네트워크 보안

What is network security

- **confidentiality** : 오직 메세지를 주고 받는 사람만 내용을 이해할 수 있어야 함 - 암호화
- **authentication** : 주고 받는 사람이 누군지 확실히 인증해야함
- **message integrity** : 메세지가 변형된 것이 아닌지 확인해야함
- **access and availability** : 서비스는 접근될수 있고 사용될 수 있어야 함

Friends and enemies : Alice, Bob, Trudy

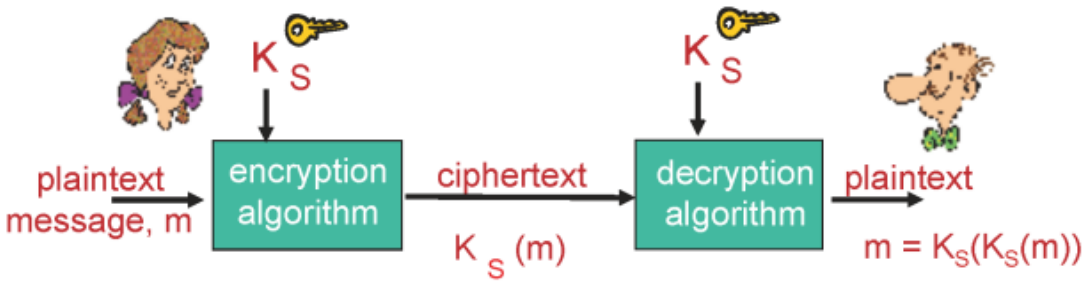
TOR : destination을 알려주지 않는 툴

Warning message : Router에 블랙리스트를 넣어 접근할 때 경고

→ TCP 연결은 함. HTTP req 보내면 중간 router에서 이걸 막아서 warning message를 포함한 response를 보냄 (dest인척)

Symmetric key cryptography

Symmetric key cryptography



symmetric key crypto: Bob and Alice share same (symmetric) key: K_S

- e.g., key is knowing substitution pattern in mono alphabetic substitution cipher

Q: how do Bob and Alice agree on key value?

둘이 같은 키(symmetric key)로 암호화/복호화하면 됨

Public Key Cryptography

symmetric key crypto

- requires sender, receiver know shared secret key
- Q: how to agree on key in first place (particularly if never "met")?

public key crypto

- radically different approach [Diffie-Hellman76, RSA78]
- sender, receiver do *not* share secret key
- *public* encryption key known to *all*
- *private* decryption key known only to receiver

같은 키를 가질 수 있는 방법?

→ Diffie가 공개키 방법 제안

RSA : another important property

The following property will be *very* useful later:

$$\underbrace{K_B^-(K_B^+(m))}_{\text{use public key first, followed by private key}} = m = \underbrace{K_B^+(K_B^-(m))}_{\text{use private key first, followed by public key}}$$

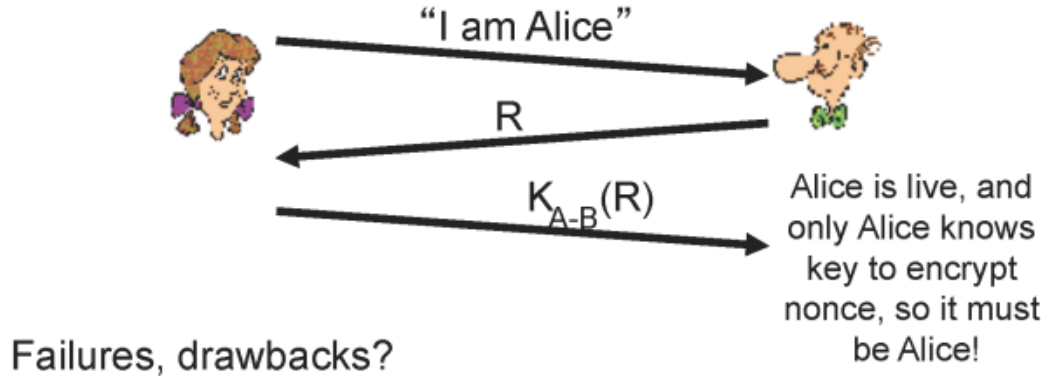
result is the same!

공개키나 개인키 어떤 것을 먼저 적용해도 상관 없음!

