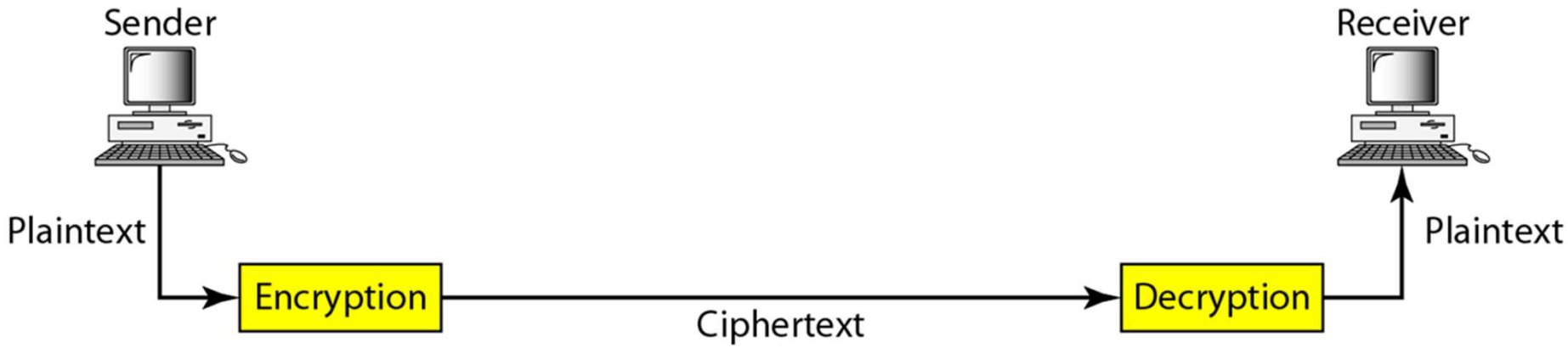
Cryptography and Network Security

Cryptography

Cryptography

- We want to send messages from sender to receiver safely
  - Eavesdroppers cannot understand the message



Cryptography-related vocabulary

- Plaintext: text before encryption
- Ciphertext: encrypted text

- Encryption: making text unreadable
- Decryption: recover unreadable text so that it is readable

- Key
  - number (or set of numbers) used to encrypt text
  - key is needed to decode ciphertext

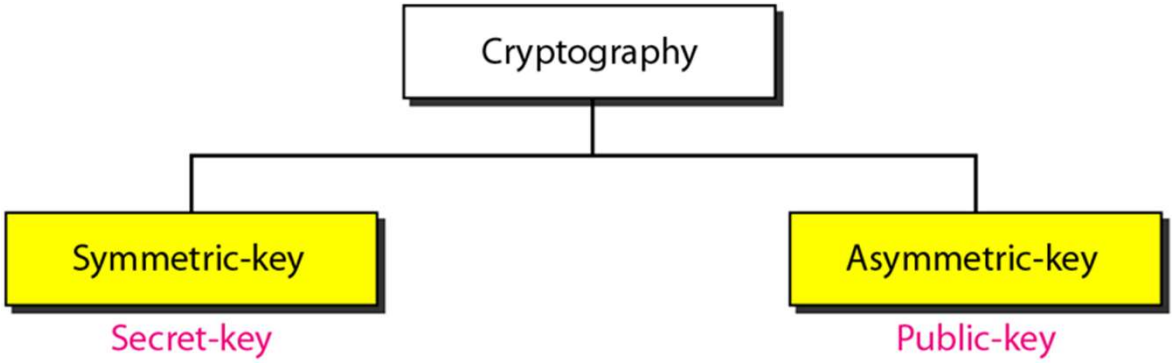
Cryptography algorithm

- Symmetric-key algorithm
  - the same key is used for encryption/decryption

    암호화한 키와 복호화한 키가 같다

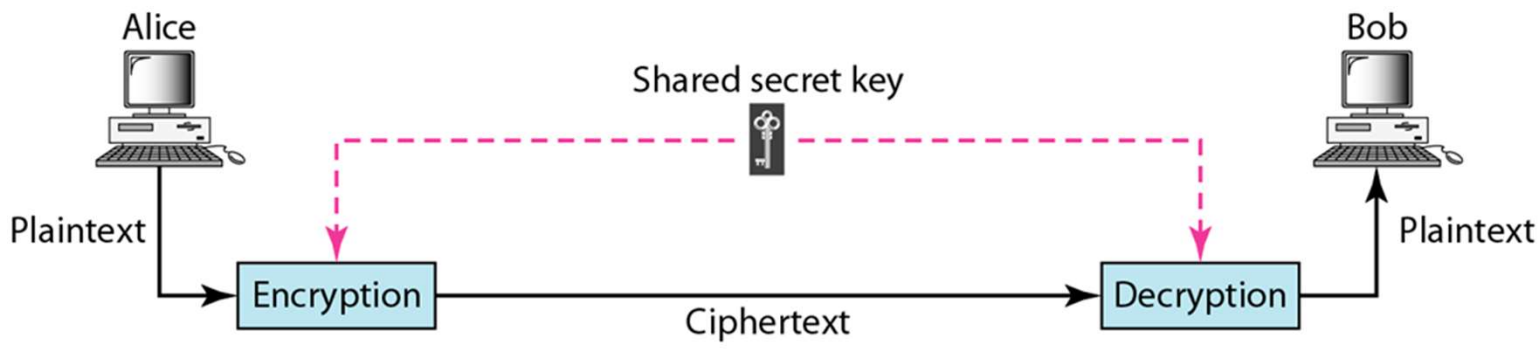
- Asymmetric-key algorithm

  암호화한 키와 복호화한 키가 다르다
  - different keys are used for encryption/decryption



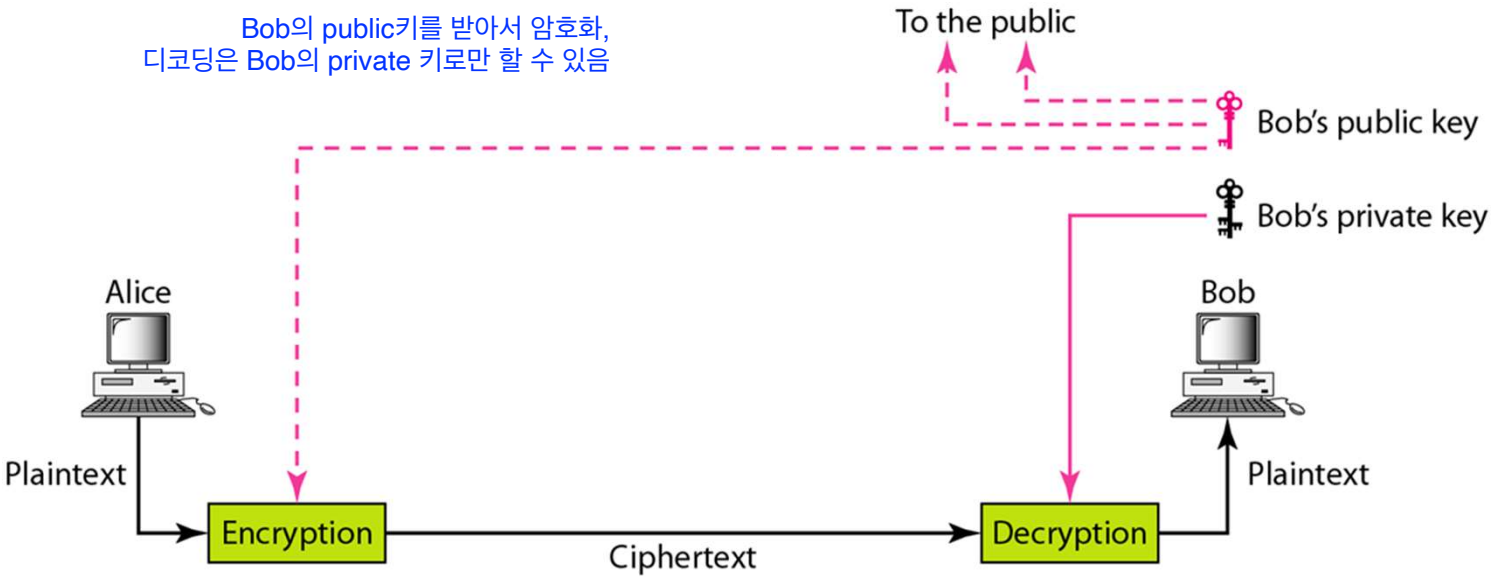
Symmetric-key Cryptography

- Alice (sender) and Bob (receiver) shares a secret key
- Alice encrypts her message using the key
- Alice sends the encrypted message to Bob
- Bob uses key to decode the message
- Eve (eavesdropper) cannot read the message without the key



Asymmetric-key Cryptography

- Bob has a public/private key pair
- Bob opens the public key to public, but secretly holds the private key
- Alice encrypts her message using Bob's public key
- The message can be decrypted only by using Bob's private key → only Bob can open this message



Symmetric key vs. Asymmetric key

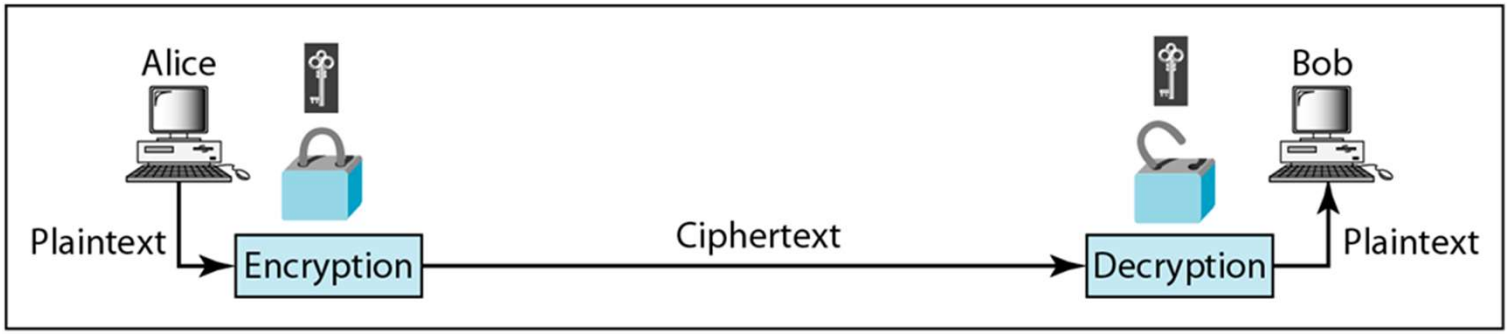
- Symmetric key
  - the same key for encryption/decryption

