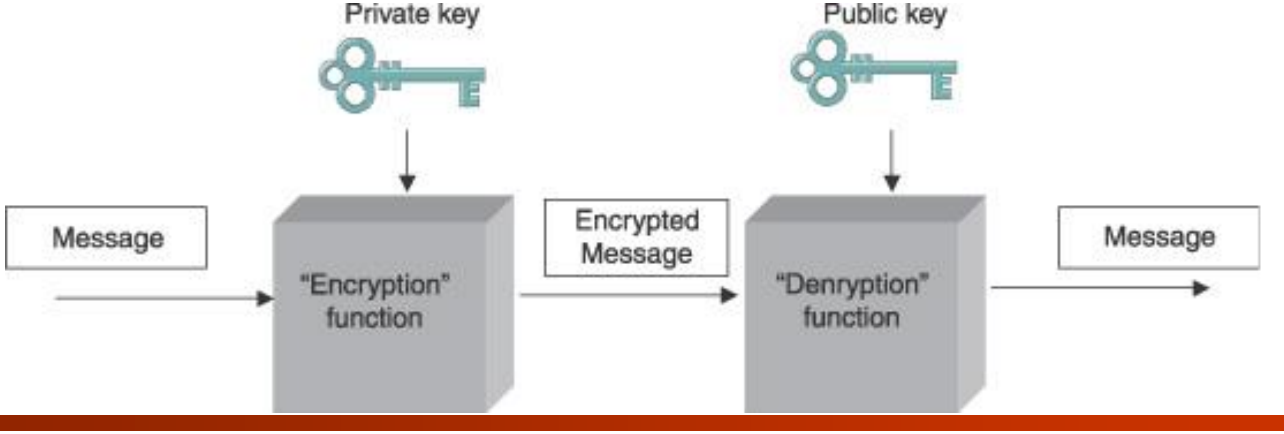
Multiplying each side by $M=sp$, then

$$M^{ed} = M^{k\phi(n)+1} = M + tspq = M + tsn$$

$$\Rightarrow M^{ed} \bmod n = M \bmod n = M \qquad \forall \mathrm{M} < n$$

Authentication

- Another application of public-key ciphers is **authentication**
- The **private key** can be used with the **encryption** function to encrypt messages so that they can only be **decrypted** using the **public key**
- **Anyone** with the public key could decrypt such a message
- It tells the receiver that such a message could only have been **created** by the **owner of the keys**
  - **Authenticate** the owner of the keys

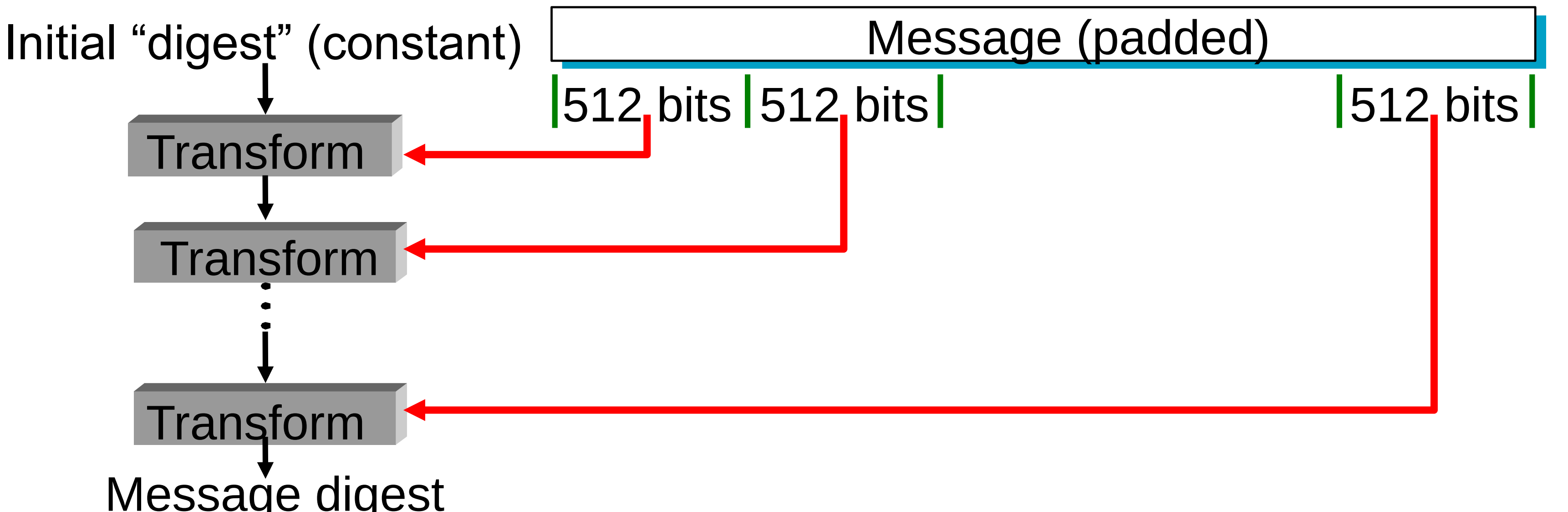


Message Integrity

- Encryption alone does not provide **data integrity**
- Sometimes two participants are worried about the possibility of an impostor sending message that claim to be from one of them
- An **authenticator** is a value, to be included in a transmitted message, that can be used to verify simultaneously the **authenticity** and the **data integrity** of a message
- An authenticator includes **redundant information** about the message contents
  - Like a checksum or cyclic redundancy check (CRC)
  - Also known as a **message integrity code (MIC)**
  - The receiver can check the MIC to verify the **validity** of the message

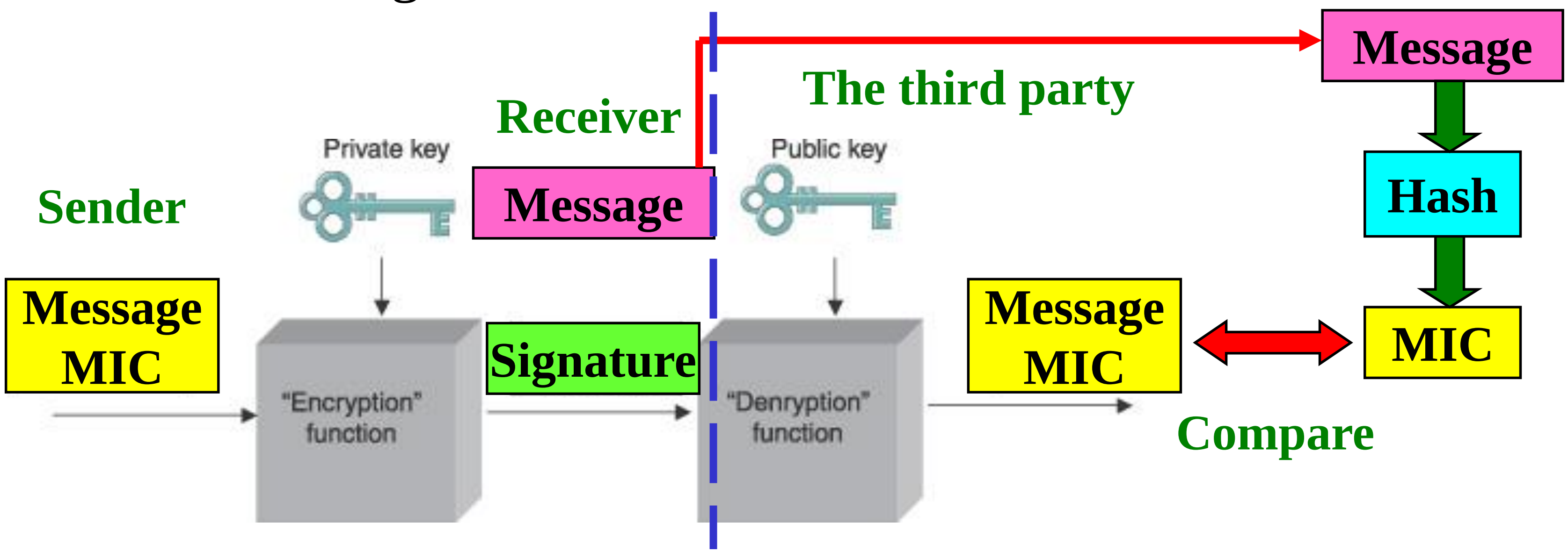
Hash Function

- One way to generate an authenticator is using a hash function
- **Hashing algorithms (message digest function):** does not involve the use of keys
  - Map a potentially **large message** into a **small fixed-length number** (cryptographic **checksum**)
  - **MD5 (Message Digest version 5)**



