IS511 Introduction to Information Security

Lecture 2

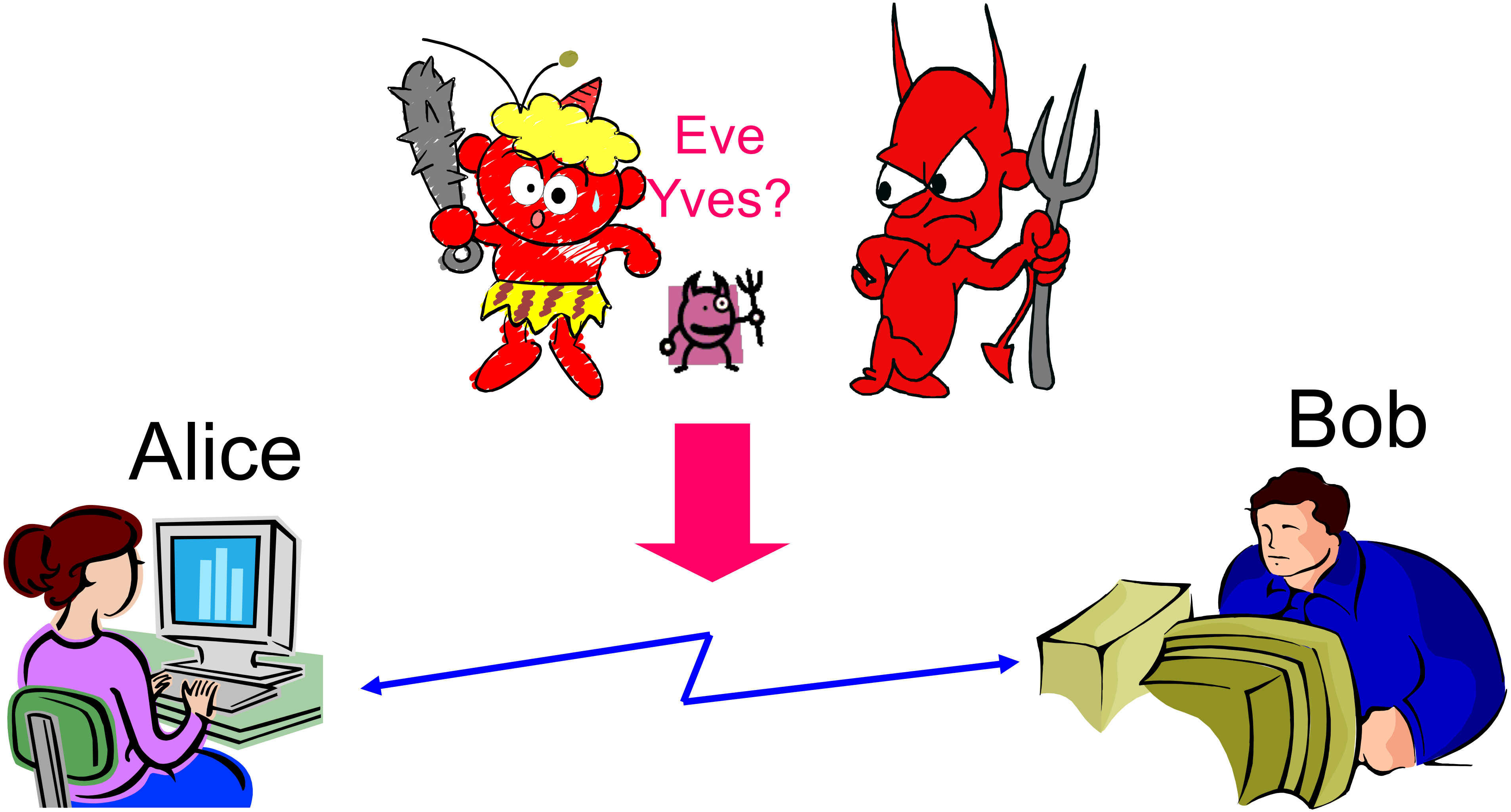
Cryptography 1

Yongdae Kim

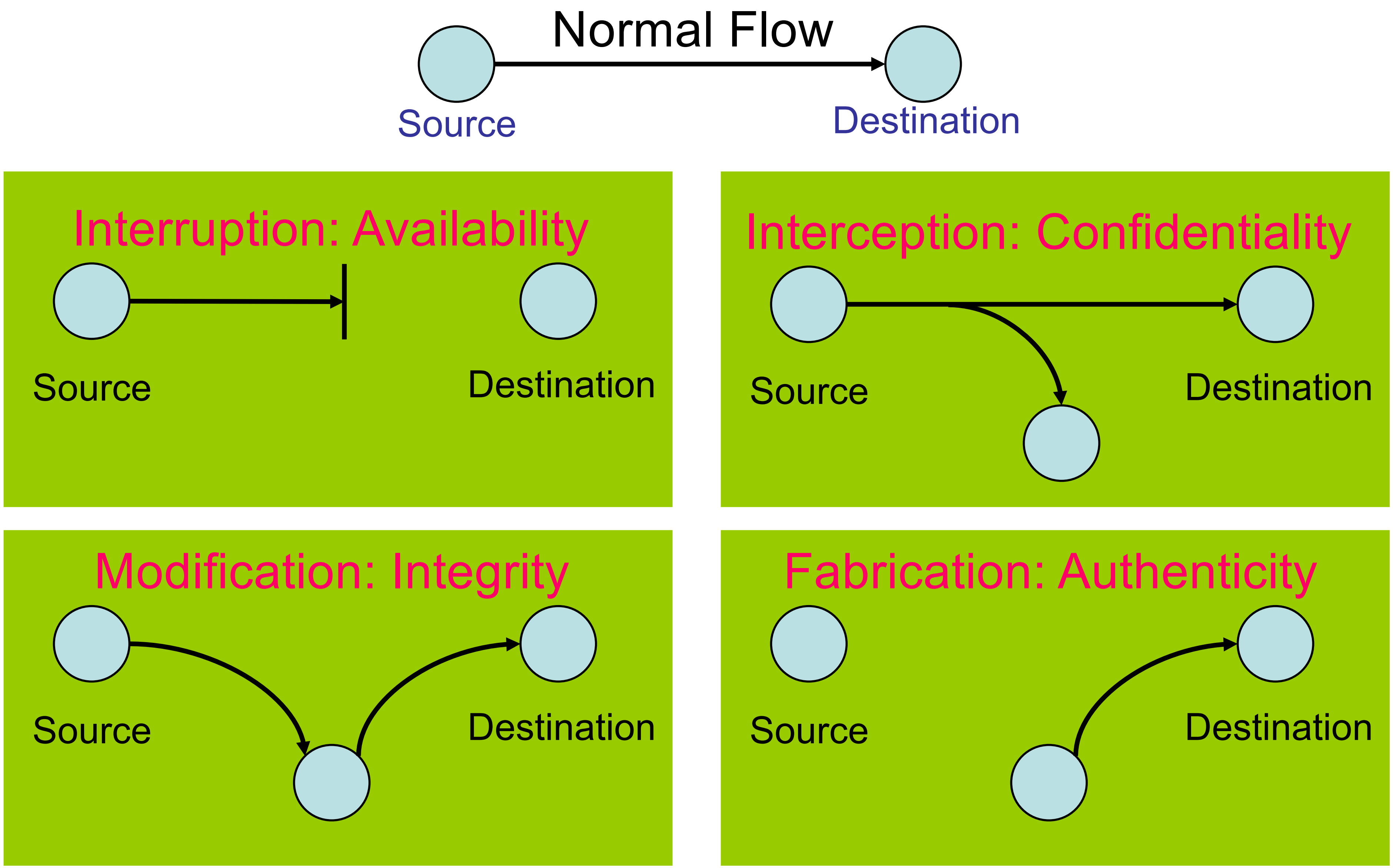
Recap

- http://syssec.kaist.ac.kr/~yongdaek/courses/is511/
- E-mail policy
  - Include [is511]
  - Profs + TA: IS511-prof@gsis.kaist.ac.kr
  - Profs + TA + Students: IS511@gsis.kaist.ac.kr
- Text only posting, email!
- Preproposal
- Proposal: English only