- Asymmetric key
  - two different keys for encryption and decryption

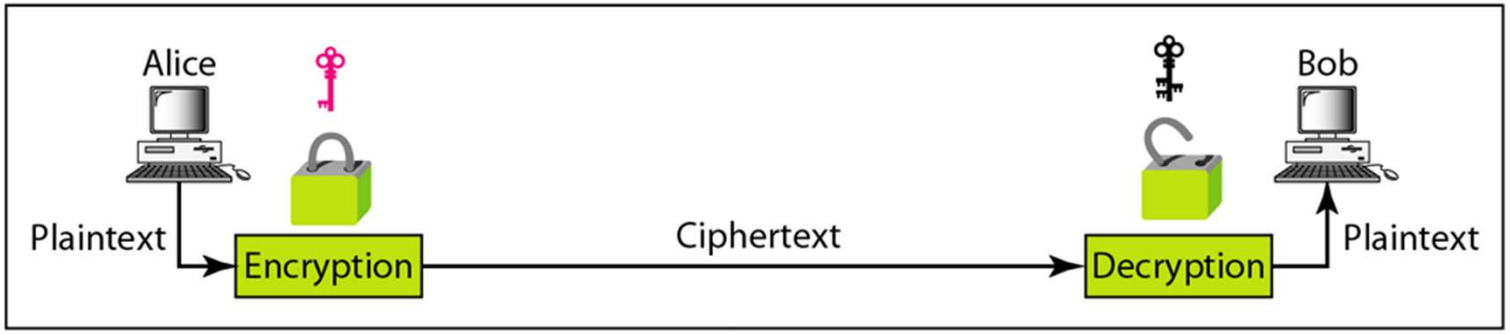
Symmetric-key cryptography

Asymmetric-key cryptography

Symmetric key vs. Asymmetric key



a. Symmetric-key cryptography

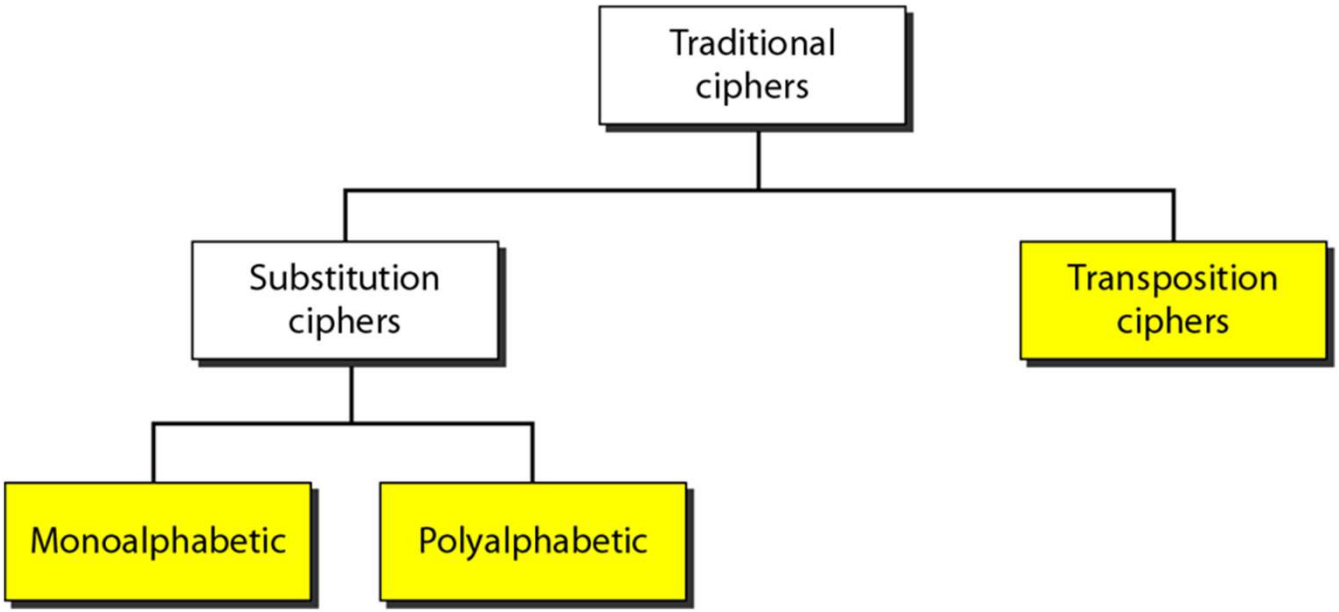


b. Asymmetric-key cryptography

Symmetric-key Cryptography

Traditional ciphers

- classic methods
  - even used in Roman empire by Julius Caesar



Substitution ciphers

- Each letter in the plaintext is substituted with another letter
- mono-alphabetic substitution
  - If two letters are the same in plaintext, it is also the same in ciphertext

Plaintext: HELLO
Ciphertext: KHOOR

- poly-alphabetic substitution
  - Two letters that are same in plaintext can be different in ciphertext

Plaintext: HELLO
Ciphertext: ABNZF

Substitution ciphers

- Shift cipher
  - the basic form of substitution cipher
  - also called “Caesar cipher”
  - shift alphabet in either direction to make the ciphertext
  - e.g.: move back 3 letters
  - A → D, B → E, ..., W → Z, X → A, Y → B, Z → C

Substitution ciphers

- Example: make ciphertext when the plaintext is "HELLO", and key = 15 (move back 15 letters)

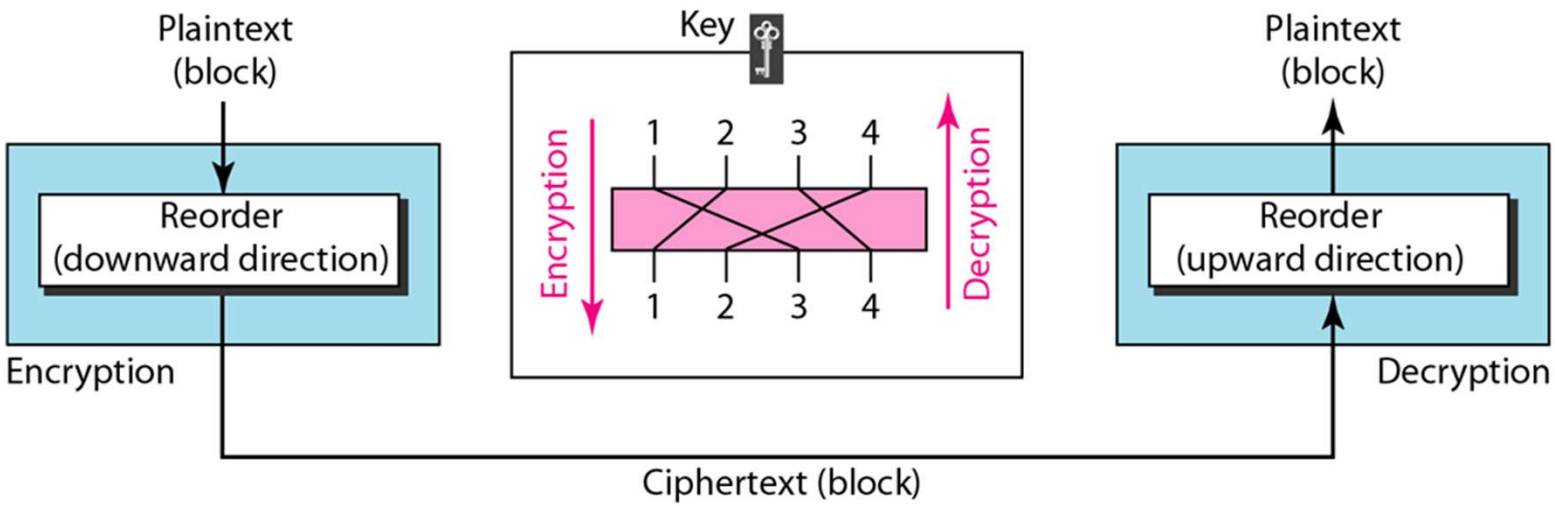
Substitution ciphers

- When key = 15, decode the ciphertext “WTAAD”.

Transposition ciphers

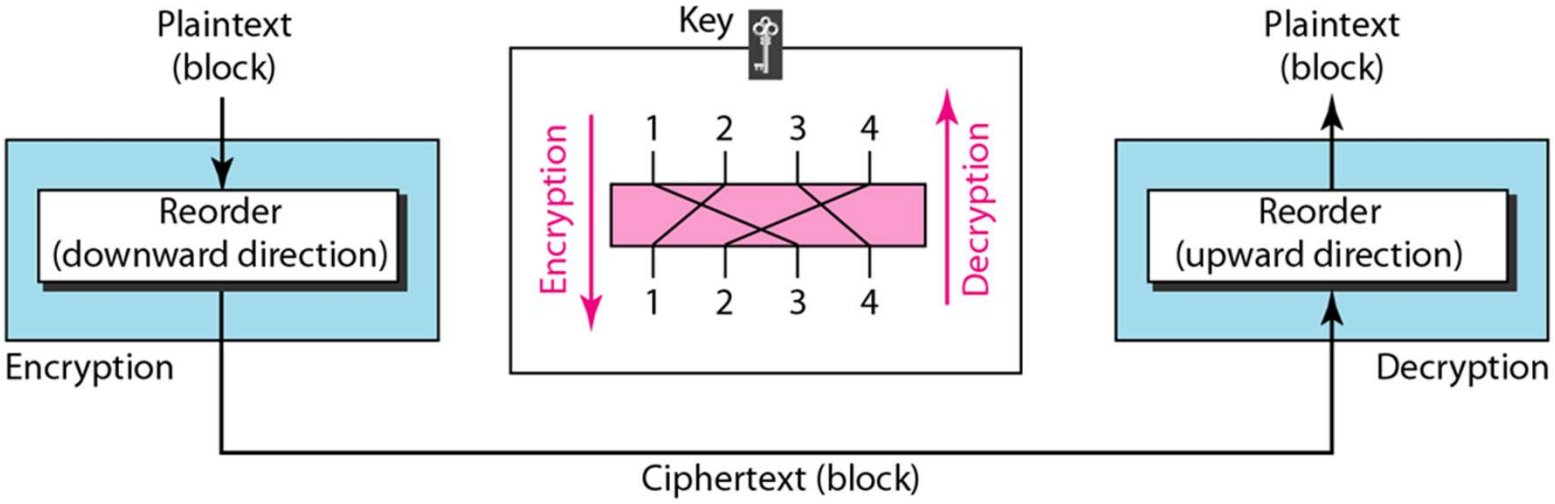
- Instead of substitution, change position of the letters

- Example
  - divide message into four blocks
  - reorder blocks as in the box below



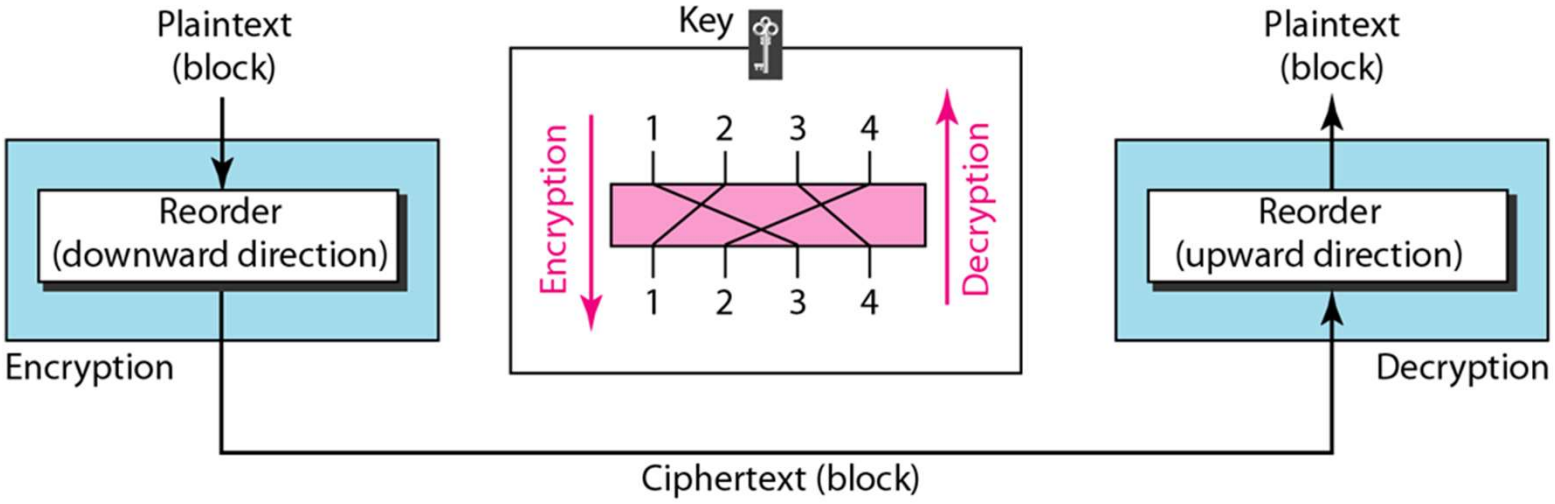
Transposition ciphers

- Make ciphertext from plaintext “HELLOMYDEAR” using the transposition cipher below.



Transposition ciphers

- The ciphertext is “ELHLMDOYAZER”. What is the plaintext, if the ciphertext was generated using transposition ciphers below?