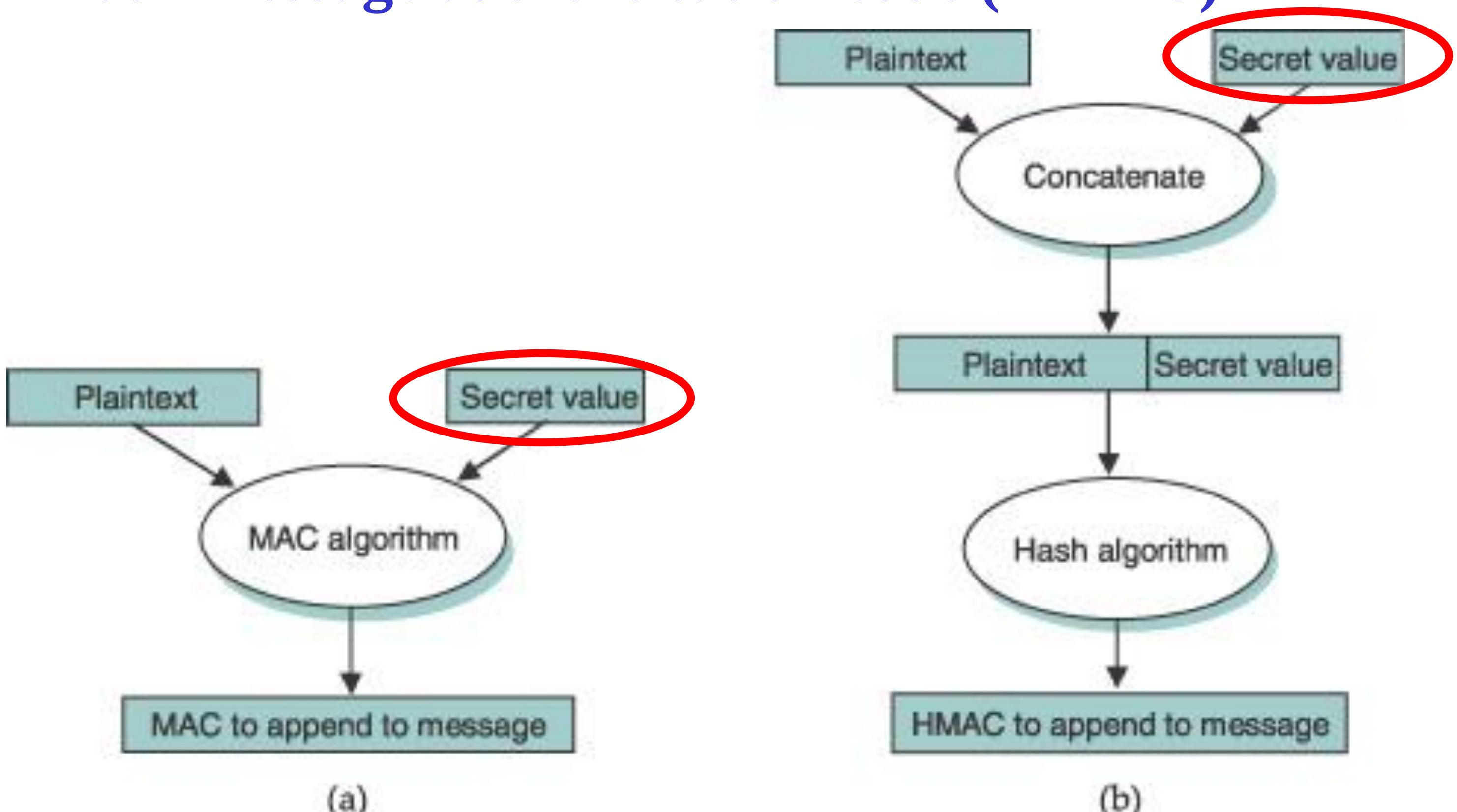
Message Integrity – Signature

- A digest encrypted with a public-key algorithm but using the **private key** is called a **digital signature**
- The receiver of a message with a digital signature can prove to any **third party** that the sender **really sent** that message
  - The third party can use the sender's **public key** to check for the message



Message Integrity – Hash with Secret Value

- Take a **secret value** known to only the sender and the receiver
  - **Message authentication code (MAC)**
  - **Hash message authentication code (HMAC)**



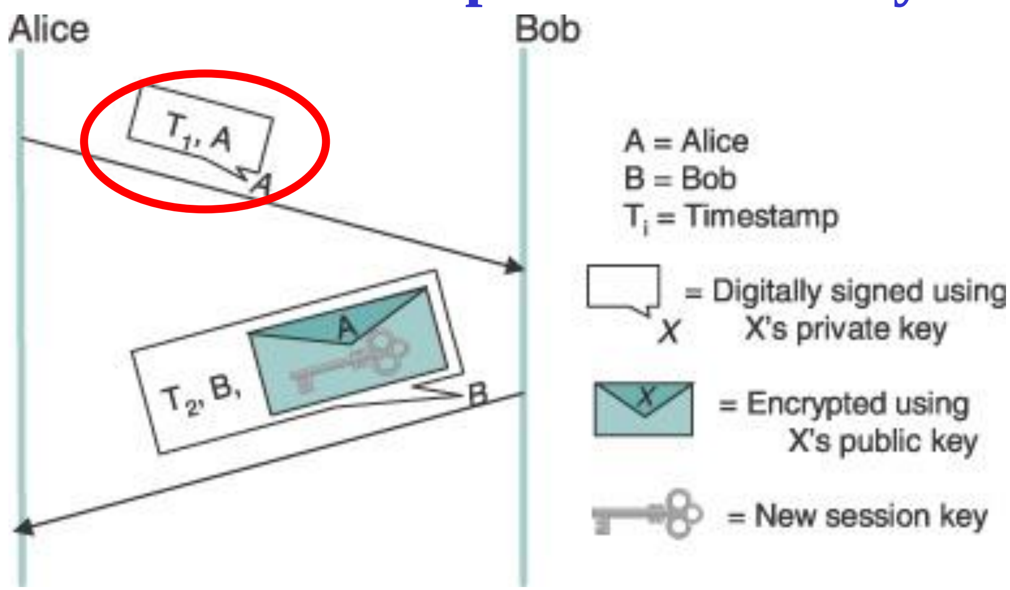
Authentication Protocols

Authentication Protocols

- Before two participants are likely to establish a secure channel between themselves
  - They wish to **verify** that the other participant is who he or she **claims to be**
- The authentication protocols may base on:
  - Using **secret key cryptography** (such as DES)
    - Need to share a secret key
  - Using **public key cryptography** (such as RSA)
    - Do not need to share a secret key