The main players



Attacks

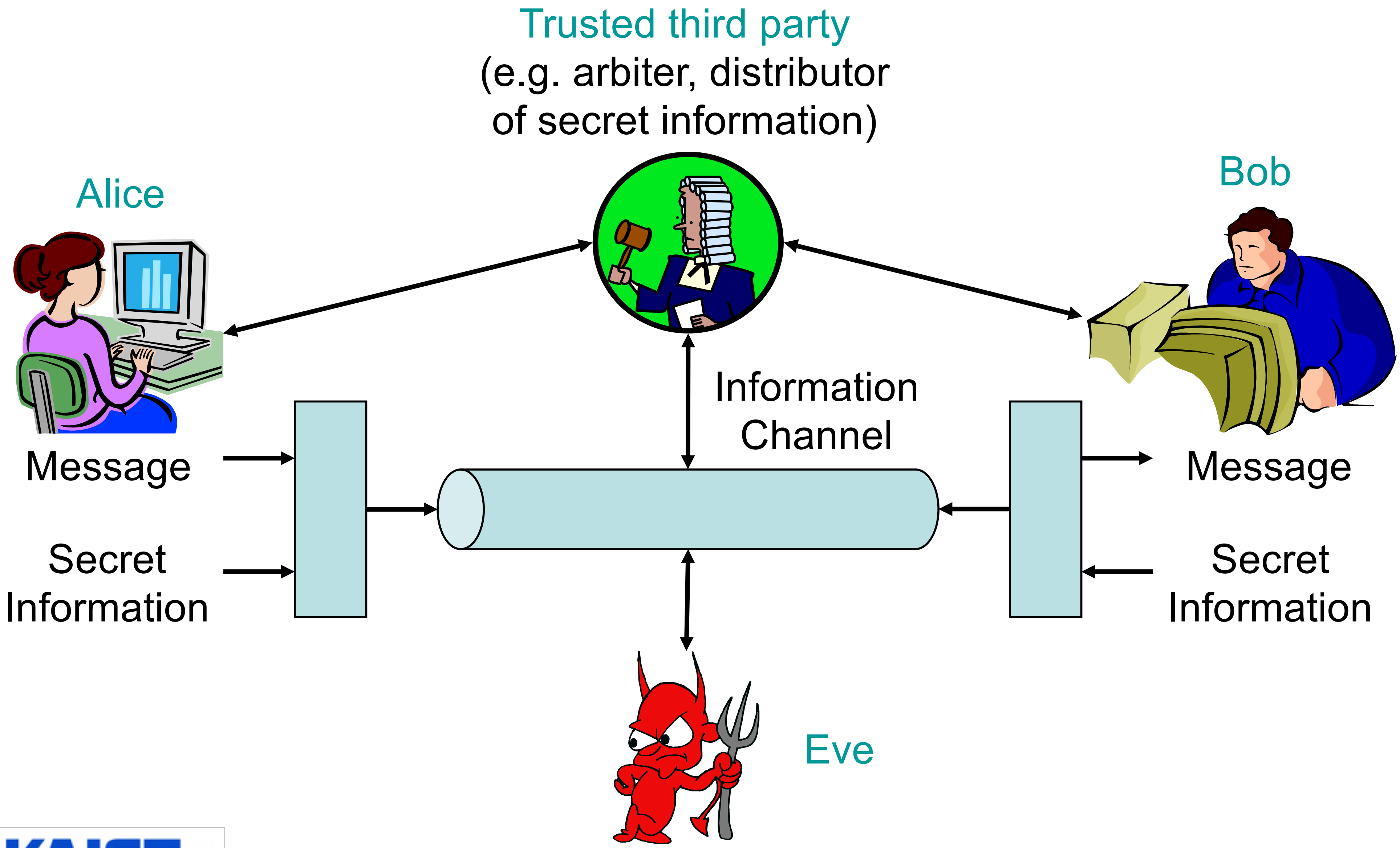

Normal Flow
Source
Destination
Interruption: Availability
Source
Destination
Interception: Confidentiality
Source
Destination
Modification: Integrity
Source
Destination
Fabrication: Authenticity
Source
Destination