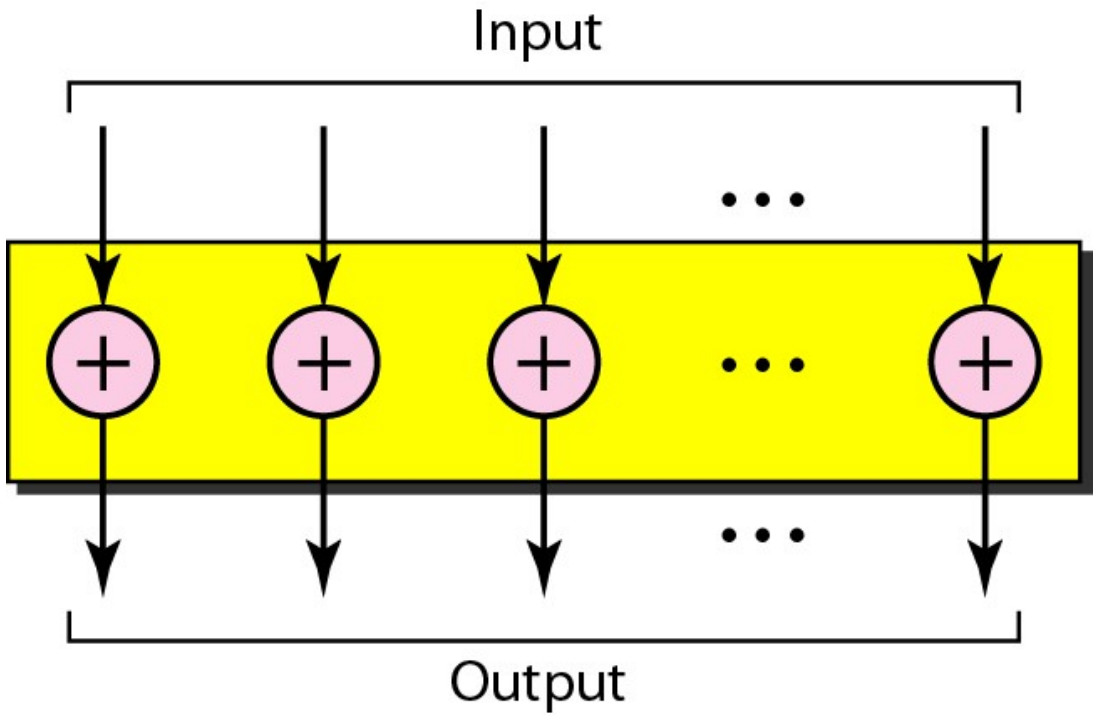
Modern ciphers

- character-based → bit-based

- need to encrypt not only text data, but also multimedia

XOR cipher

- Apply bitwise XOR to plaintext and key to make the ciphertext



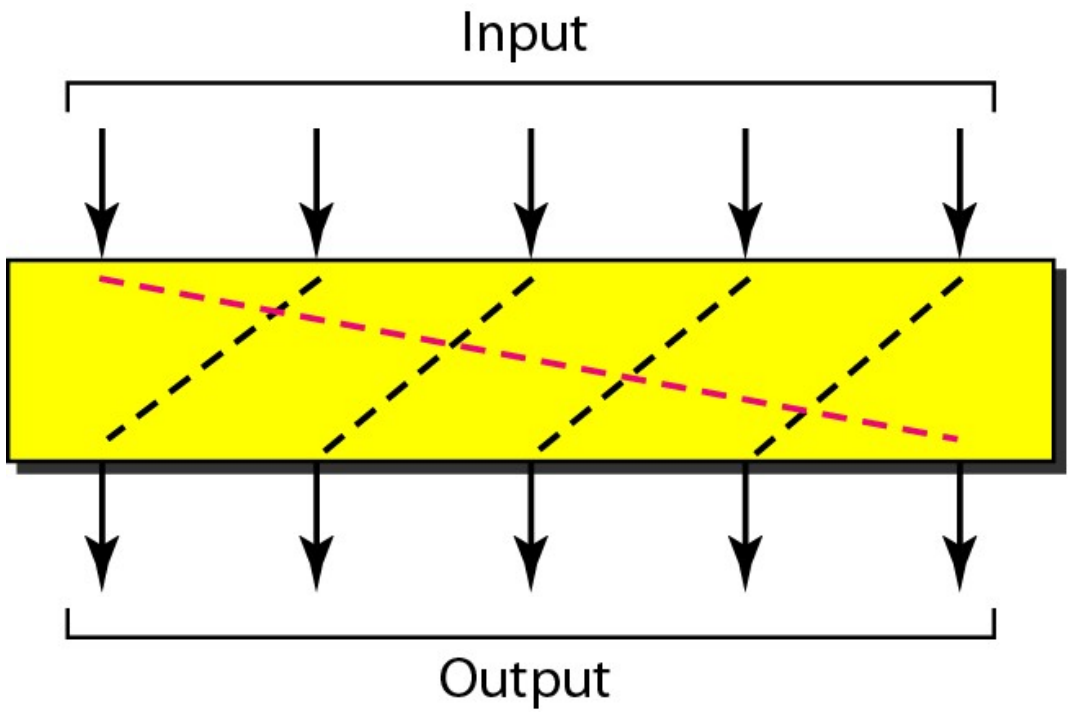
XOR cipher

- Plaintext: 01010111 01101101 01101011 01101011
  - “Wiki”
- Key: 11110011
- Ciphertext: 10100100 10011010 10011000 10011010
- To decode, apply XOR on ciphertext and the key

Rotation cipher

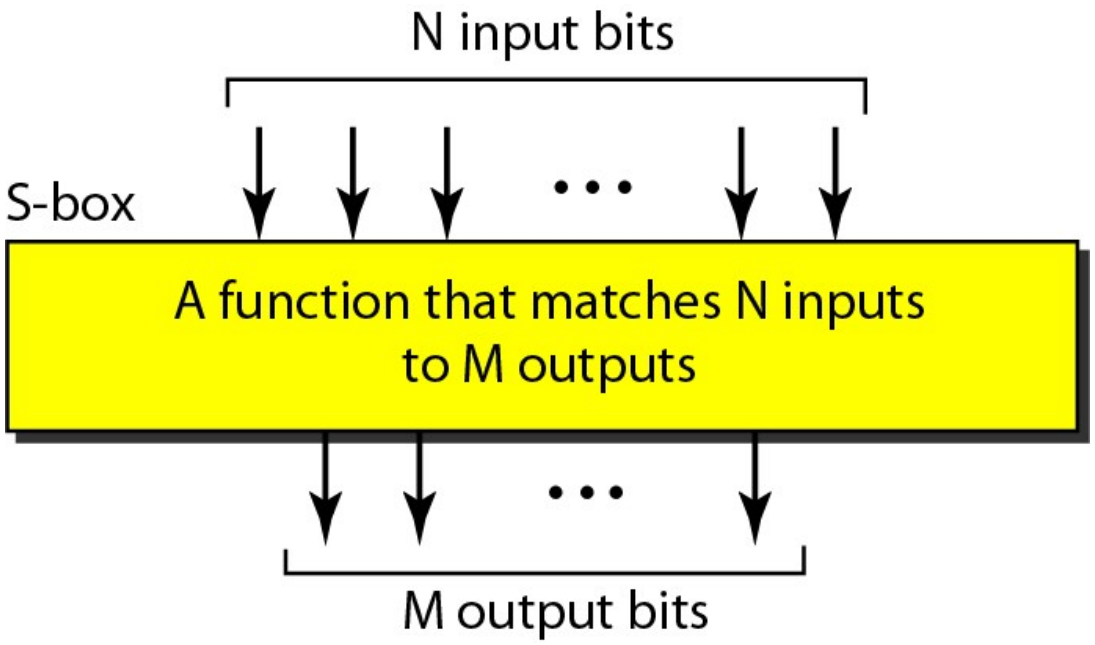
- Input bit is shifted to left or right to make the ciphertext

- A type of transposition cipher



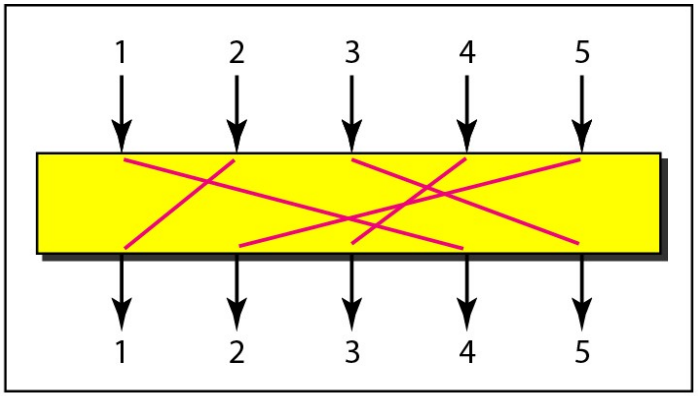
S-box

- S-box (substitution box): inputs N bits, outputs M bits

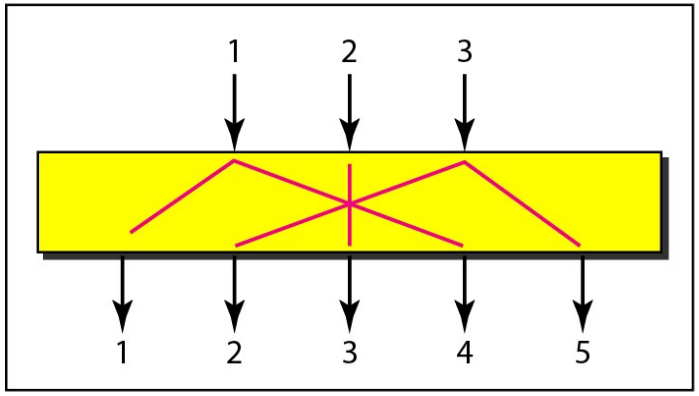


P-box

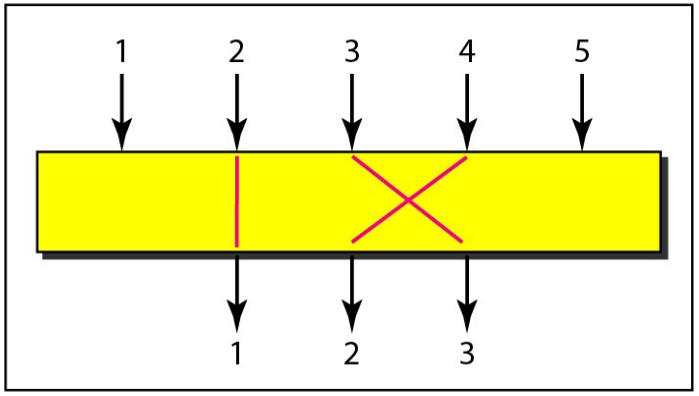
- P-box (permutation box): Transposes input.
  - straight permutation, expansion permutation, compression permutation, etc.



a. Straight



b. Expansion



c. Compression

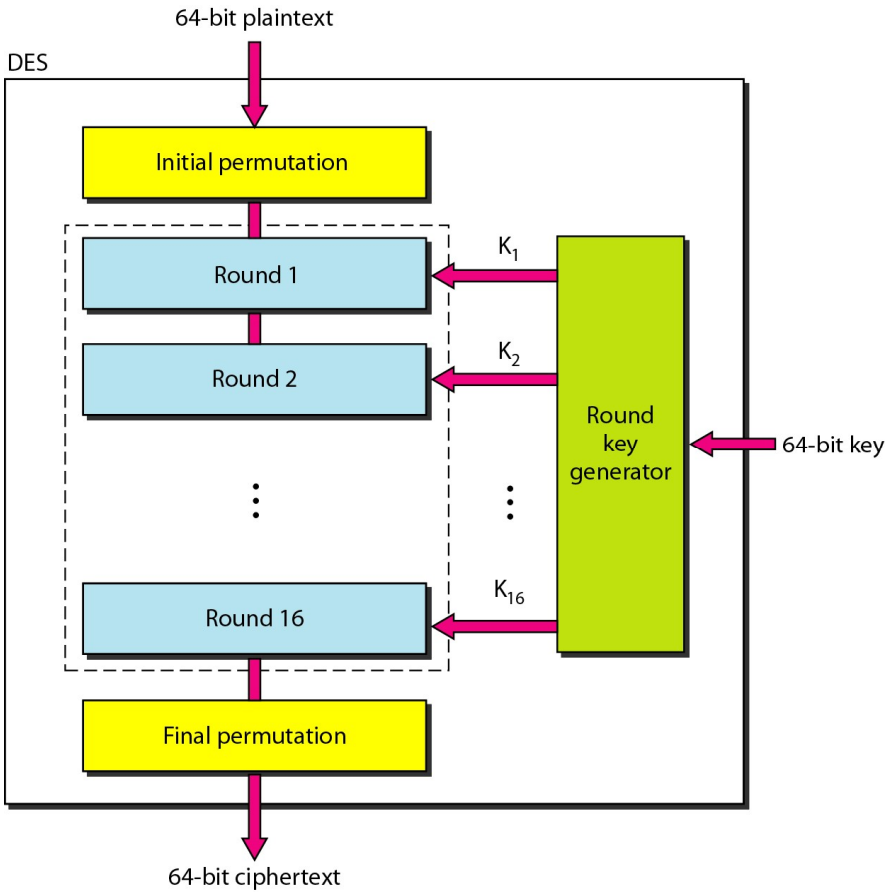
DES

- Data Encryption Standard
- a representative symmetric key encryption algorithm
- designed by IBM
- Key = 56 bits (+ 8 parity bits)

- A type of round cipher
  - round cipher: encrypt a message many times to improve safety

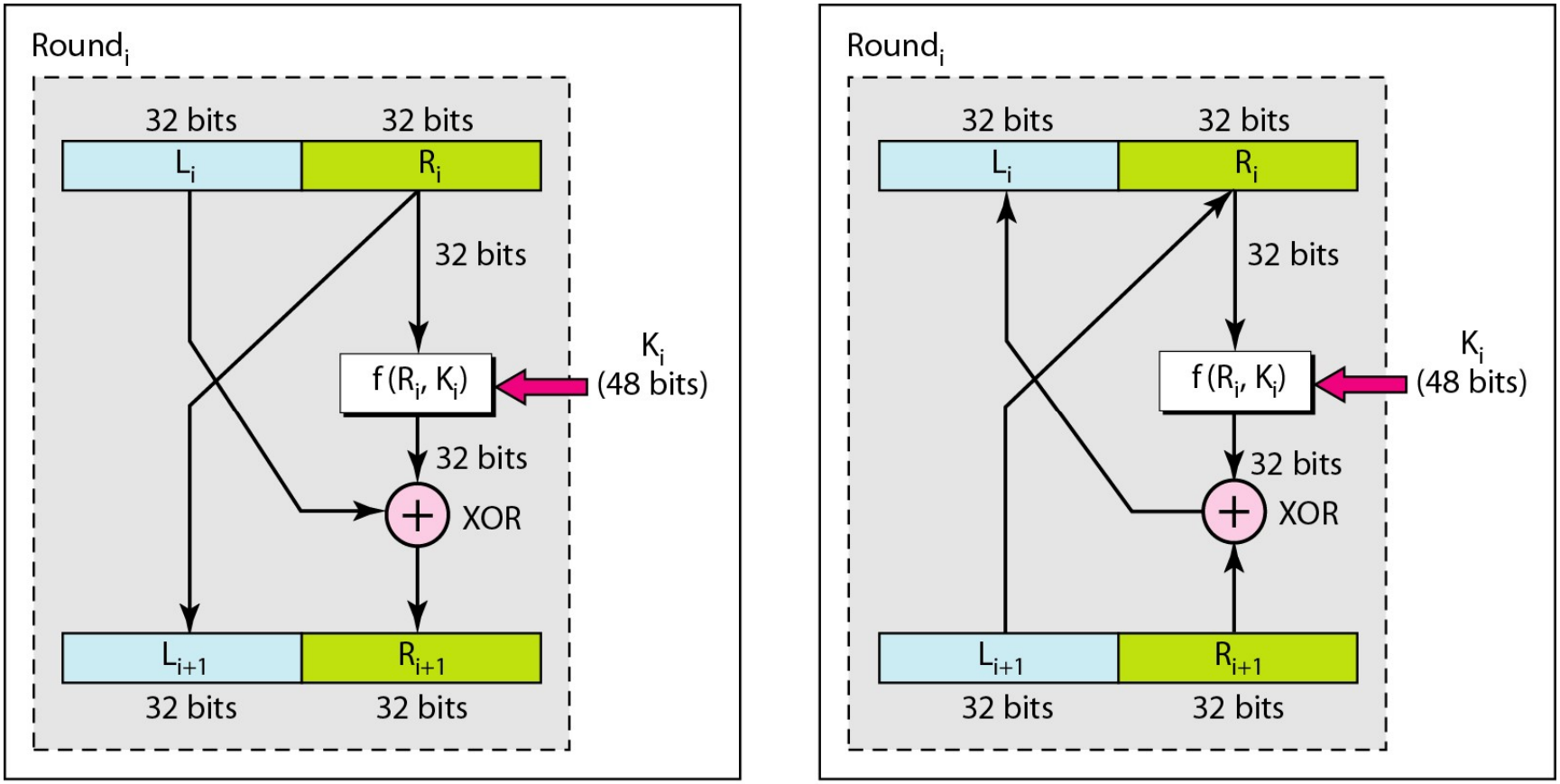
DES

- DES flow diagram
- The round key generator makes sixteen 48-bit subkeys from a single 64-bit string (56 bits key + 8 parity bits)