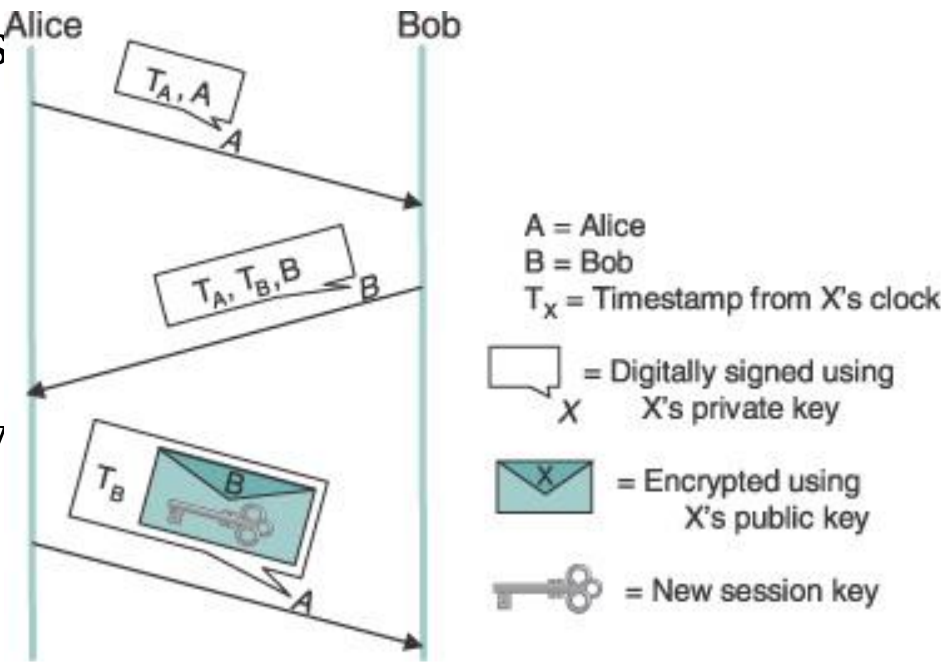
Public-key Authentication (Synchronous)

- In the first protocol, Alice and Bob's clocks are **synchronized**
- Alice sends Bob a message with a **timestamp** and her **identity** in plaintext plus her **digital signature**
- Bob uses the digital signature to authenticate the message, and the timestamp to verify its **freshness**
- Bob sends back a message with a **timestamp** and his **identity** in plaintext, and a new **session key** encrypted by Alice's **public key**, all digitally signed
- Alice can verify the authenticity and freshness of the message



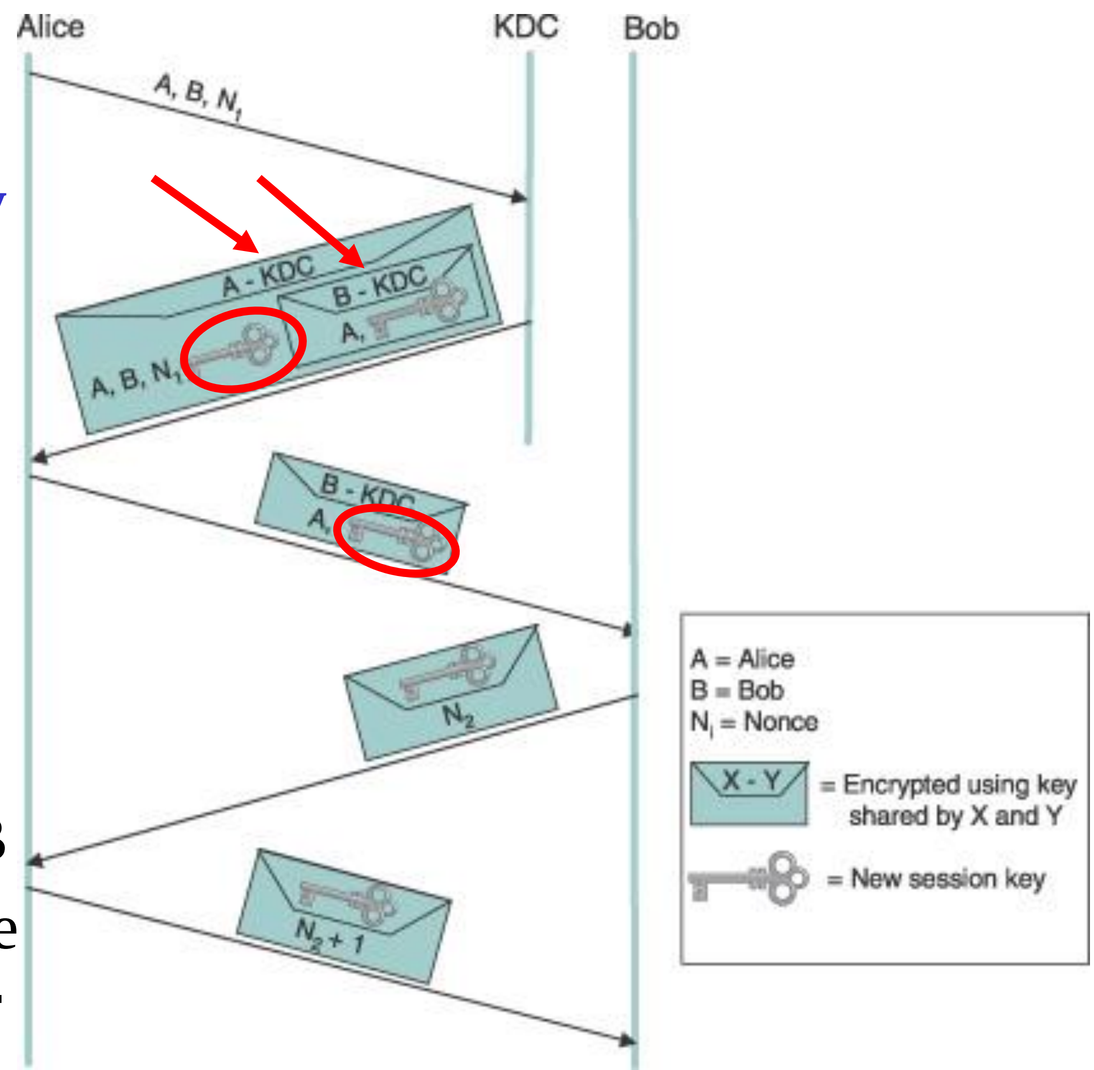
Public-key Authentication (Asynchronous)

- The second protocol **does not** rely on clock synchronization
- Alice sends Bob a digitally signed message with $\mathbf{T_A}$ and **A**
- Bob cannot be sure that the message is fresh, since their clocks are **not** synchronized
- Bob sends back a digitally signed message with $\mathbf{T_A}$, $\mathbf{T_B}$ and **B**
- Alice can verify the freshness of Bob's reply by comparing her **current time**
- Alice sends Bob back a signed message with $\mathbf{T_B}$ and an encrypted new session key
- Bob can verify the freshness of Alice's reply



Symmetric-key Authentication

- It involves three parties: Alice, Bob, and a **KDC**
  - KDC is a trusted **key distribution center** (also known as **Authentication Server, AS**)
- A (B) and KDC already share a secret key
- Finally, a **session key** is shared between A and B
- A & B can communicate **directly** with each other



Diffie-Hellman Key Agreement

- Two parameter p and g
- Parameter p is a prime number
- Parameter g is the generator of the group {1,2, …, p-1}
- For every n in {1,2, …, p-1}, there is some k such that $n=g^k \bmod p$
- Alice generates a random number a in {1,2, …, p-1}
- Bob generates a random number b in {1,2, …, p-1}
- Alice's public value is $g^a \bmod p$
- Bob's public value is $g^b \bmod p$
- Then they exchange their public values

Diffie-Hellman Key Agreement

- Alice computes g^{ab} mod p= (g^{b} mod p)^{a} mod p
- Bob computes g^{ab} mod p= (g^{a} mod p)^{b} mod p
- Alice and Bob now have g^{ab} mod p as their shared symmetric key
- Discrete logarithm problem: knowing the public value public value is g^{a} mod p is difficult to compute a for suitably large p
- Note that Diffie-Hellman does not authenticate the participants
- Suffer from the man-in-the-middle attack