Goal: avoid playback attack

nonce: number (R) used only *once-in-a-lifetime*

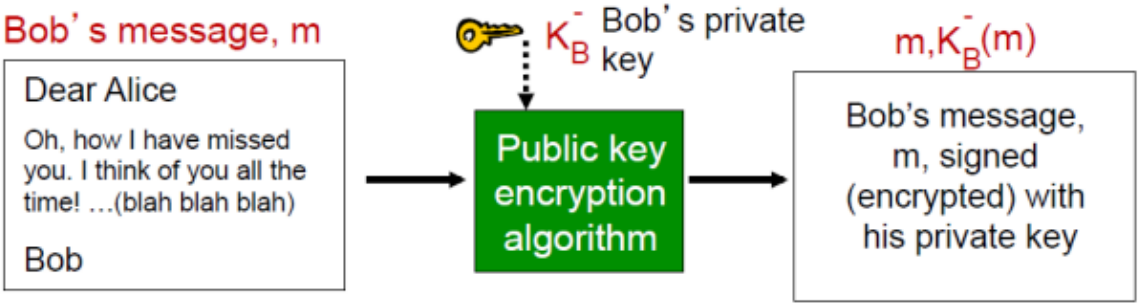
ap4.0: to prove Alice "live", Bob sends Alice *nonce*, R. Alice must return R, encrypted with shared secret key



Failures, drawbacks?

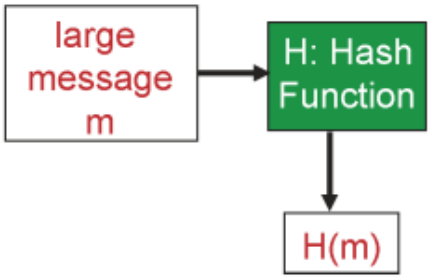
Digital signatures

simple digital signature for message m



Bob이 보내고 싶은 msg를 bob의 public key로 암호화 → Alice는 bob의 public key로 복호화, 변형됐는지 확인

Message digests



computationally expensive to public-key-encrypt long messages

goal: fixed-length, easy- to-compute digital "fingerprint"

- apply hash function H to *m*, get fixed size message digest, *H(m)*.

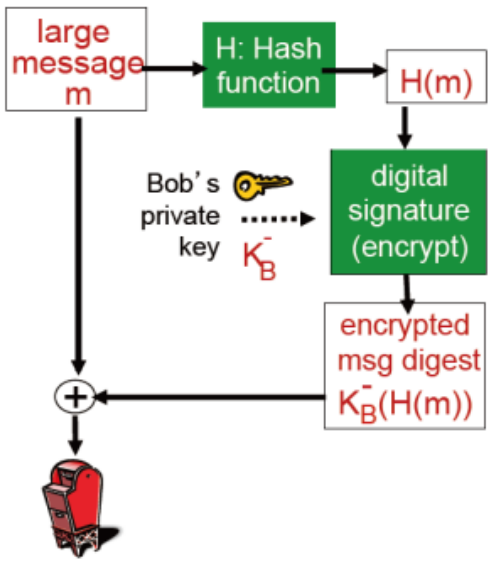
Hash function properties:

- many-to-1
- produces fixed-size msg digest (fingerprint)
- given message digest x, computationally infeasible to find m such that x = H(m)