DES

- a round

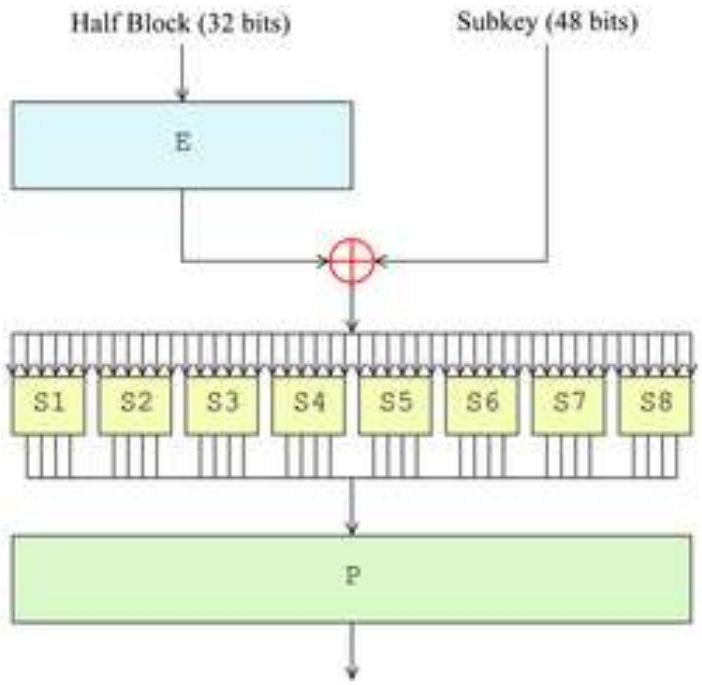


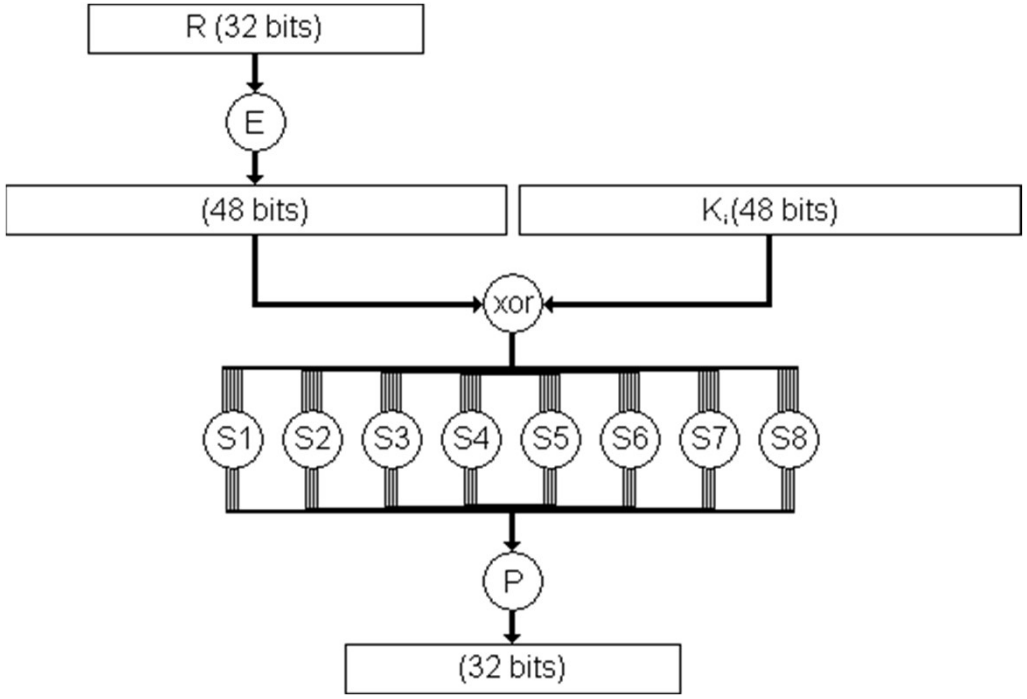
a. Encryption round

b. Decryption round

DES

- Feistel function





DES

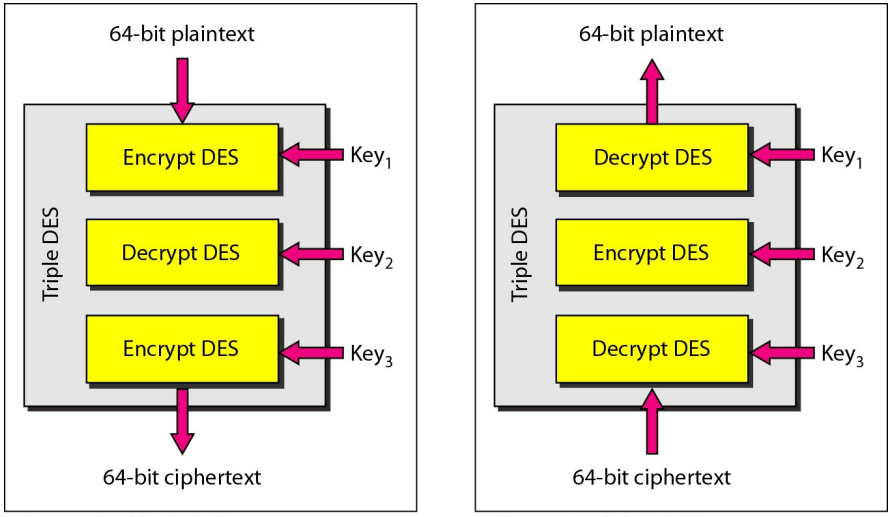
- In 1977, selected as the standard encryption method in US
- 56-bit key

컴퓨터 속도가 빨라져서 안씀

- Attack against DES: Brute-force attack
  - tries all possible keys
  - 2^56 = 72,057,594,037,927,936
- In 1998, Supercomputer “Deep Crack” finds the key in 56 hrs
- 6 months later, Deep Crack and distributed.net together finds the key in 22 hrs and 15 minutes
- In 1999, DES is still the standard encryption method, but Triple DES (3DES) becomes a recommended method 세번 암호화

Triple DES (3DES)

- Since 56-bit key is too short, encrypt three times in order to increase key size
  - If Key1 = Key3, the key is 112 bits
  - If we use three different keys, the key is 168 bits
  - Encrypt → Decrypt → Encrypt method:
    - for compatibility with DES
    - If Key1=Key2=Key3, it is the same with DES



a. Encryption Triple DES

b. Decryption Triple DES

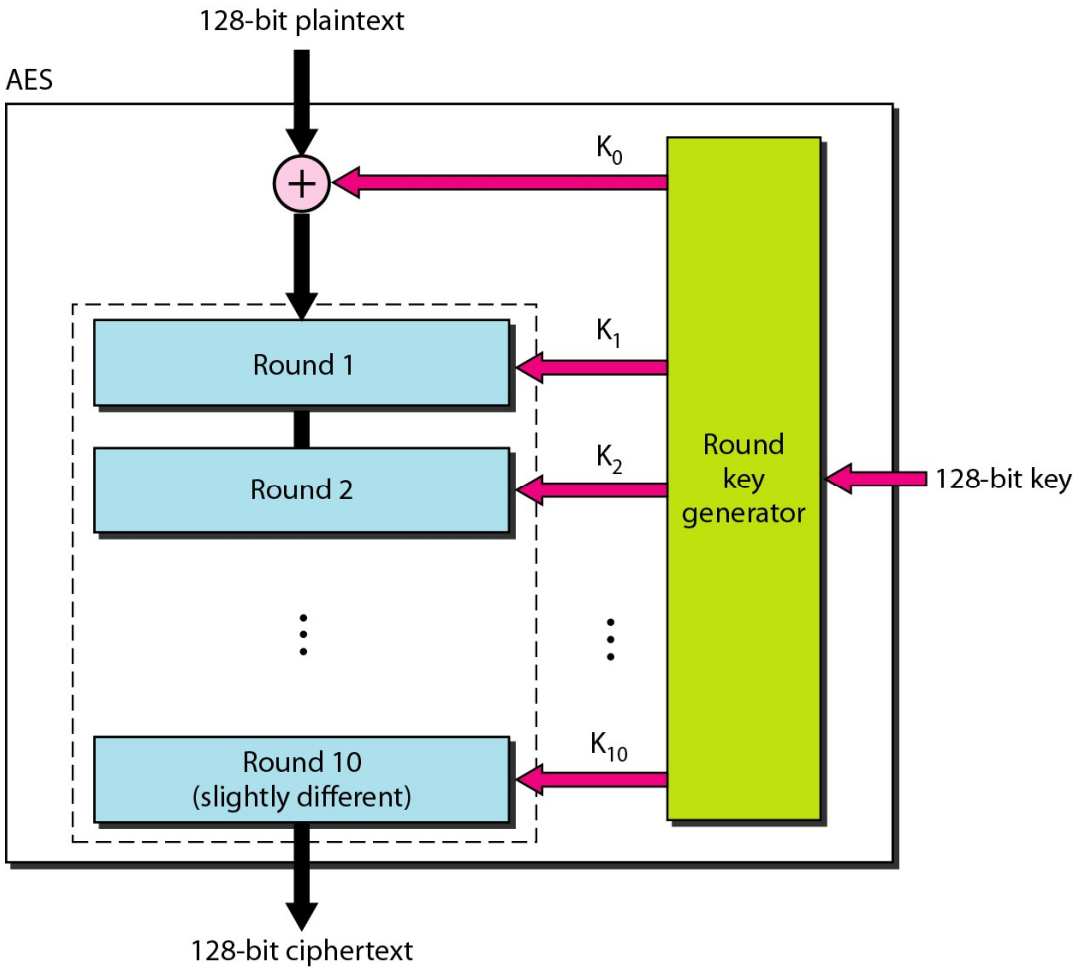
AES

- Advanced Encryption Standard
- Similar to 3DES, developed to make up for short keys of DES
- In 2002, AES is selected as the standard encryption method
- key size: 128, 192 or 256

Size of Data Block	*Number of Rounds*	*Key Size*
128 bits	10	128 bits
	12	192 bits
	14	256 bits