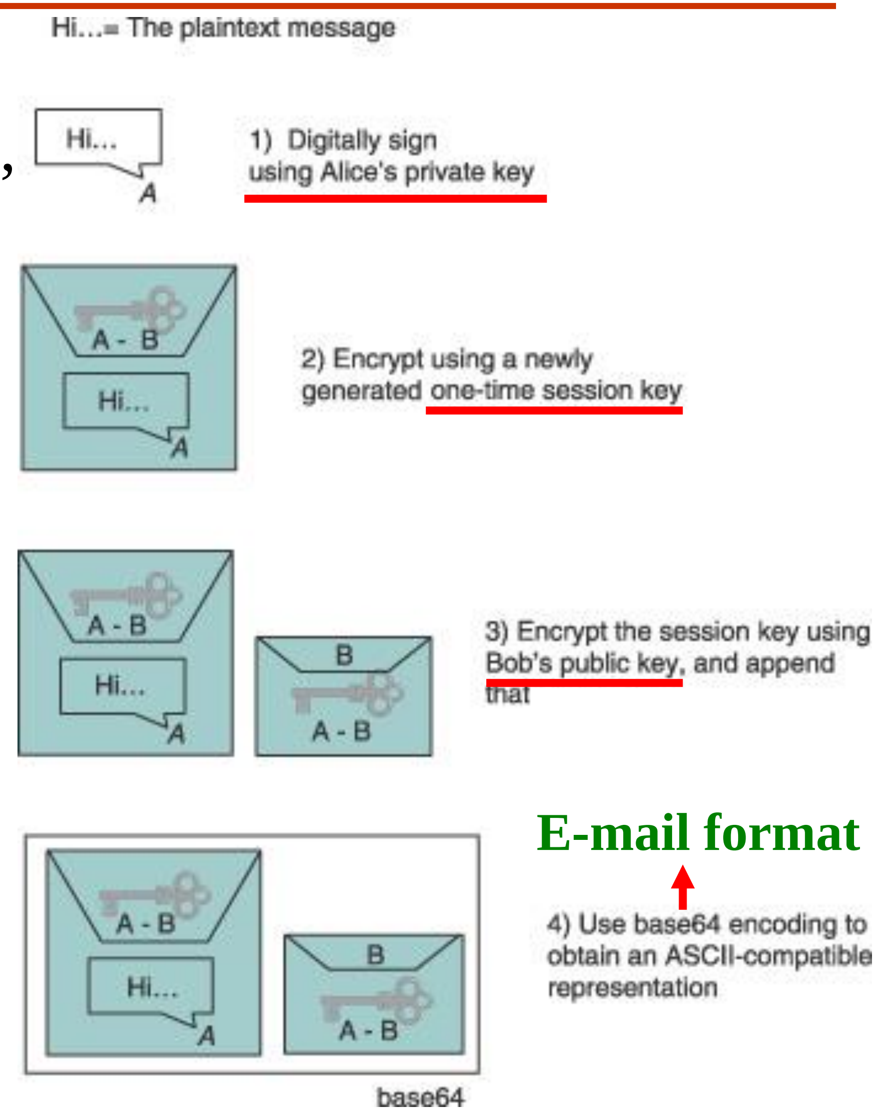
Secure Systems

Secure Systems

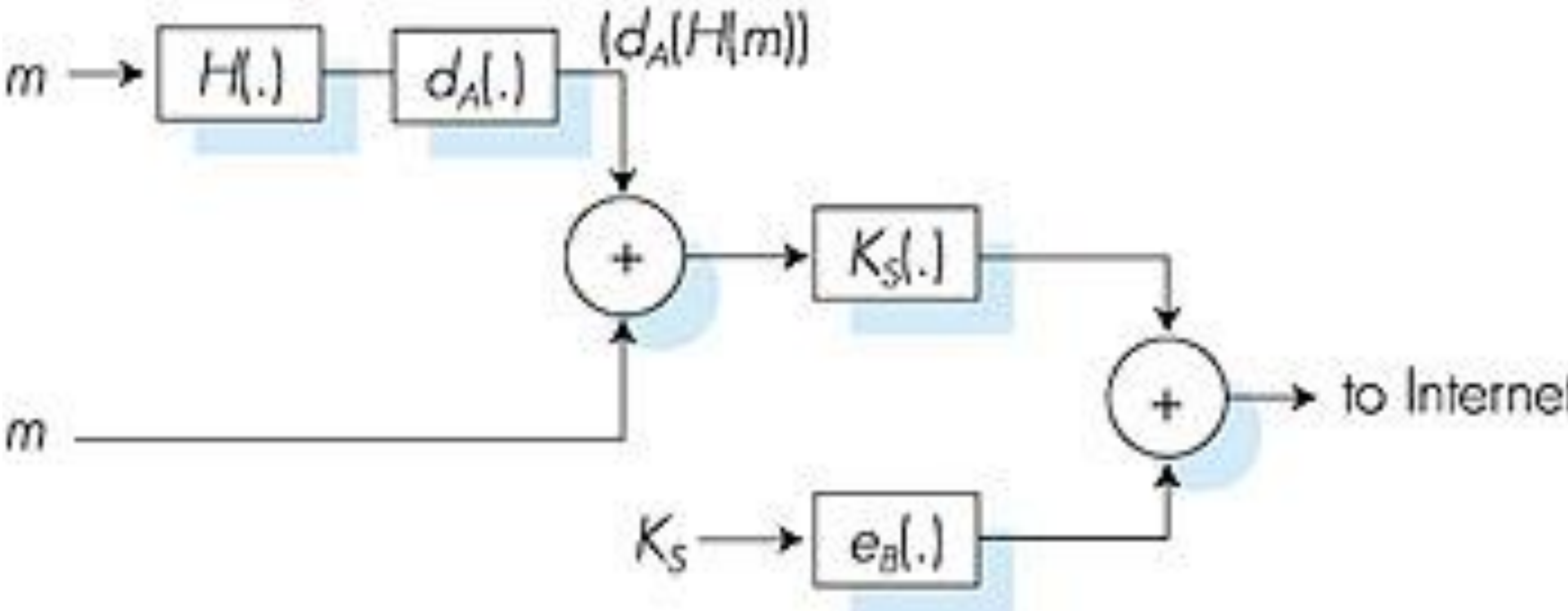
- Systems that operate at the **application** layer:
  - **Pretty Good Privacy (PGP)** provides electronic mail security
  - **Secure Shell (SSH)** provides secure remote login services
- For **transport** layer:
  - **Transport Layer Security (TLS)**
- For **network (IP)** layer:
  - **IP Security (IPsec) protocols**
- For **data link** layer:
  - **IEEE 802.11i** for WLAN

Pretty Good Privacy (PGP)

- PGP provides **authentication, confidentiality**, **data integrity**, and **non-repudiation**
- The confidentiality, and **receiver** authentication rely on the receiver having a known **public key**
- The non-repudiation, and **sender** authentication rely on the sender having a known **public key**



- Provide secrecy, sender authentication, message integrity.



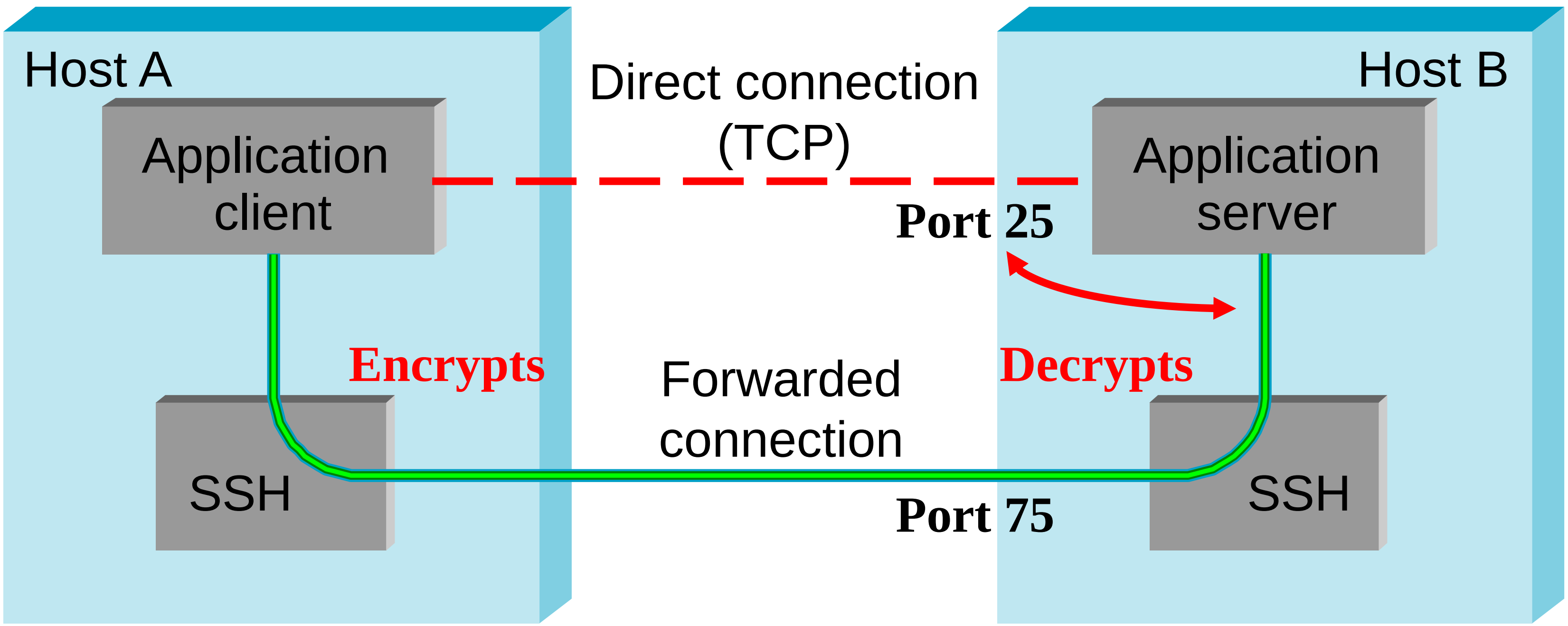
Note: Alice uses both her private key, Bob's public key.