Taxonomy of Attacks

- Passive attacks
  - Eavesdropping
  - Traffic analysis

- Active attacks
  - Masquerade
  - Replay
  - Modification of message content
  - Denial of service

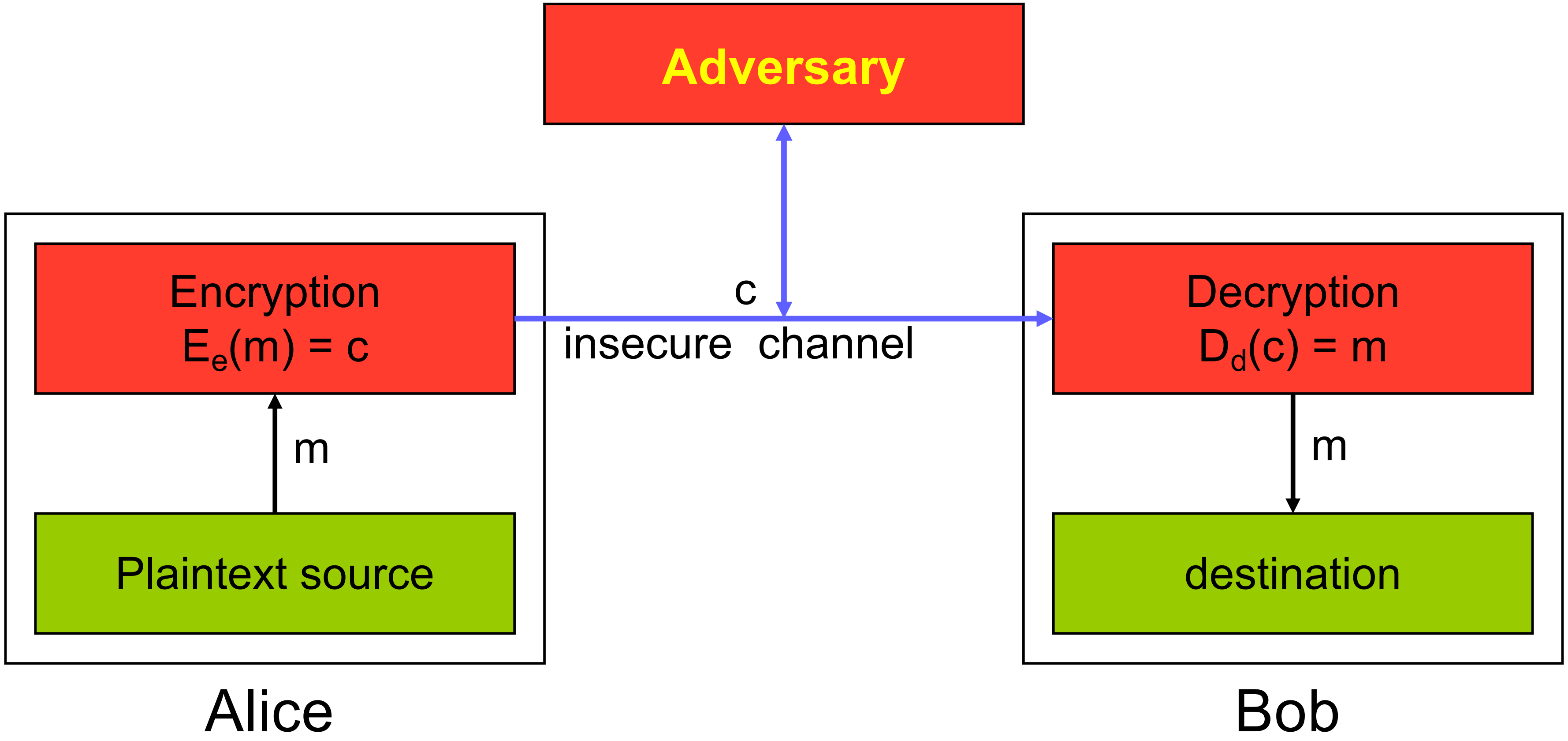
Big picture


Trusted third party
(e.g. arbiter, distributor
of secret information)
Alice
Bob
Information
Channel
Message
Secret
Information
Message
Secret
Information
Eve