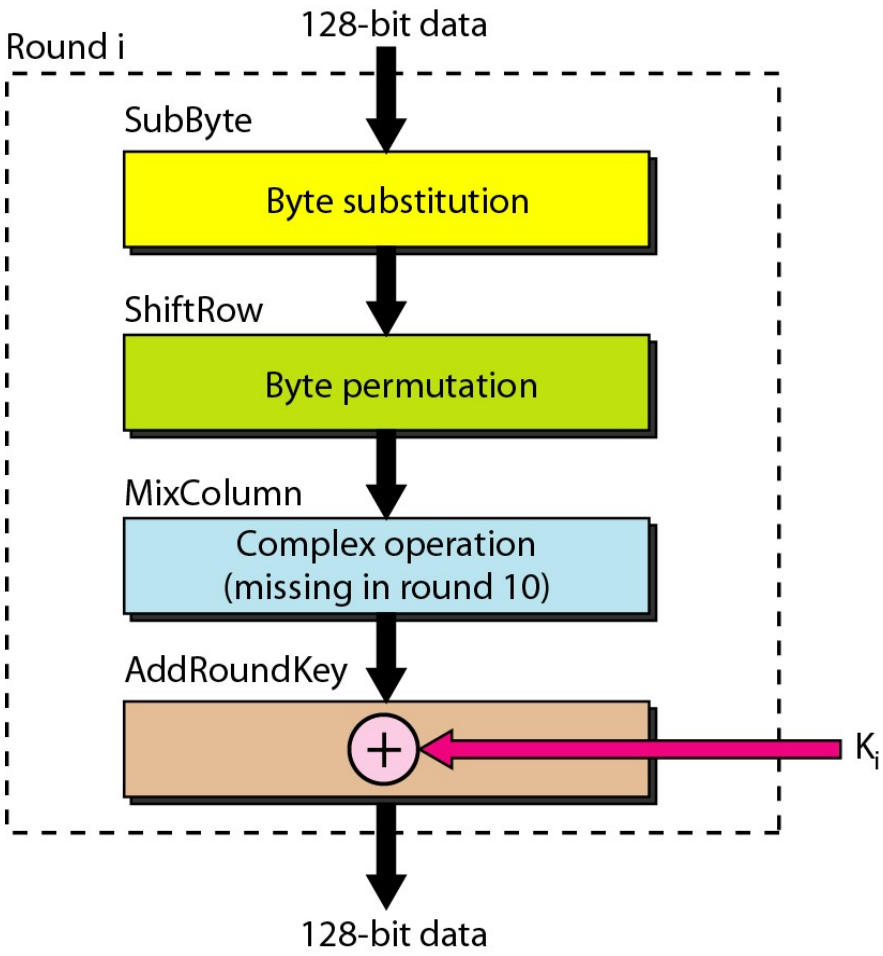
AES

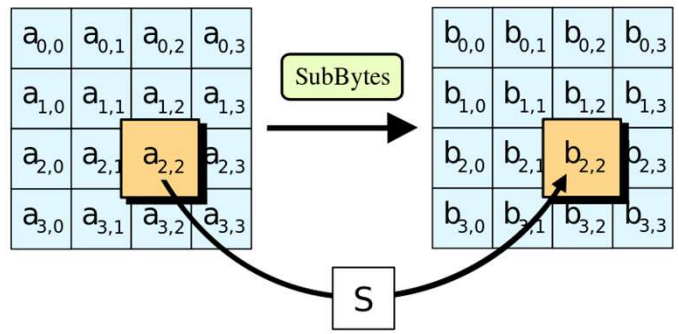
- AES flow

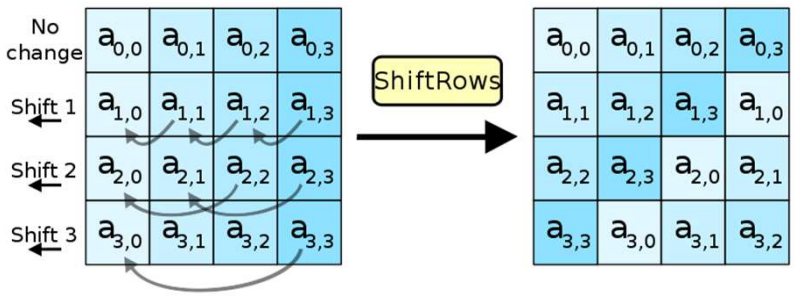

128-bit plaintext
AES
K_0
K_1
K_2
K_{10}
Round 1
Round 2
Round 10
(slightly different)
Round
key
generator
128-bit key
128-bit ciphertext

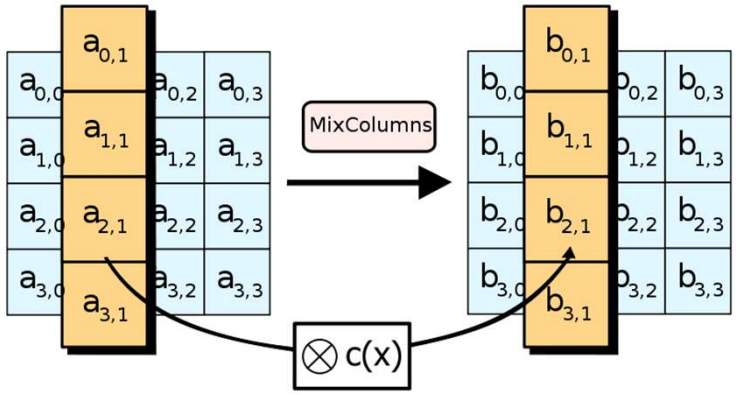
AES

- AES round





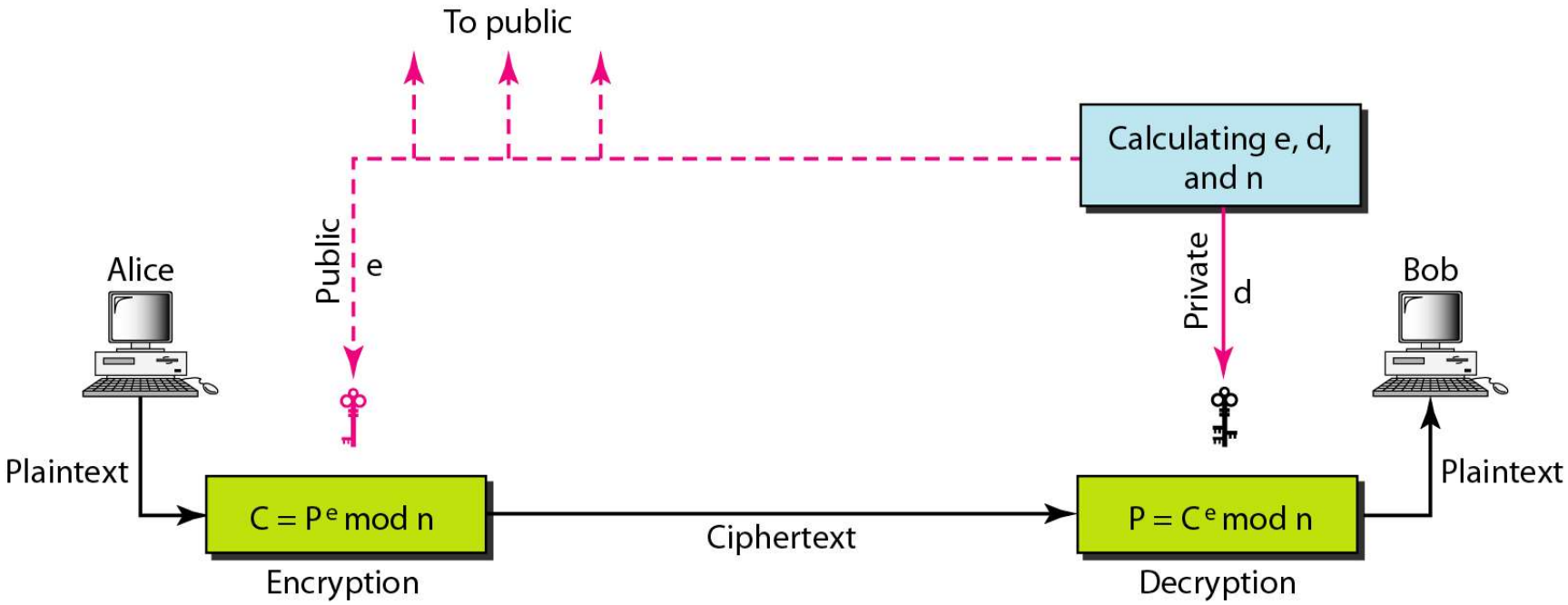




Asymmetric-key Cryptography

RSA

- Rivest, Shamir, and Adleman → RSA
- public key: e
- private key: d



RSA: key generation

- Bob makes public key and private key
  - Select two <u>prime numbers</u>, p and q
  - n = p x q
    - n is the "modulus" for encryption and decryption
  - $\Phi = (p-1) \times (q-1)$
  - select a random number e
  - select d so that $d \times e = 1 \bmod \Phi$ e * d mod pi = 1
    - e.g.) if $\Phi = 60$, $1 \bmod \Phi = 1, 61, 121, 181, \ldots$

- Bob opens e and n: public key
- Bob does not open d and Φ: private key

RSA: encryption

- If Alice wants to send a message to Bob, Alice computes the ciphertext as follows

- $C = P^e (\mathrm{mod}\ n)$

RSA: decryption

- Bob knows d, so it uses d to decode the ciphertext as follows.

- $P = C^d (\mathrm{mod}\ n)$

- The message cannot be decode without d. Thus, only Bob can decode this message

RSA: example

- Bob generates keys
  - selects $p = 7$, $q = 11$
  - $n = 7 \times 11 = 77$
  - $\Phi = 6 \times 10 = 60$

  - select $e = 13$
  - if $d = 37$, condition is satisfied ($13 \times 37 = 481$, $481 \bmod 60 = 1$)

  - publish e and n

RSA: example

- Alice wants to send message to Bob

- Suppose plaintext: 5
- Ciphertext is computed as follows.

- $C = 5^{13} \bmod 77 = 26$

- ciphertext: 26
- send this to Bob

RSA: example

- Bob receives 26. He decodes the ciphertext as follows.
- $P = 26^{37} \bmod 77 = 5$
- Thus, the message Alice sent is “5”.

RSA: real example

- p and q are selected among 512-bit prime numbers
  - <u>very big prime numbers</u>

p = 9613034531358350457419158128061542790930984559499621582258315087964794045505647063849125716018034750312098666606492420191808780667421096063354219926661209

q = 12060191957231446918276794204450896001555925054637033936061798321731482148483764659215389453209175225273226830107120695604602513887145524969000359660045617

RSA: 실제 사용 예

- compute n and Φ from p and q

n = 11593504173967614968892509864615887523771457375454144775485526137614788
5408326350817276878815968325168468849300625485764111250162414552339182 9
2716250765677272746009708271412773043496050055634727456662806009992403 7
1029914244722922157727985317270338393813346926841373276220009666766718 3
183108837342082344437095 3

ϕ = 11593504173967614968892509864615887523771457375454144775485526137614788
5408326350817276878815968325168468849300625485764111250162414552339182 9
2716250765675105423360849291675203448262798811755478765701392344440571 6
9895817281960982263610754672118646121713591073586406140088851702653772 7
7264467341066243857664128

RSA: 실제 사용 예

- select e and find proper d

e = 35535

d = 58008302860037763936093661289677917594669062089650962180422866111380593852
822358731706286910030021710859044338402170729869087600611530620252495988 44
480475682409662470814858171304632406440777048331340108509473852956450719 36
774061197326557424237217617674620776371642076003370853332885321447088595 51
36670294831

- suppose plaintext is P

P = 190708182608182600261904181 9

RSA: 실제 사용 예

- Generate ciphertext. $C = P^e (\mathrm{mod}\ n)$

$$\mathbf{C} = 47530912364622682720636555061054518094237179607049171652323924305445296061319932856661784341835911415119741125200568297979457173603610127821884789274156609048002350719071527718591497518846588863210114835410336165789846796838676373376577746562507928052114814184404814184430812773059004692874248559166462108656$$

- When Bob receives C, he decodes the ciphertext using
- $P = C^d (\mathrm{mod}\ n)$

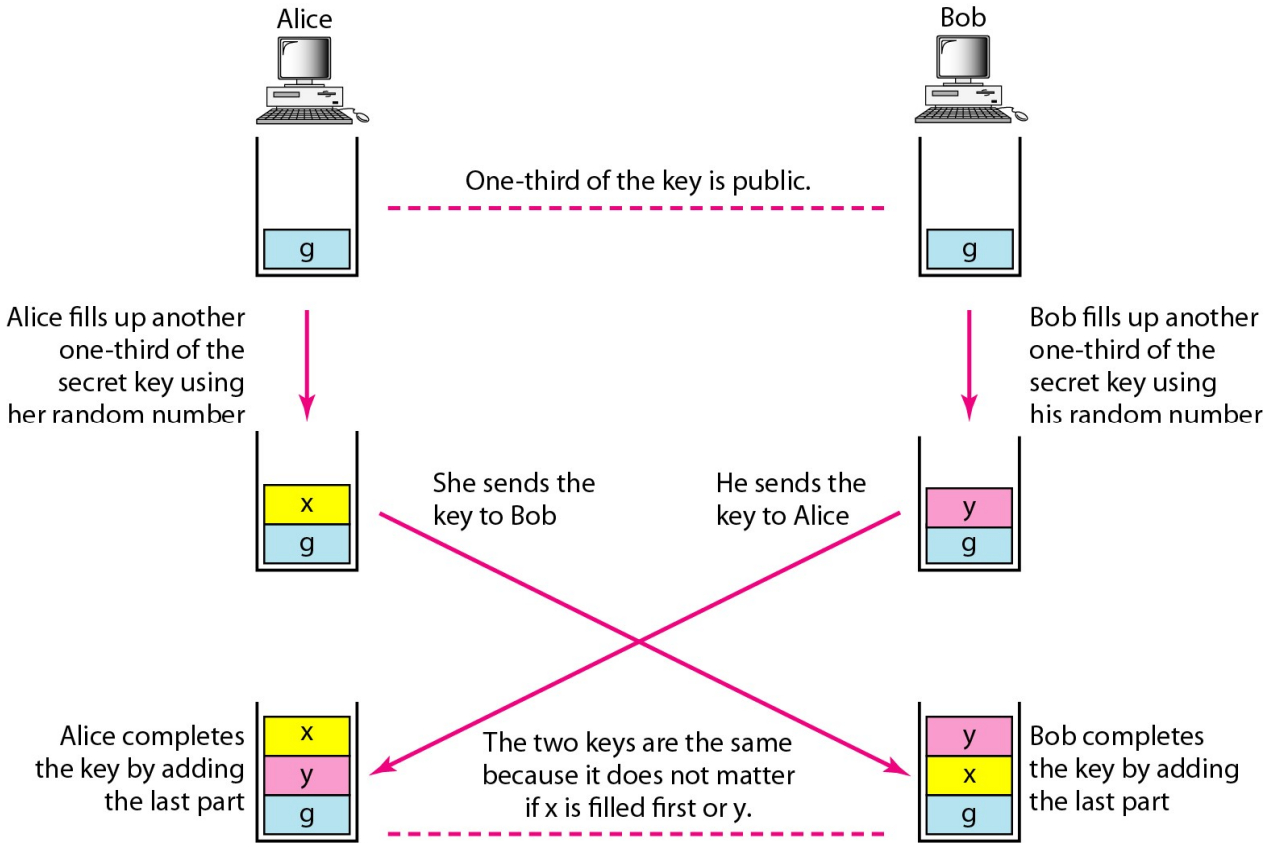
$$\mathbf{P} = 19070818260818260026190418 19$$