Secure Shell (SSH)

- The SSH provides a **remote login service** and is intended to replace the less secure Telnet and rlogin programs
- SSH consists of three protocols:
  - **SSH-TRANS:** is a **transport layer** protocol
  - **SSH-AUTH:** is an **authentication** protocol
  - **SSH-CONN:** is a **connection** protocol
- SSH can also support other insecure **TCP-based** applications
  - Run the applications over a secure **"SSH tunnel"**
  - Use the SSH-CONN protocol

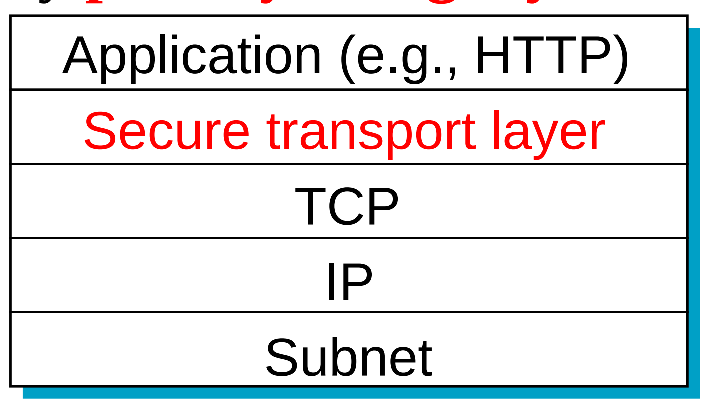
Secure Shell (SSH)

- When messages arrive at the well-known **SSH port** on the server
  – SSH **decrypts** the connects, and then
  – Forwards the data to the **actual port** at which the server is listening



Transport Layer Security (TLS)

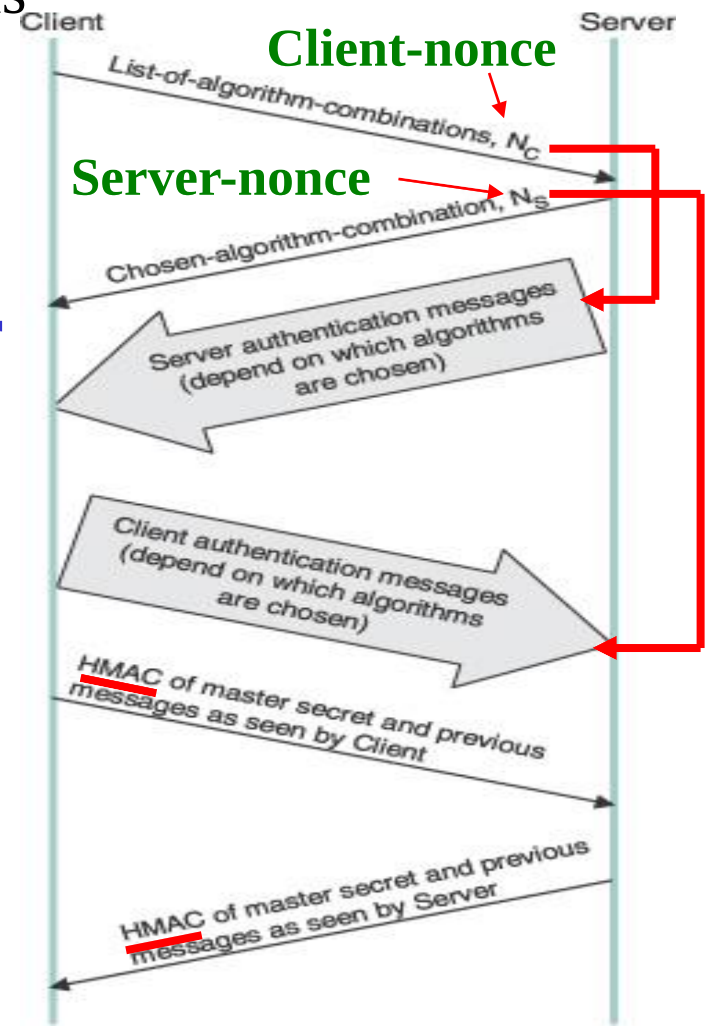
- Since **World Wide Web** becomes popular and has been applied for **commercial applications**
  - Such as making purchases by **credit card**
  - Some level of security would be necessary for transactions on the Web
- **TLS** looks just like a **normal transport protocol**, except for the fact that it is **secure**
  - Provides the necessary **privacy**, **integrity**, and **authentication**



Transport Layer Security (TLS)

- When HTTP is used in this way, it is known as **HTTPS (secure HTTP)**
  - HTTP is **unchanged**
  - It simply delivers data to and accepts data from the **TLS layer** rather than TCP
- TLS is broken into two parts:
  - **Handshake protocol:** is used to negotiate parameters of the communication
  - **Record protocol:** is used for actual data transfer

Handshake Protocol



Secure electronic transactions (SET)

- designed for payment-card transactions over Internet.
- provides security services among 3 players:
  - customer
  - merchant
  - merchant's bank

  All must have certificates.
- SET specifies legal meanings of certificates.
  - apportionment of liabilities for
- Customer's card number passed to merchant's bank without merchant ever seeing number in plain text.
  - Prevents merchants from stealing, leaking payment card numbers.
- Three software components:
  - Browser wallet
  - Merchant server
  - Acquirer gateway
- See text for description of SET

IP Security (IPSEC)

- **IPSEC** consists of two pieces:
  - The first piece is a pair of protocols that implement the available **security services**
    - **Authentication Header (AH):** provides **access control**, connectionless message **integrity**, **authentication** and **anti-replay** protection
    - **Encapsulating Security Payload:** supports these same services, plus **confidentiality**
  - The second piece is the support for **key management**
    - **ISAKMP: Internet Security Association and Key Management Protocol**