Terminology for Encryption

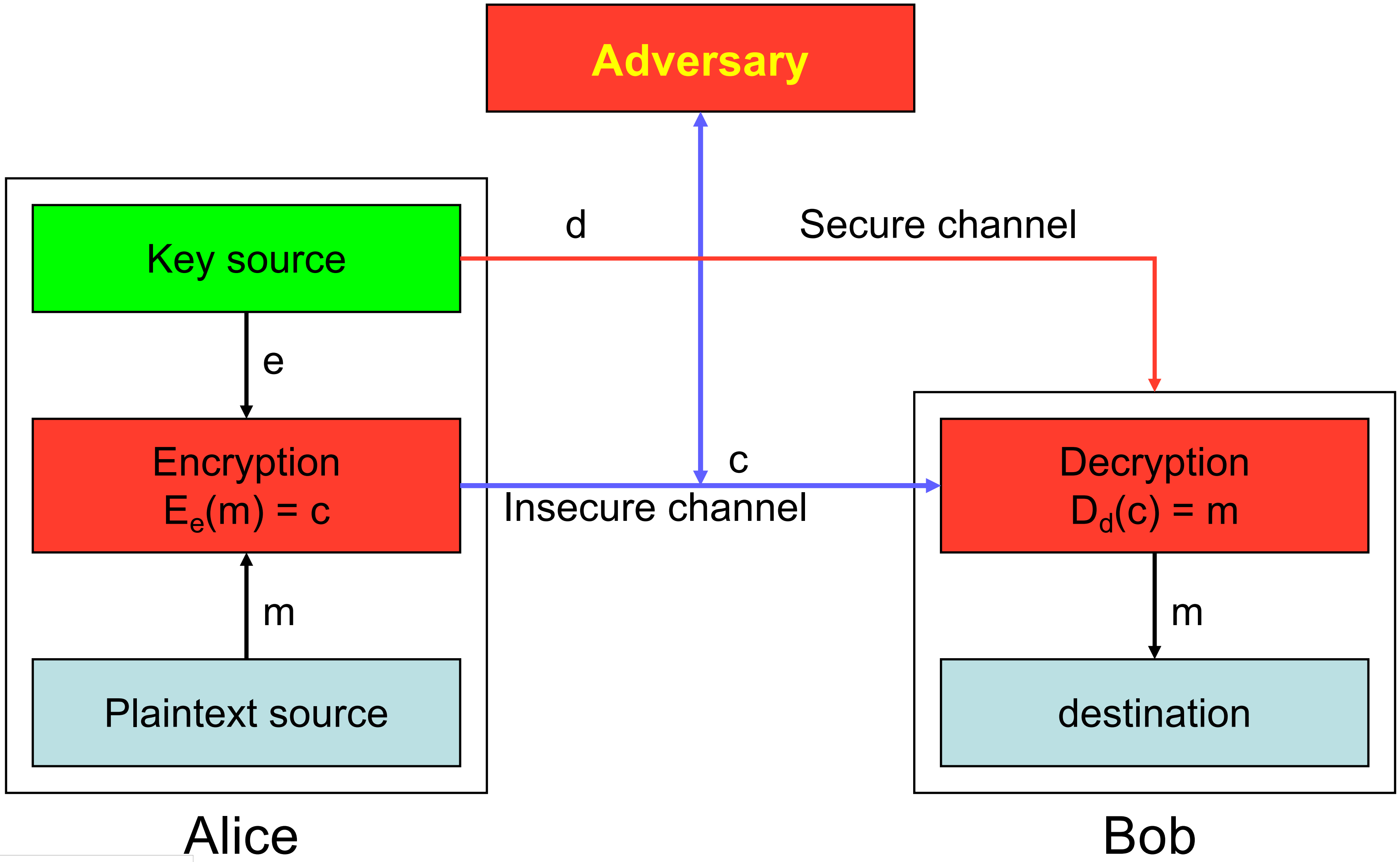
- A denotes a finite set called the *alphabet*
- M denotes a set called the *message space*
  - M consists of strings of symbols from an alphabet
  - An element of M is called a *plaintext*
- C denotes a set called the *ciphertext space*
  - C consists of strings of symbols from an alphabet
  - An element of C is called a *ciphertext*
- K denotes a set called the *key space*
  - An element of K is called a *key*
- E_e is an *encryption function* where $e \in K$
- D_d called a *decryption function* where $d \in K$