RSA

- pros
  - To use symmetric key algorithm, sender and receiver should share a secret key
  - It is not easy to share a secret key secretly
  - RSA does not need the process of sharing a key
- cons
  - complex, thus slow
- usage example
  - Transfer secret key using RSA, and after then use the secret key to communicate using symmetric key cryptography

Diffie-Hellman

- A method to share a secret key
  - point: x nor y is directly sent



Diffie-Hellman

- Alice wants to send Bob its secret key
  - Alice select p and g and send them to Bob. p is a prime number.
  - Alice selects a number x and computes R_1
    - $R_1 = g^x \bmod p$
  - Bob selects a number y and computes R_2
    - $R_2 = g^y \bmod p$
  - Alice sends R_1 to Bob, and Bob sends R_2 to Alice.
  - Alice computes K
    - $K = (R_2)^x \bmod p$
  - Bob computes K
    - $K = (R_1)^y \bmod p$

Diffie-Hellman: example

- suppose g=7 and p=23
- Alice: selects x=3. $R_1 = 7^3 \bmod 23 = 21$.
- Bob: selects y=6. $R_2 = 7^6 \bmod 23 = 4$.
- Alice sends 21 to Bob
- Bob sends 4 to Alice
- Alice computes K: $K = 4^3 \bmod 23 = 18$
- Bob computes K: $K = 21^6 \bmod 23 = 18$

- K is the shared secret key

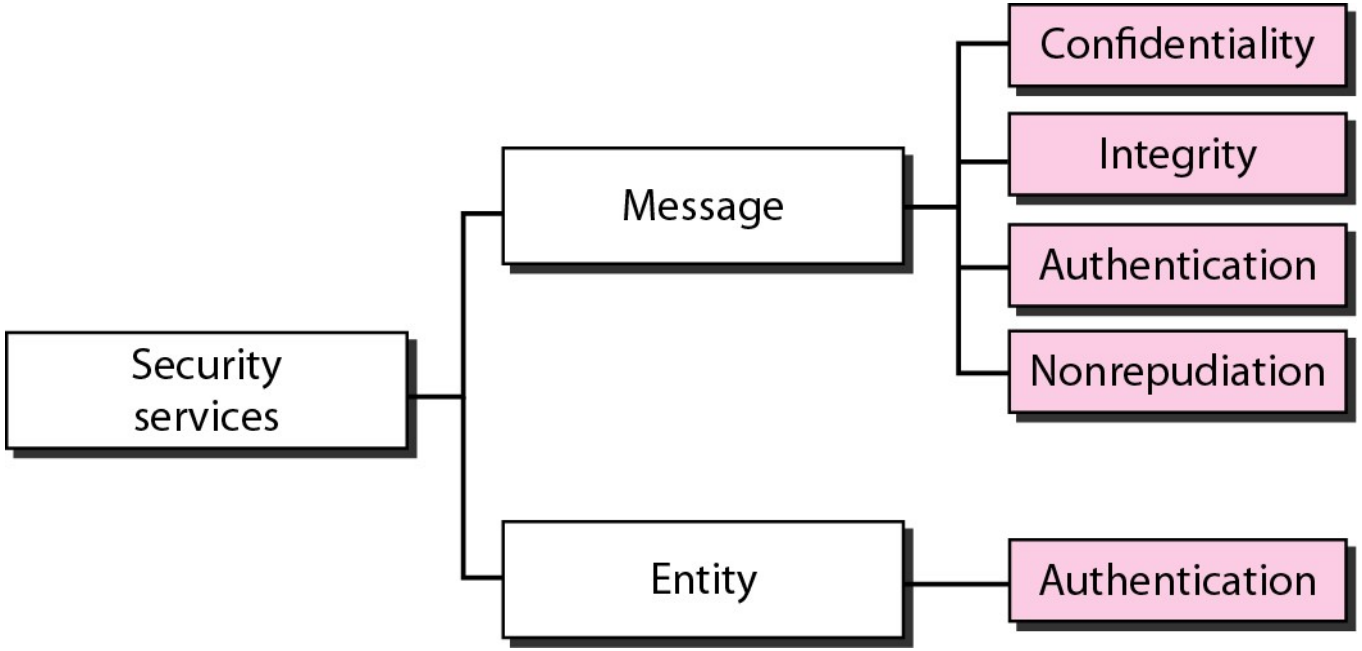
Diffie-Hellman

- Suppose Eve (eavesdropper) overheard R_1 and R_2.

- Eve has to find out x and y to know the secret key, but this is very hard.

Network Security

Security Services



Security Services

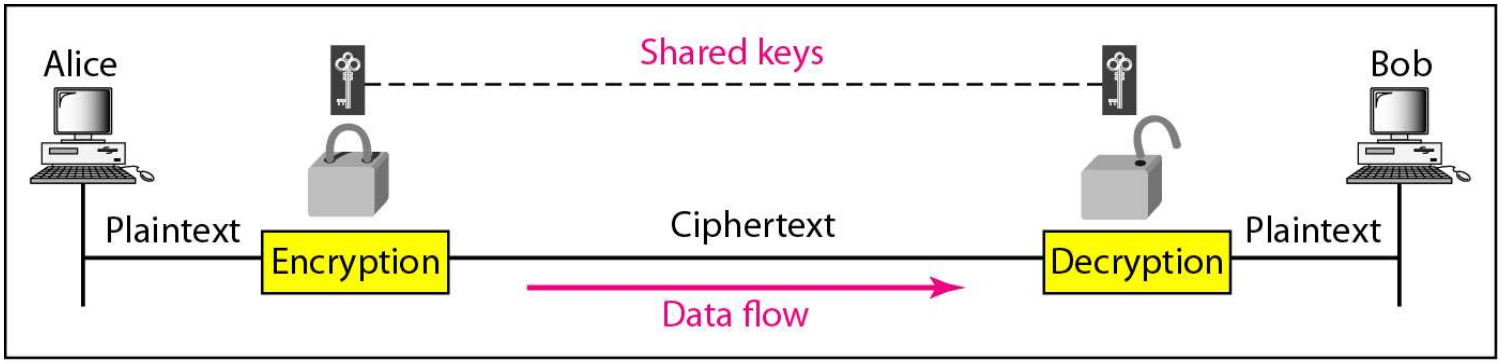
- Message Confidentiality (기밀성)
  - Eavesdropper cannot understand message sent from sender to receiver
- Message Integrity (무결성)
  - Message is not modified before it is received by the receiver
- Message Authentication (인증)
  - Confirm that message is actually sent from the sender and not someone else

Security Services

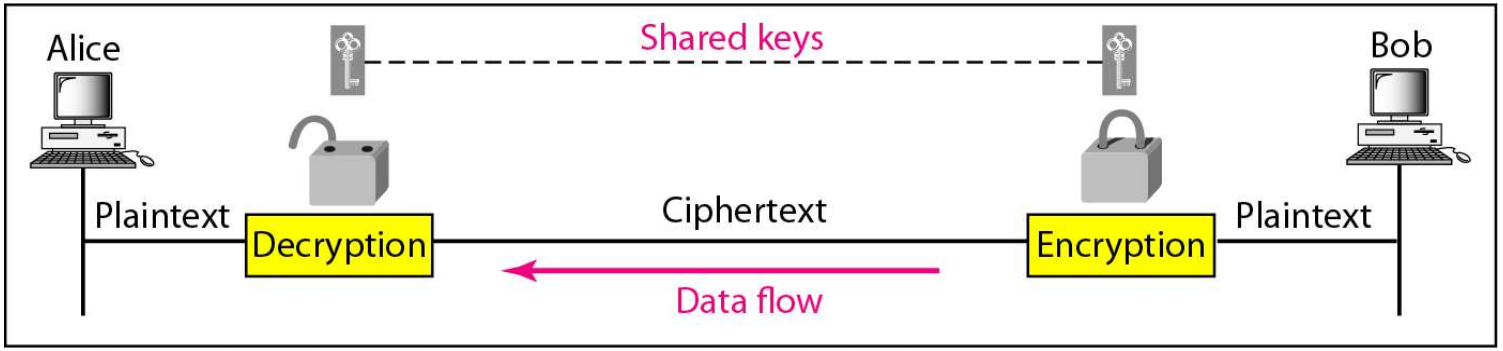
- Message Nonrepudiation (부인 방지)
  - Sender cannot deny that it has sent a message
- Entity authentication (개체 인증)
  - Confirm identity of the user

Message Confidentiality

- confidentiality guarantee using symmetric key
- symmetric key is usually sent using asymmetric key cryptography



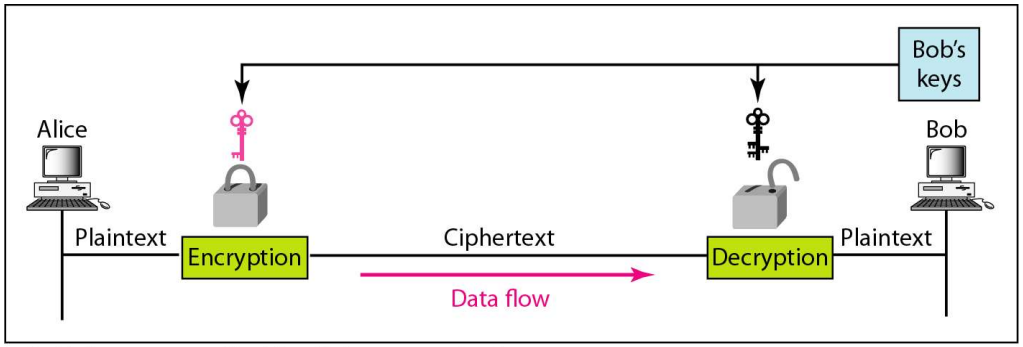
a. A shared secret key can be used in Alice-Bob communication



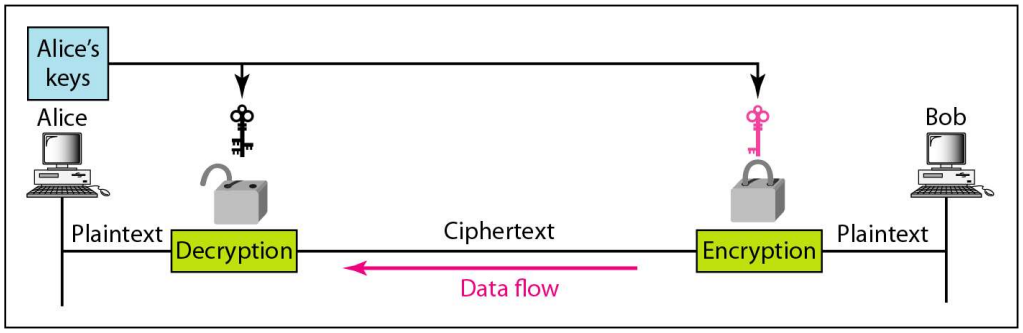
b. A different shared secret key is recommended in Bob-Alice communication

Message Confidentiality

- confidentiality guarantee using asymmetric key
- sender and receiver both need to generate public/private key pair
- Due to complexity, not good for long messages



a. Bob's keys are used in Alice-Bob communication



b. Alice's keys are used in Bob-Alice communication

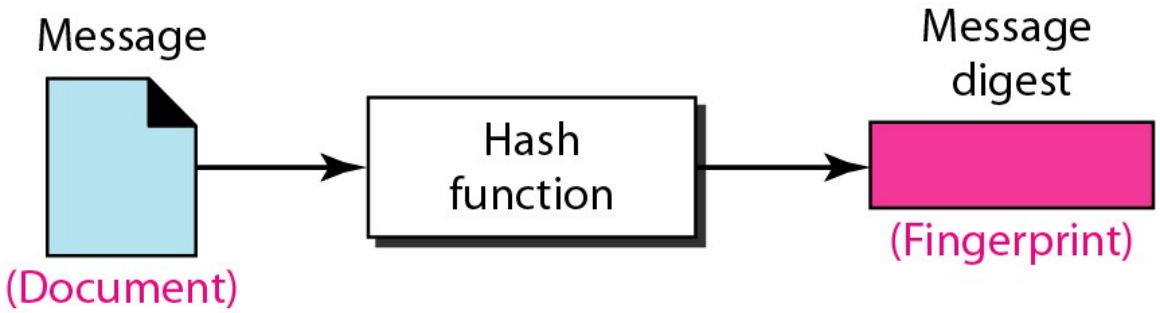
Message Integrity

- Encryption/decryption provides confidentiality, but not integrity
- Sometimes, a document does not need confidentiality, but needs integrity
- To assure that the message is not modified during transfer, a message digest is used