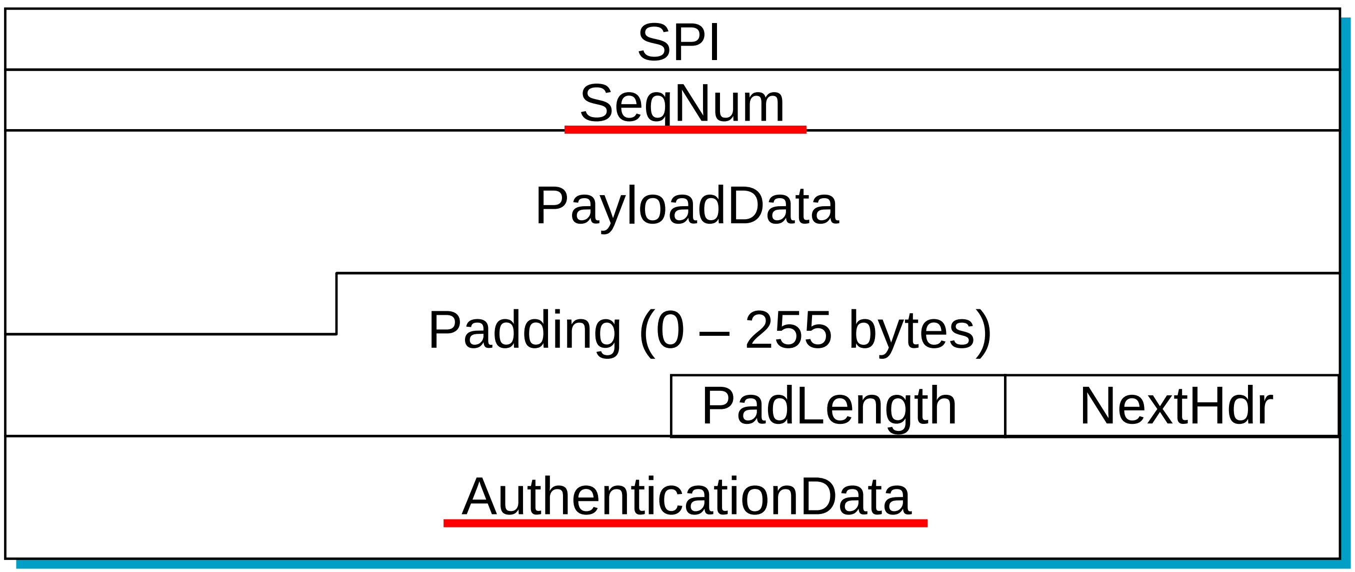
IP Security (IPSEC)

- **Authentication Header (AH):**
  - **NextHdr:** is the type of the next payload after the AH
  - **PayloadLength:** is the length of the AH
  - **Reserved:** is reserved and set to 0
  - **SPI:** identifies the security association for this datagram
  - **SeqNum:** is used to protect against **replay**
  - **AuthenticationData:** contains the **message integrity code** for this packet

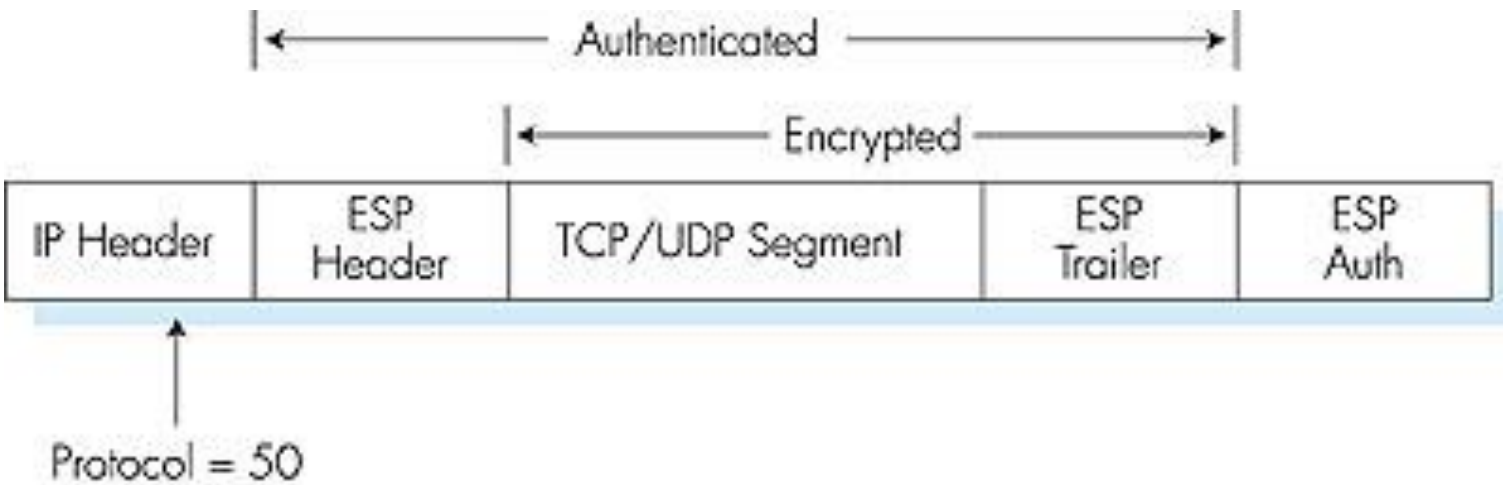
NextHdr	PayloadLength	Reserved
SPI		
SeqNum		
AuthenticationData		

IP Security (IPSEC)

- **Encapsulating Security Payload (ESP):**
  - **PayloadData:** contains the data described by the NextHdr
  - **PadLength:** is the length of the padding



ESP Tunnel Mode

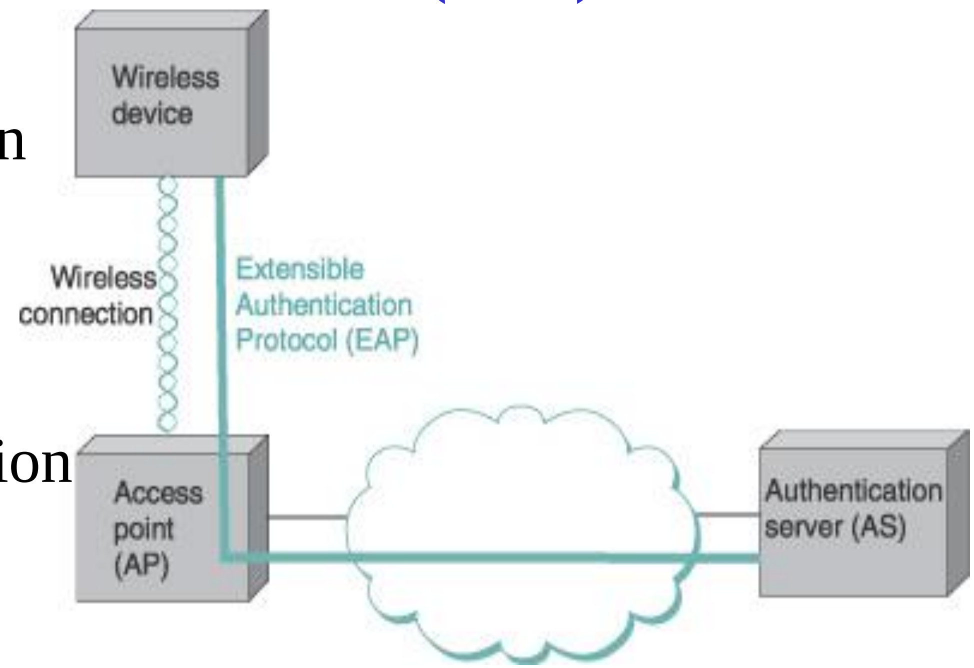


Wireless Security (IEEE 802.11i)

- **IEEE 802.11i** provides **authentication**, **message integrity**, and **confidentiality** to IEEE 802.11 at the **link layer**
- 802.11i authentication supports two modes. In either mode, the end result of successful authentication is a shared pairwise master key
  - **Personal mode:** provides weaker security but is more convenient and economical for situations like a **home** 802.11 network
    - Uses a **preconfigured password**
    - Between **wireless device** and the **AP** (access point)
  - **Stronger authentication mode:** is based on the IEEE 802.1X framework for controlling access to a LAN
    - Uses an **authentication server (AS)**

Wireless Security (IEEE 802.11i)