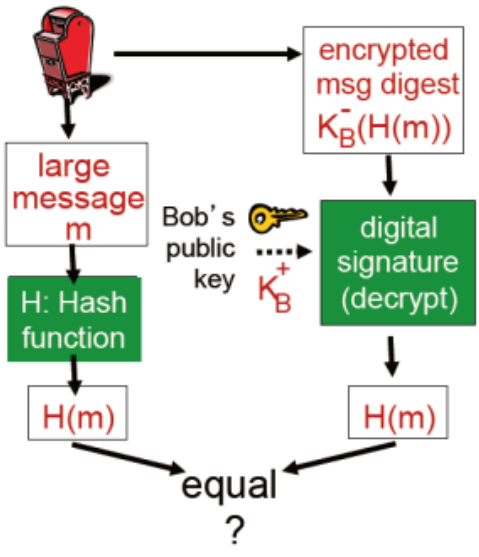
Message의 Hash값을 주로 사용

Digital signature = signed message digest

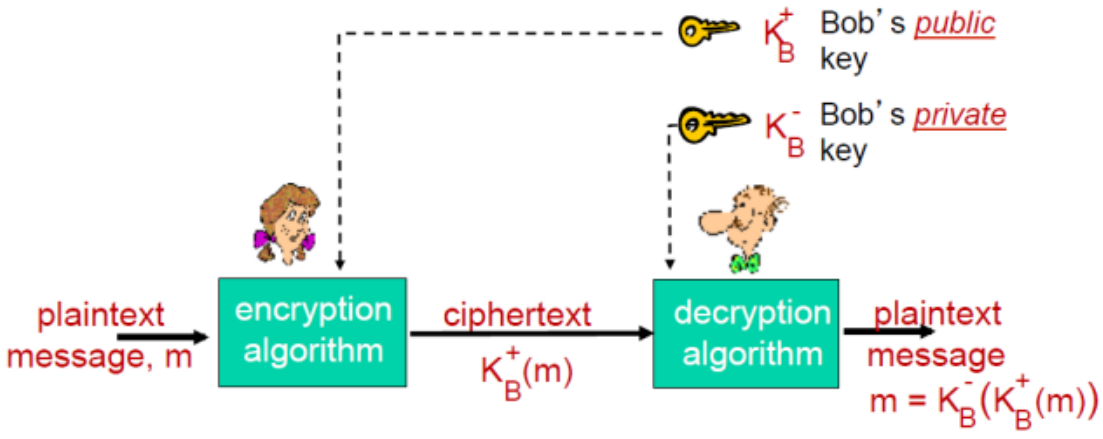
Bob sends digitally signed message:



Alice verifies signature, integrity of digitally signed message:



Public key cryptography

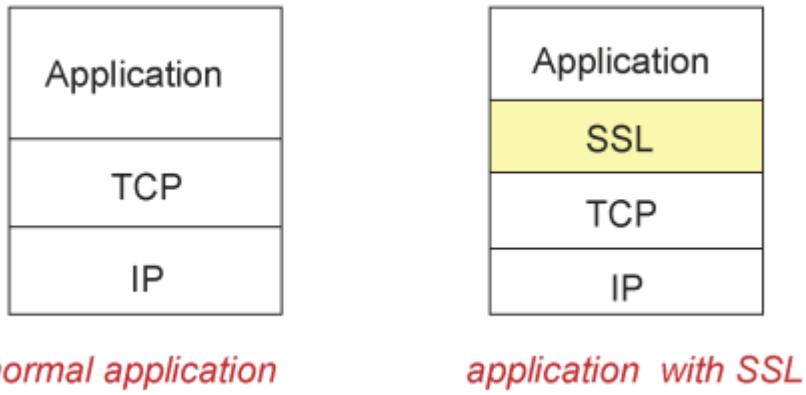


public key를 믿을 수 있어야 함 !

→ 인증기관에 private key로 보관되어 있음

인증기관의 key → browser에 들어가있음

SSL and TCP/IP

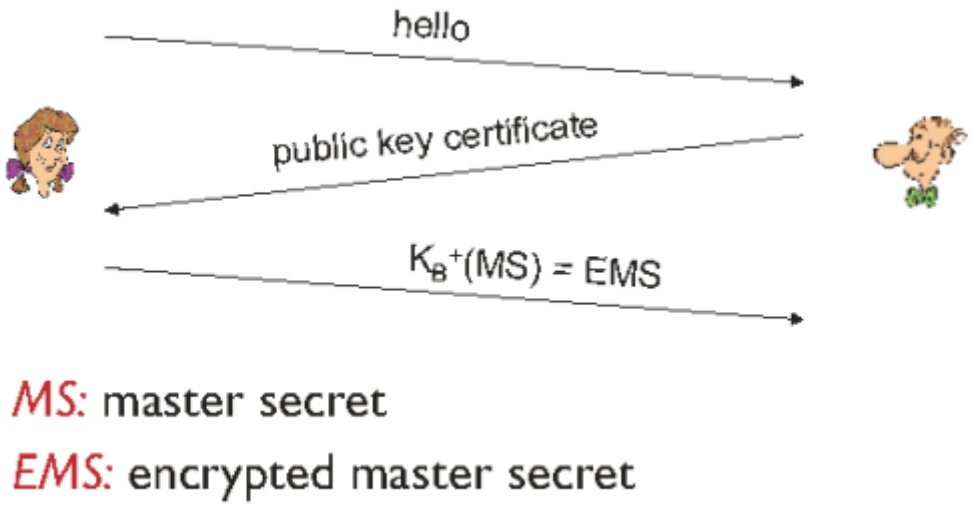


HTTPS : HTTP를 SSL을 사용해 socket을 내려보냄

Toy SSL : a simple secure channel

- *handshake:* Alice and Bob use their certificates, private keys to authenticate each other and exchange shared secret
- *key derivation:* Alice and Bob use shared secret to derive set of keys
- *data transfer:* data to be transferred is broken up into series of records
- *connection closure:* special messages to securely close connection

handshake



연결이 되어있어야함.