중간에 메시지 바꿔치는 어택 - integrity attack

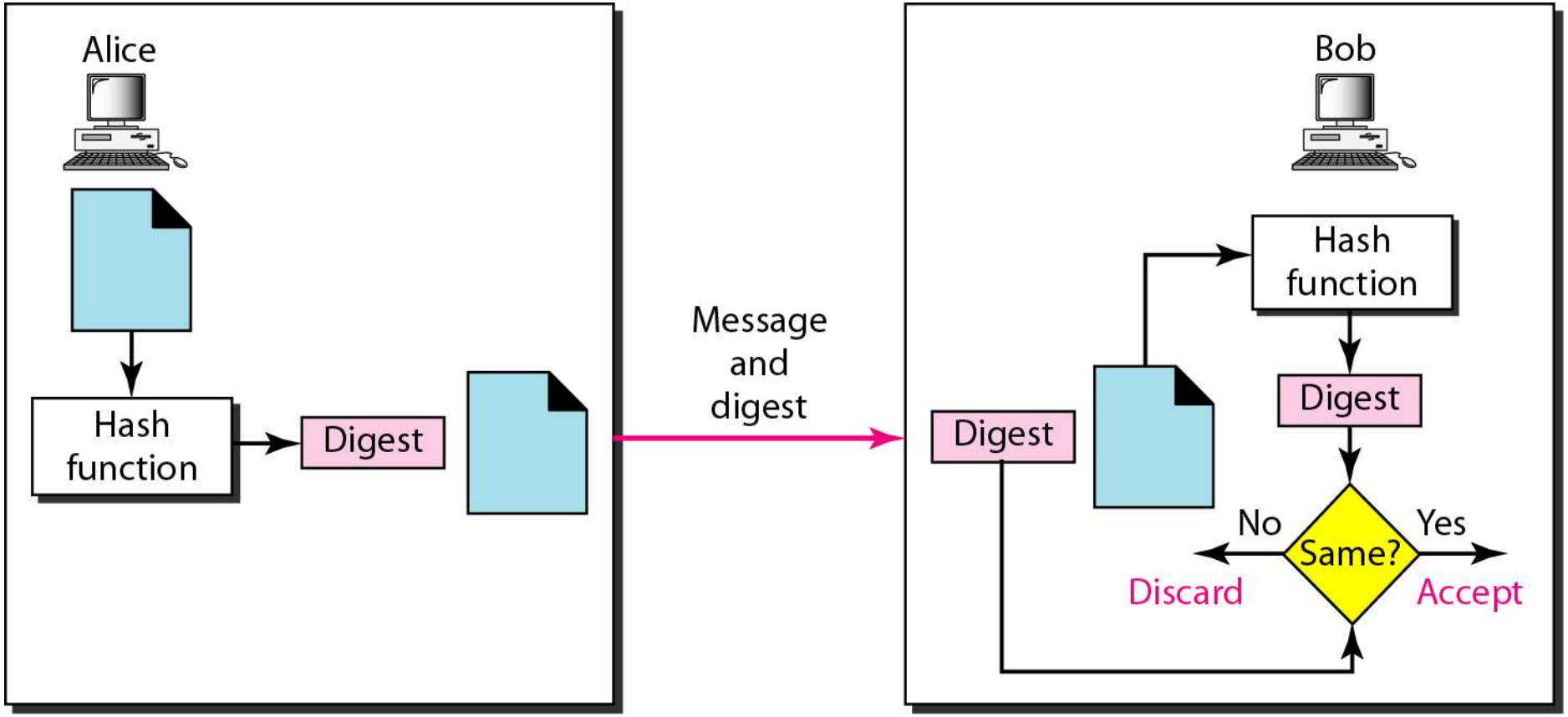
Message Integrity

- Sender uses hash function to generate message digest



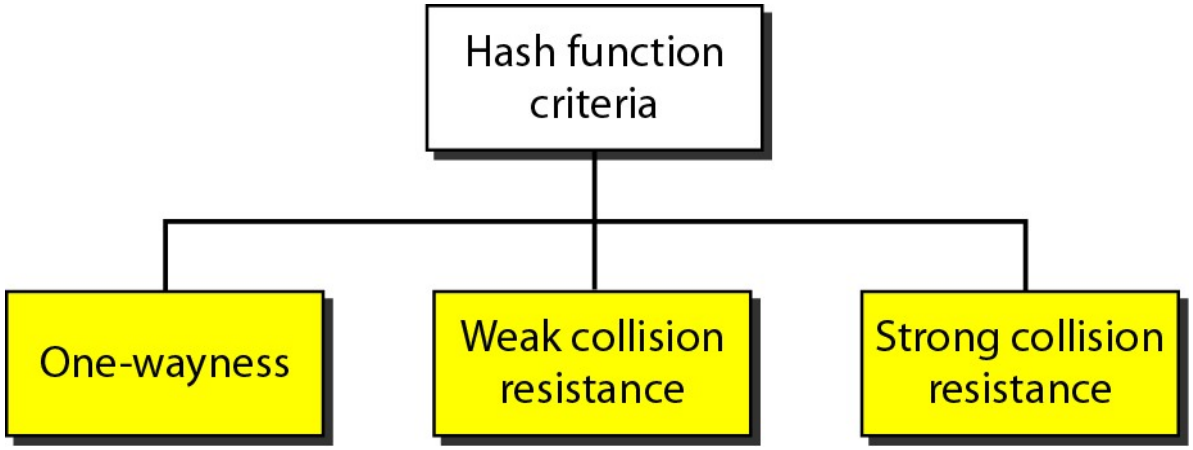
- Sender sends message and message digest to the receiver
- Receiver receives message, generates message digest using the same hash function, and compares it with the message digest received from sender
- Message digest should be sent secretly (using encryption)

Message Integrity



hash functions

- Hash functions used for providing message integrity should have the following characteristics

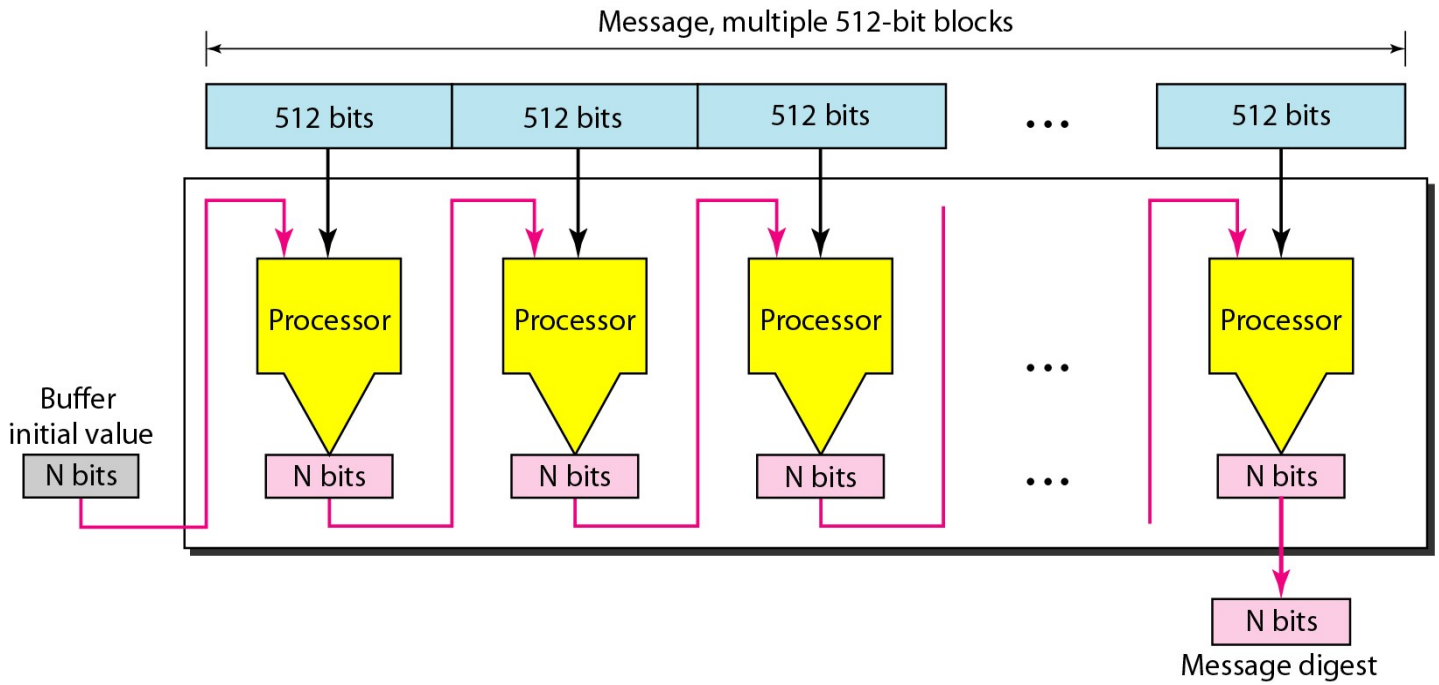


hash functions

- One-wayness digest로 원문을 복원할 수 없다
  - Message cannot be derived from digest
- Weak collision resistance hash가 같은 걸로 바꿔치기하는 게 어려워야 한다
  - For a message x, one cannot find another message y that results in the same hash value x를 주고 그것과 같은 hash를 가진 y를 찾는 게 어려움
- Strong collision resistance x를 주지 않고도 그것과 같은 hash를 가진 y를 찾는 게 어려움
  - One cannot find two messages x and y that result in the same hash value

hash function: SHA-1

- Secure Hash Algorithm – 1
- Designed by NIST (National Institute of Standards and Technology)
- creates 160-bit message digest



hash function: SHA-1

- Since results from a block is used as input to the computation of next block, a slight change in the message will result in a totally different hash value

- SHA1("The quick brown fox jumps over the lazy dog") = 2fd4e1c67a2d28fced849ee1bb76e7391b93eb12

- SHA1("The quick brown fox jumps over the lazy dog.") = 408d94384216f890ff7a0c3528e8bed1e0b01621

Message authentication

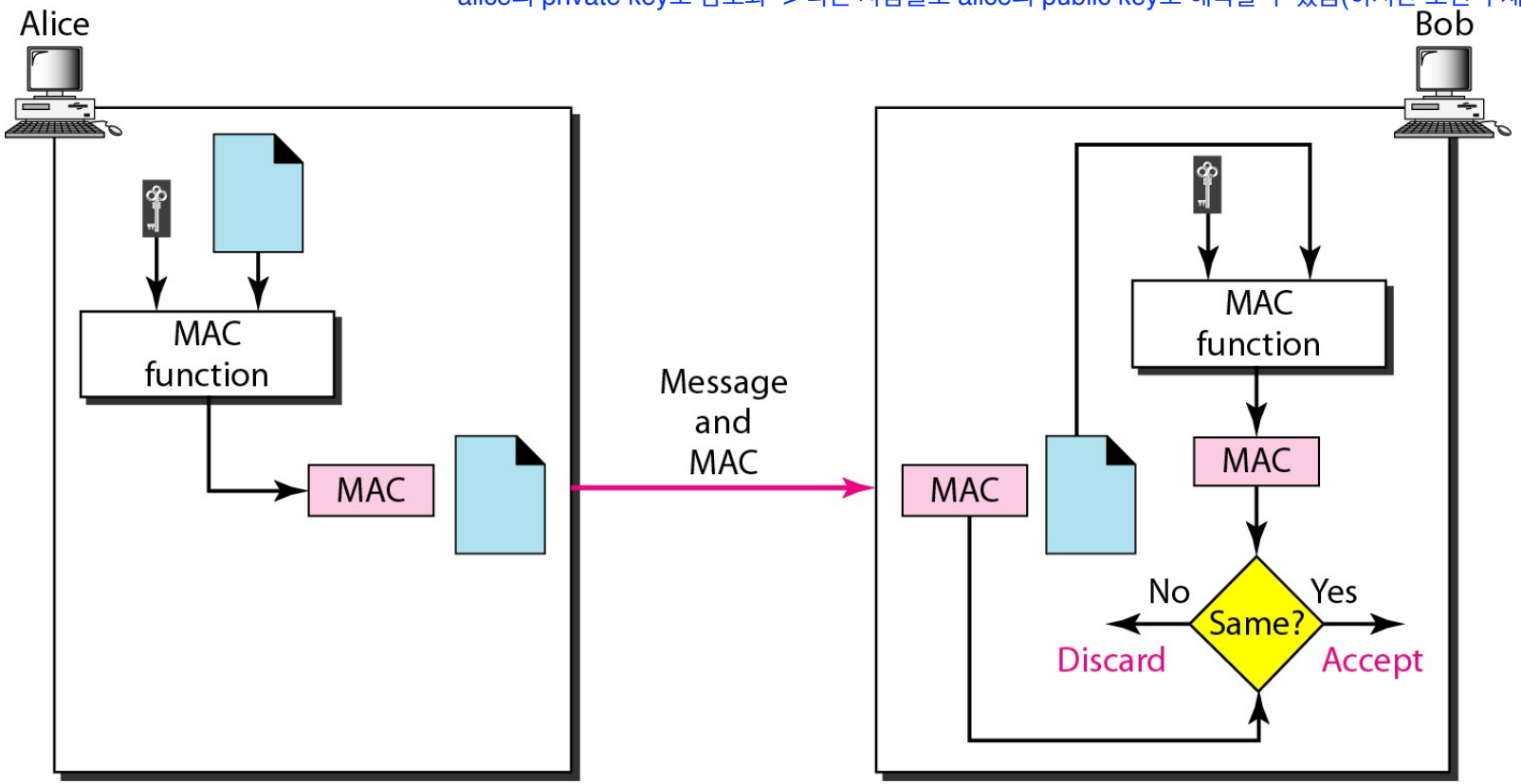
- Hash functions can provide message integrity, it cannot guarantee that the message is actually sent by the intended sender