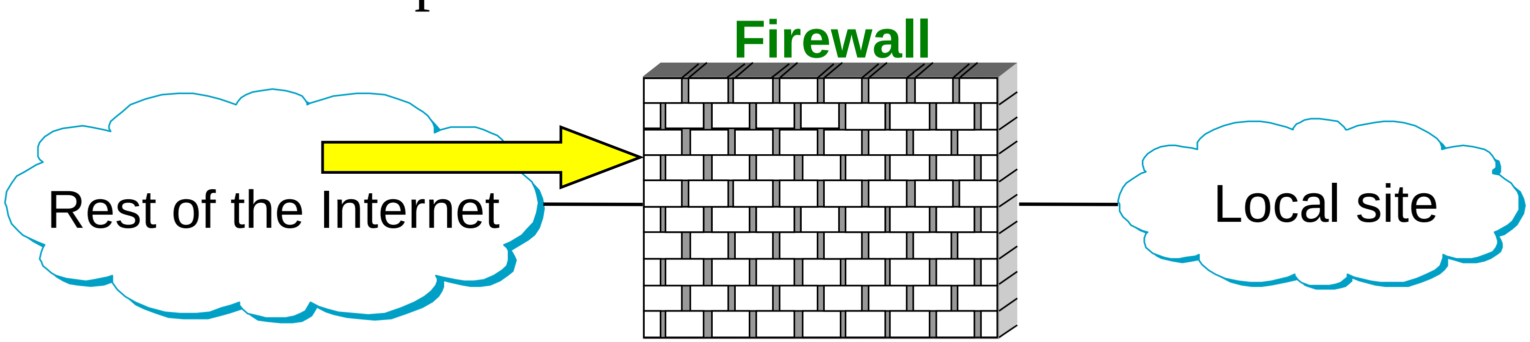
- For stronger authentication mode, the **AS** and **AP** must be connected by a **secure** channel
  - The AP forwards authentication messages between the wireless device and the AS
- The **Extensible Authentication Protocol (EAP)** is used
  - Is designed to support **multiple** authentication methods
  - smart cart, Kerberos, one-time passwords, public-key authentication



Firewalls

Firewalls

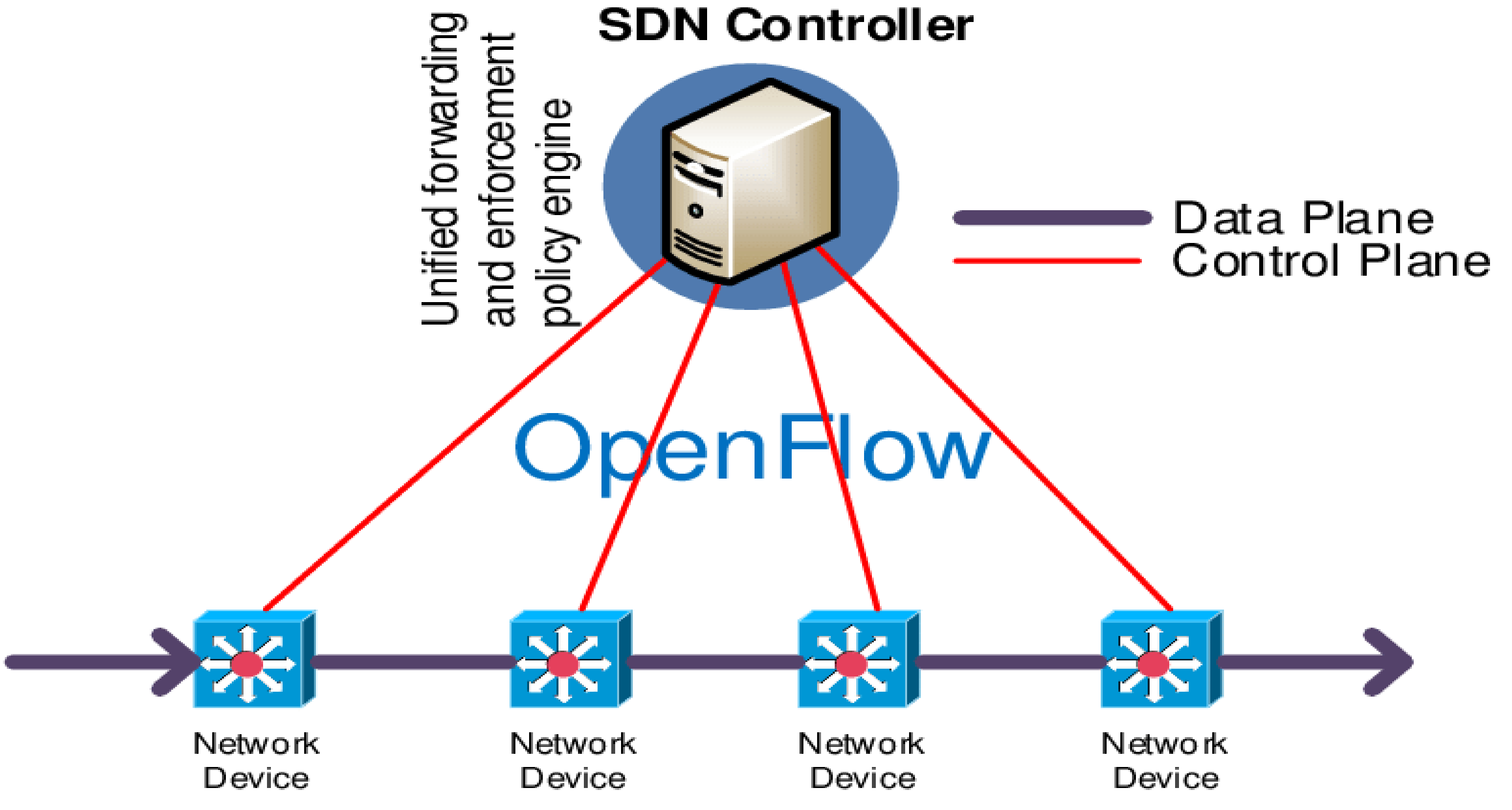
- A **firewall** is a specially programmed **router** that sits between a site and the rest of the network
  - A router connects to two or more networks and it **forwards or filters** the packets that flow through it
- The firewall might filter packets **based on the destination IP or source IP**
  - Prevent external users to **access** a particular host
  - Prevent an unwanted **flood** of packets from an external host
- Such a flood of packets is called a **denial-of-service attack**



Filter-Based Firewalls

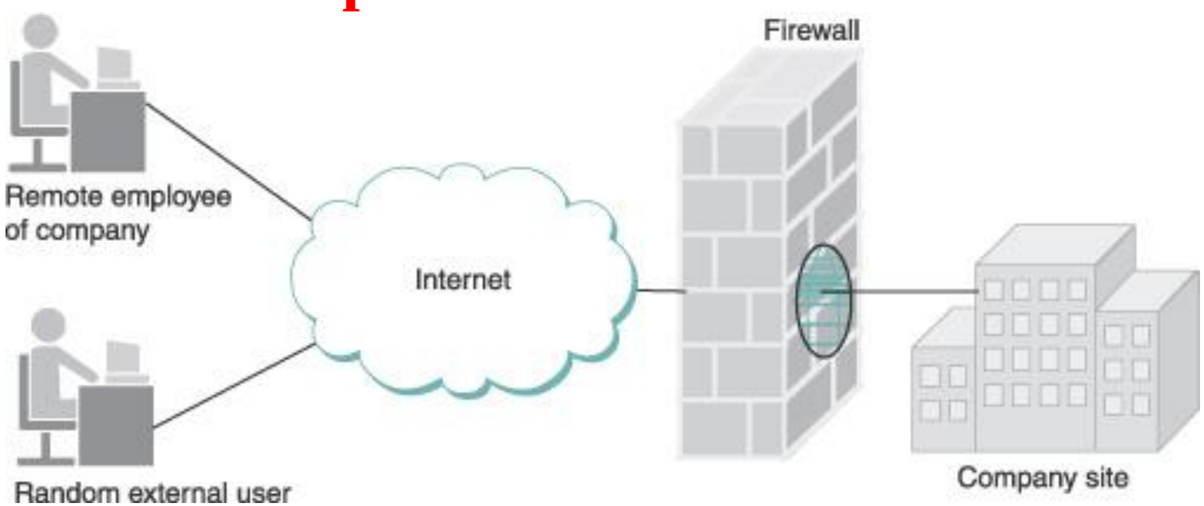
- **Filter-based firewalls** are the simplest and most widely deployed type of firewalls
  - Configured with **a table of addresses** that characterize the packets they **will**, and **will not**, be forwarded
- Generally, each entry in the table is a 4-tuple:
  - It gives the **IP address and TCP port number for both the source and destination**
- For example: to filter **<192.12.13.14, 1234, 128.7.6.5, 80>**
  - Filter all packets from port 1234 on host 192.12.13.14 addressed to port 80 on host 128.7.6.5
- For example: to filter **<*, *, 128.7.6.5, 80>**
  - Filter **all packets** addressed to port 80 on host 128.7.6.5