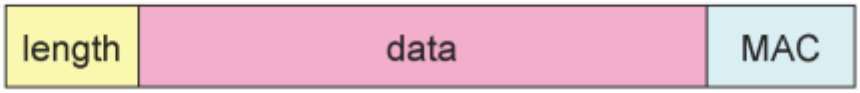
공개키를 보내주면 인증기관을 통해 확인.

secret key를 공개키로 암호화해서 전송

Secret key가 유출되었을 때 피해를 최소화 하기 위해, secret key는 역할에 따라 4개 생성

data records

- why not encrypt data in constant stream as we write it to TCP?
  - where would we put the MAC? If at end, no message integrity until all data processed.
  - e.g., with instant messaging, how can we do integrity check over all bytes sent before displaying?
- instead, break stream in series of records
  - each record carries a MAC
  - receiver can act on each record as it arrives
- issue: in record, receiver needs to distinguish MAC from data
  - want to use variable-length records



여기서의 MAC - 보안에서 MAC

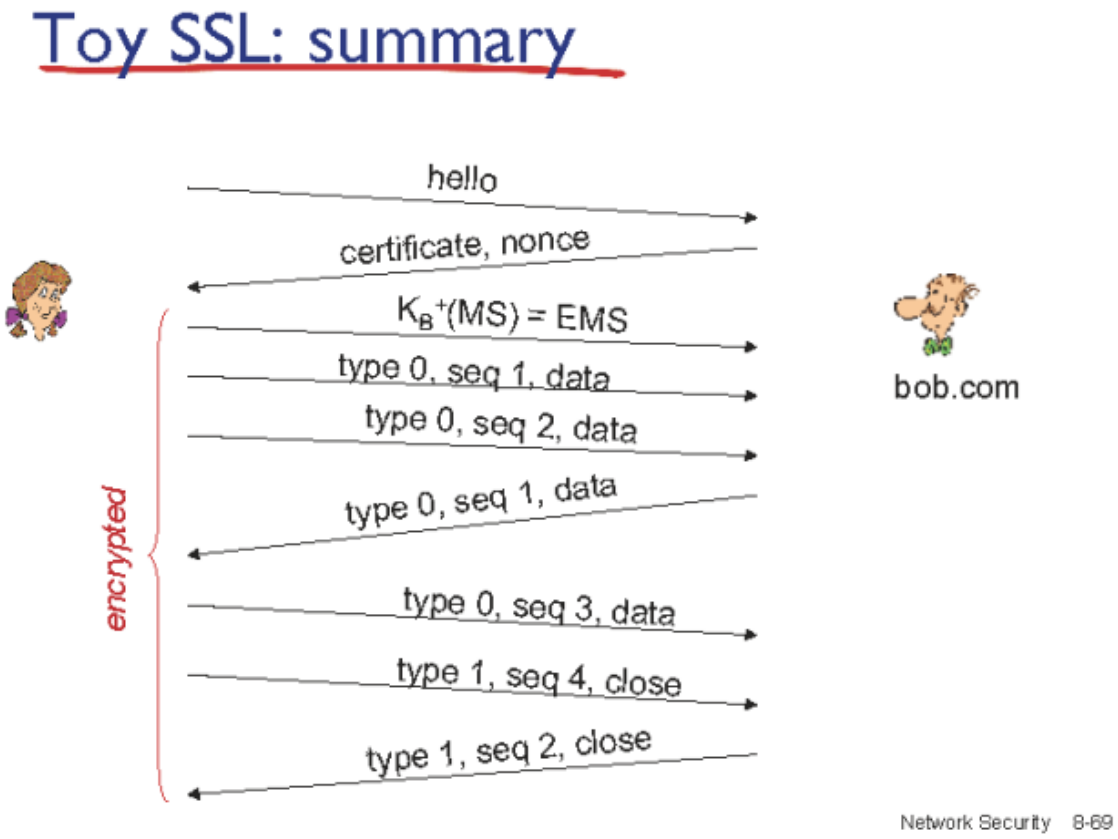
MAC (Message Authentication Code) 메시지 인증 코드는 메시지의 인증에 쓰이는 작은 크기의 정보