- For message authentication, a key should be used in generating the message digest

MAC (Message Authentication Code)

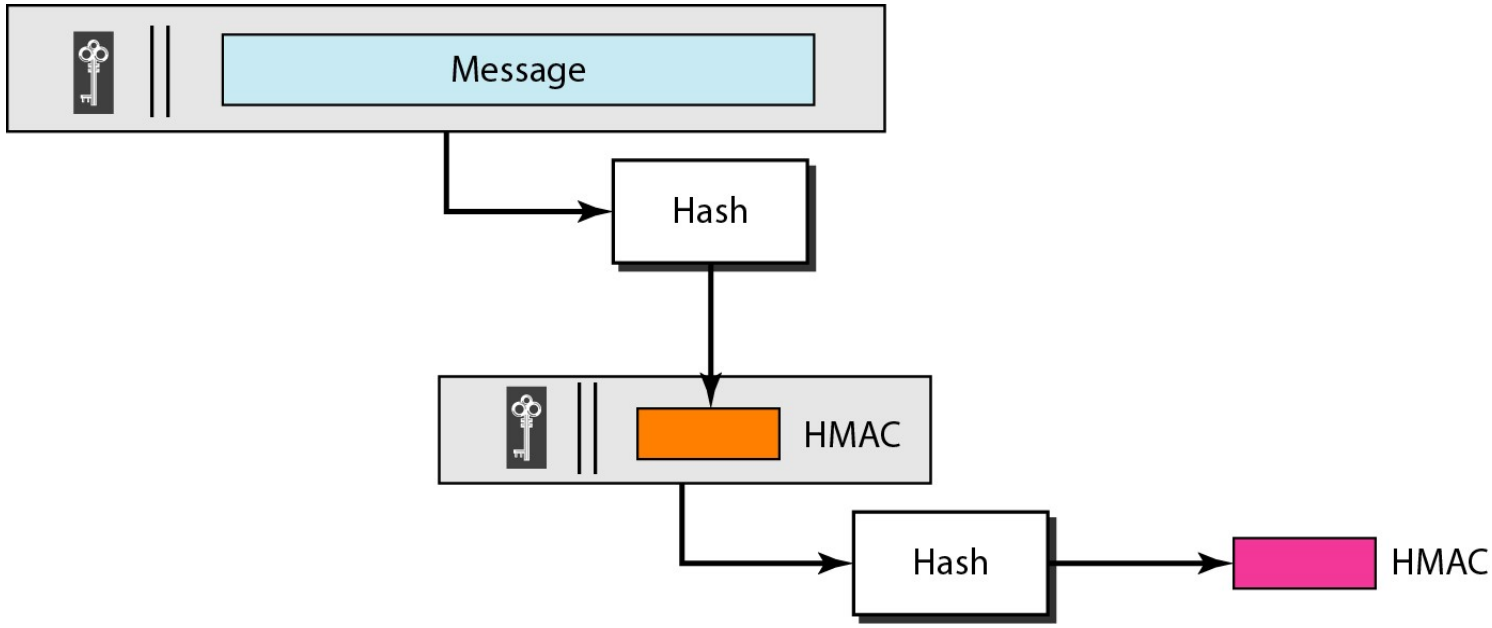
- keyed hash function: hash function that includes key
- Shared key between Alice and Bob is used to generate message digest

alice의 private key로 암호화 -> 다른 사람들도 alice의 public key로 해독할 수 있음(하지만 보낸 주체가 alice 인건 확실)



HMAC

- HMAC (Hashed MAC): a standard method for message authentication



Digital Signature

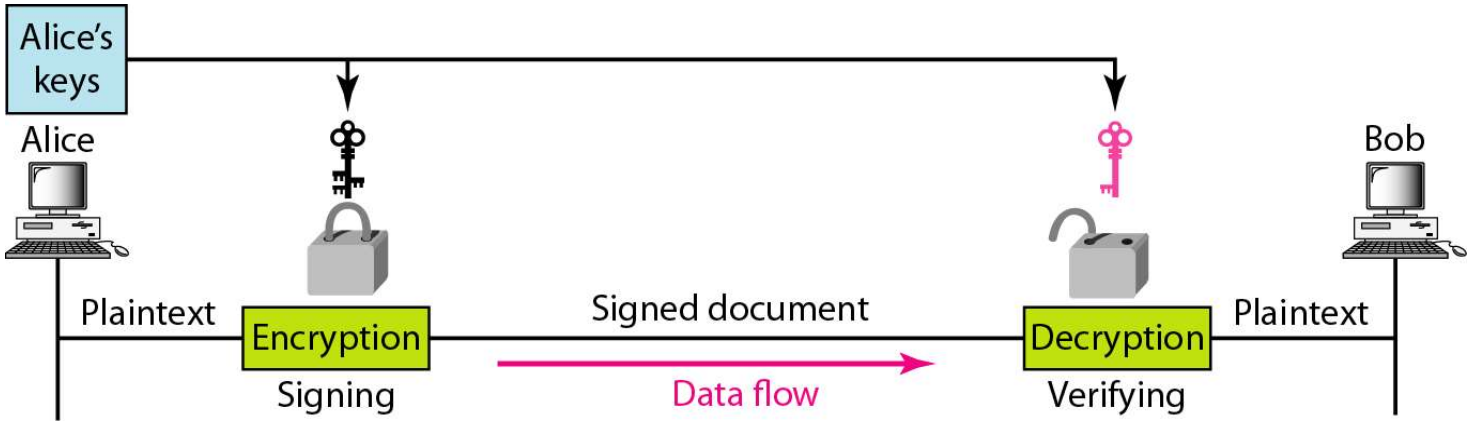
- To use MAC, shared secret key is necessary
  - sharing a key is difficult

- Digital signature uses asymmetric keys to authenticate the sender

Digital Signature

- Alice uses her private key to encrypt the document

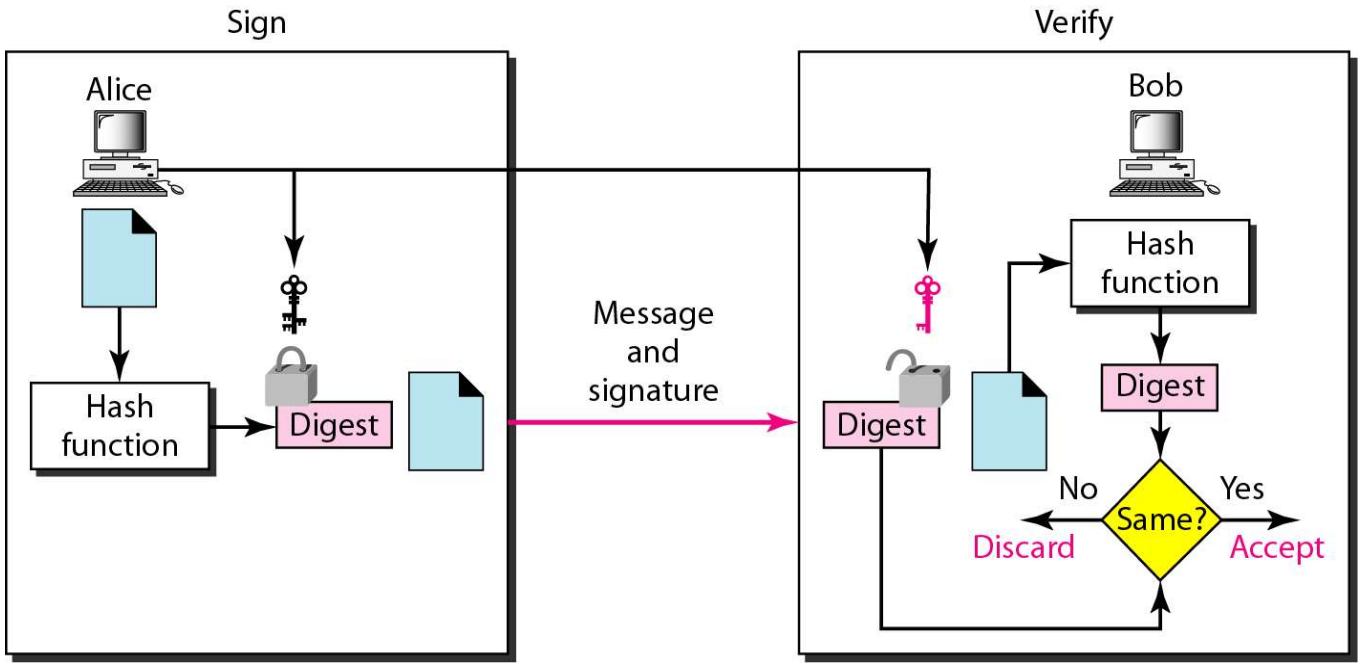
- Bob uses Alice's public key to decode the document.
- If it is decoded, it is sure that Alice sent the document.



Digital Signature

- Asymmetric key encryption takes a lot of time

- Thus, digital signature is often used on message digest, rather than the message itself

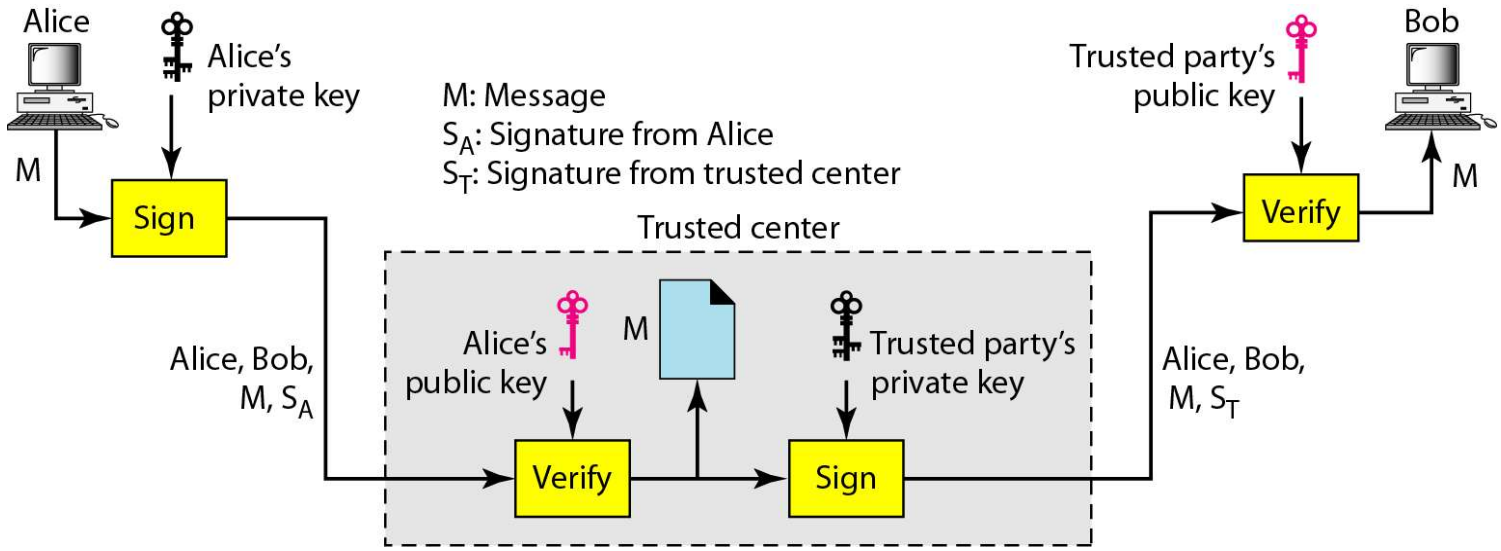


Message Nonrepudiation

- Prevent sender from denying it has sent a message
- A trusted 3rd party is needed for this

Message Nonrepudiation

- A copy of message signed by Alice's key is stored at the Trusted center
- Later on, the Trusted center signs the message with its private key and sends it to Bob, and Bob receives it
- If Alice denies that she has sent the message, Bob can compare the message with its copy stored at the trusted center



End of Class

snr계산