SDN and NFV


SDN Controller
Unified forwarding
and enforcement
policy engine
Data Plane
Control Plane
OpenFlow
Network
Device
Network
Device
Network
Device
Network
Device

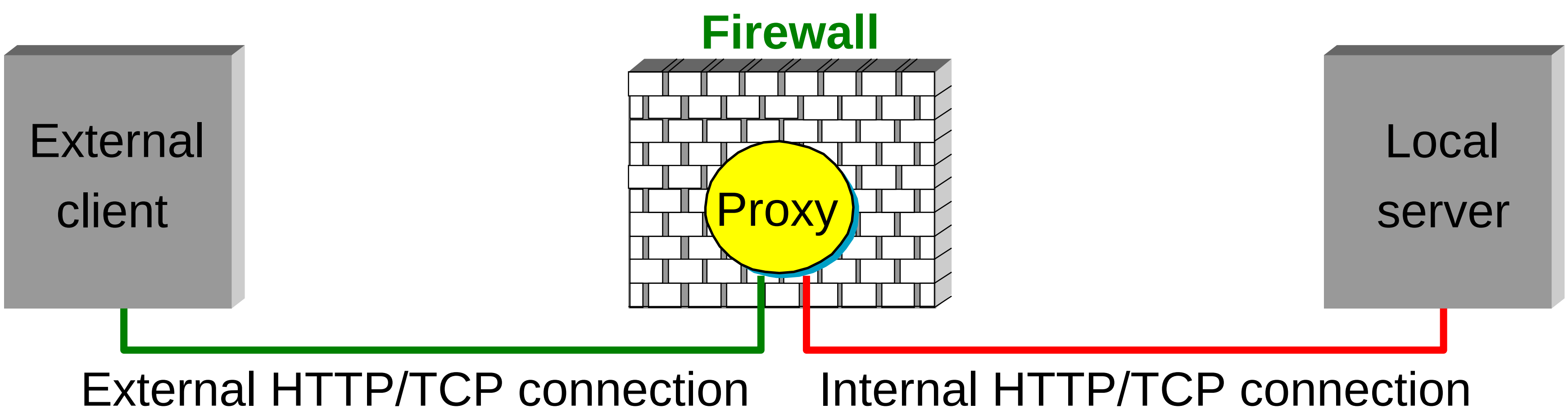
Proxy-Based Firewalls

- A **proxy** is a process that sits between a **client process** and a **server process**
  - To the client, the proxy appears to be the **server**
  - To the server, the proxy appears to be the **client**
- Considering a corporate Web server, some of the server's pages are accessible to **all external users**, and some pages are restricted to **corporate users** at one or more remote sites



Proxy-Based Firewalls

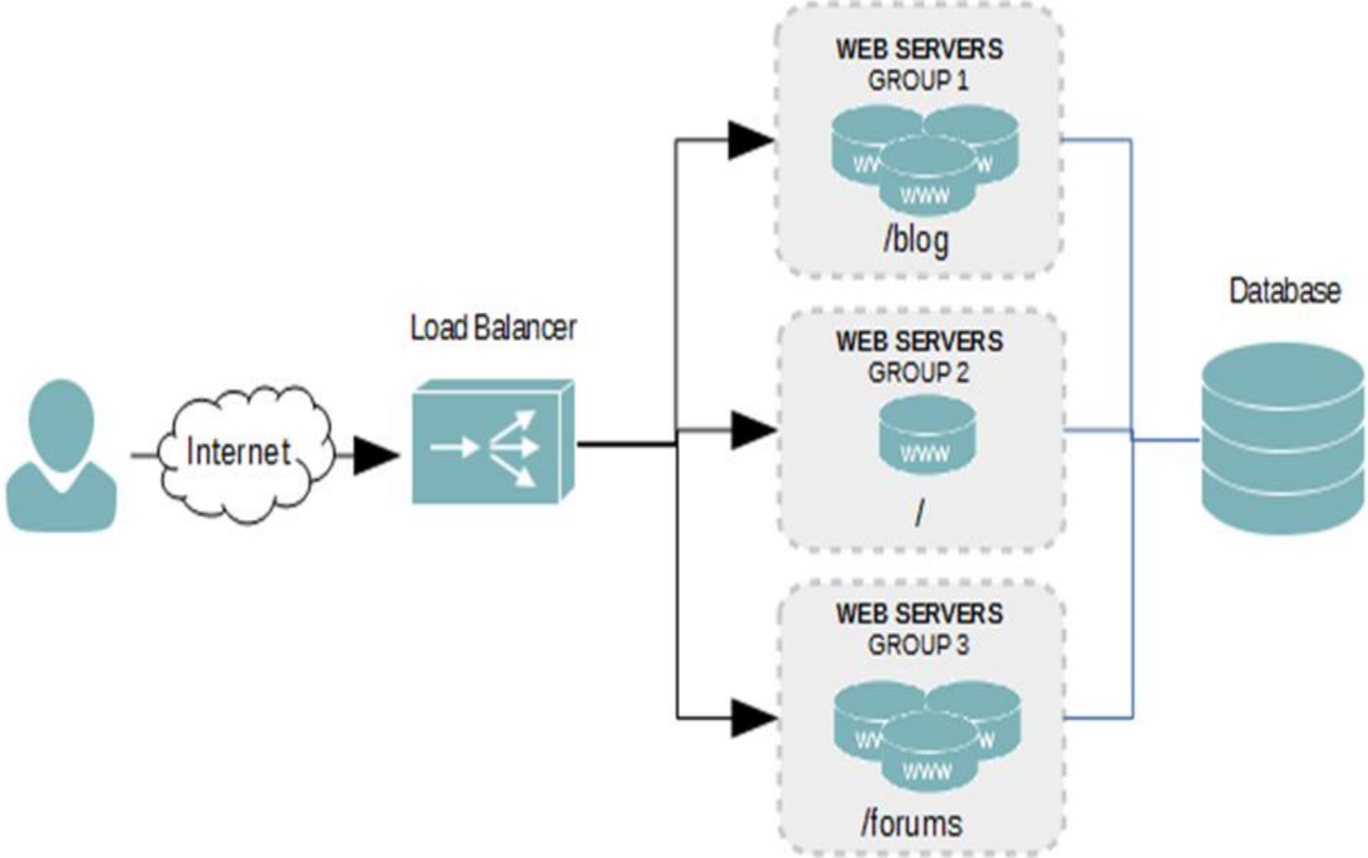
- The solution is to **put an HTTP proxy on the firewall**
- Remote users establish an **HTTP/TCP** connection to the proxy
- The proxy looks at the **URL** contained in the request message
  - If the requested page is **allowed** for the source host
    - The proxy establishes **a second HTTP/TCP connection** to the server and forwards the request to the server
    - The proxy forwards the response to the remote user



Proxy-Based Firewalls

- If the requested page is **not allowed**
  - The proxy does not create this second connection
  - **Returns an error to the source**
- The proxy has to **understand** the HTTP protocol in order to response to the client
- Once an HTTP proxy is in place for security reasons
  - It might be extended to decide which of **many local Web servers** to forward a given request to **(i.e. load balance)**
  - It might also **cache hot Web pages**
    - Access the server only once for multiple requests
- Proxies can be defined for applications other than HTTP
  - For example, FTP and Telnet proxies

Load balancing



Amazon AWS: virtual machine, virtual network, virtual swtich