어느 서버로 이동하는지는 확인할 수 있지만 메세지 내용은 확인할 수 없음

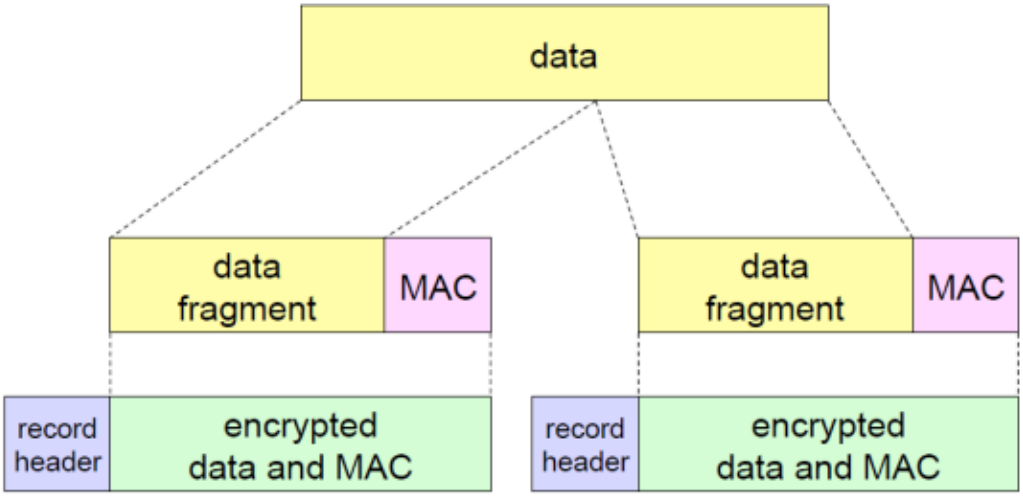
H(data|Kcode|Seq|Type)

순서를 바꿀 수 있으므로 Seq 번호 붙임

data전송이 끝났는지 확인 해야함 - Type에 따라 끝났는지 확인 (0/1)



SSL record protocol



record header: content type; version; length

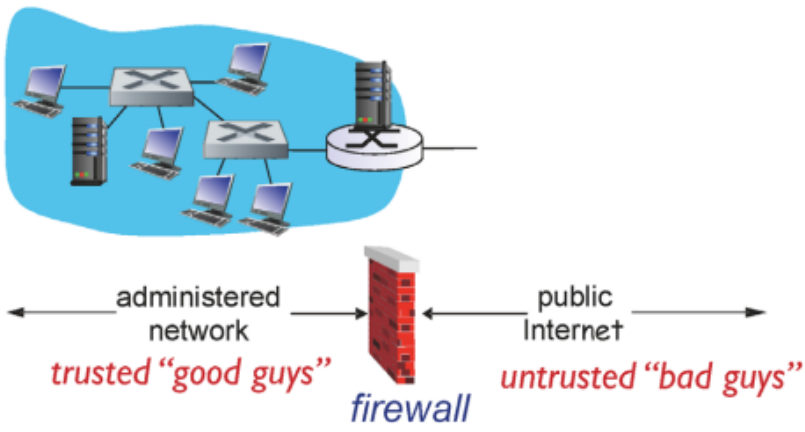
MAC: includes sequence number, MAC key M_x

fragment: each SSL fragment 2^{14} bytes (~16 Kbytes)

Firewalls

firewall

isolates organization's internal net from larger Internet, allowing some packets to pass, blocking others



firewall : 외부로 나가고 들어오는 packet을 중간에 검사

웬만한 곳에서 다 사용

Policy	*Firewall Setting*
No outside Web access.	Drop all outgoing packets to any IP address, port 80
No incoming TCP connections, except those for institution's public Web server only.	Drop all incoming TCP SYN packets to any IP except 130.207.244.203, port 80
Prevent Web-radios from eating up the available bandwidth.	Drop all incoming UDP packets - except DNS and router broadcasts.
Prevent your network from being used for a smurf DoS attack.	Drop all ICMP packets going to a "broadcast" address (e.g. 130.207.255.255).
Prevent your network from being tracerouted	Drop all outgoing ICMP TTL expired traffic

Access Control Lists

❖ *ACL:* table of rules, applied top to bottom to incoming packets: (action, condition) pairs

action	source address	dest address	protocol	source port	dest port	flag bit
allow	222.22/16	outside of 222.22/16	TCP	> 1023	80	any
allow	outside of 222.22/16	222.22/16	TCP	80	> 1023	ACK
allow	222.22/16	outside of 222.22/16	UDP	> 1023	53	---
allow	outside of 222.22/16	222.22/16	UDP	53	> 1023	----
deny	all	all	all	all	all	all