Encryption

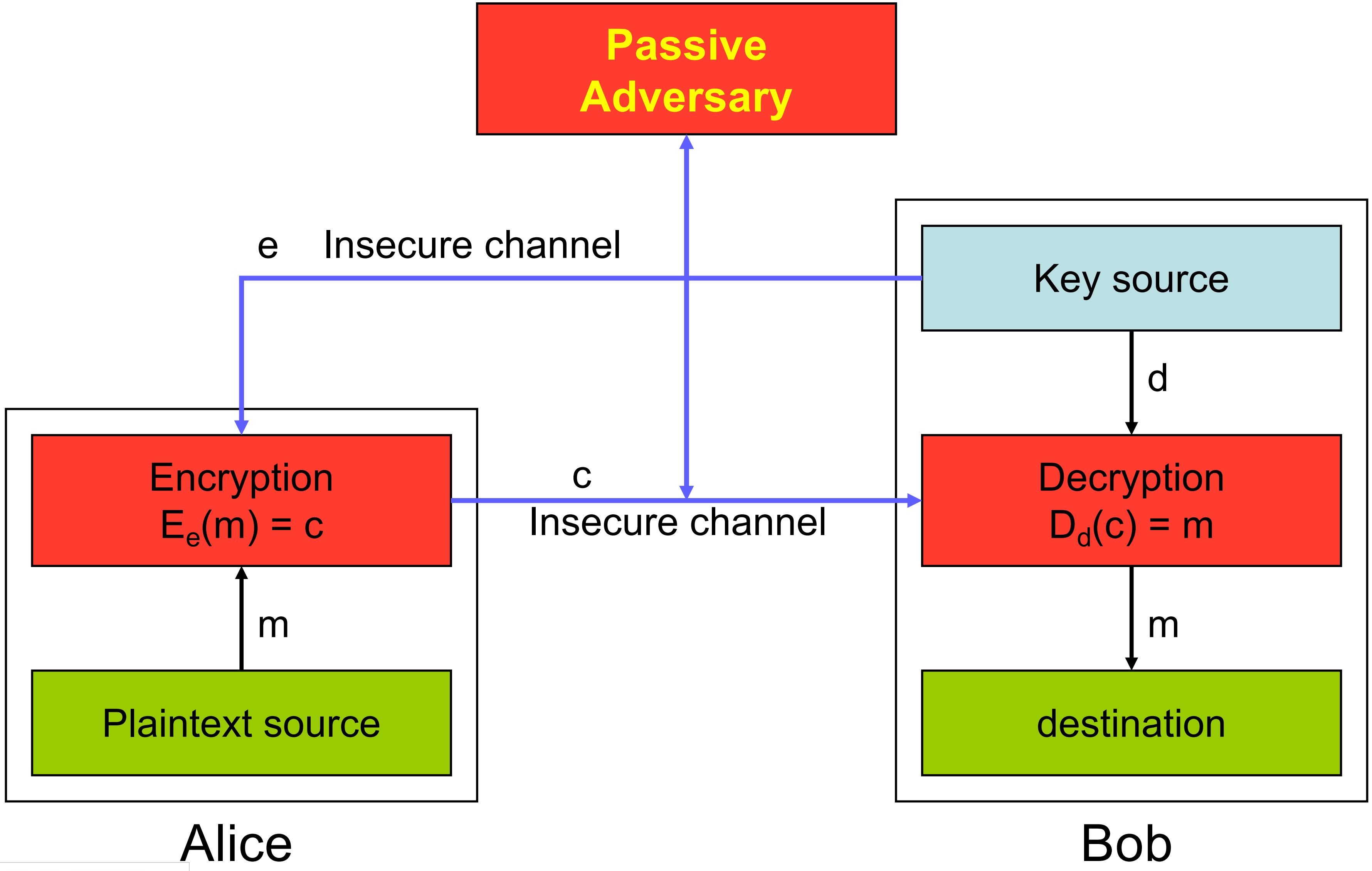


- Why do we use key?
  - Or why not use just a shared encryption function?

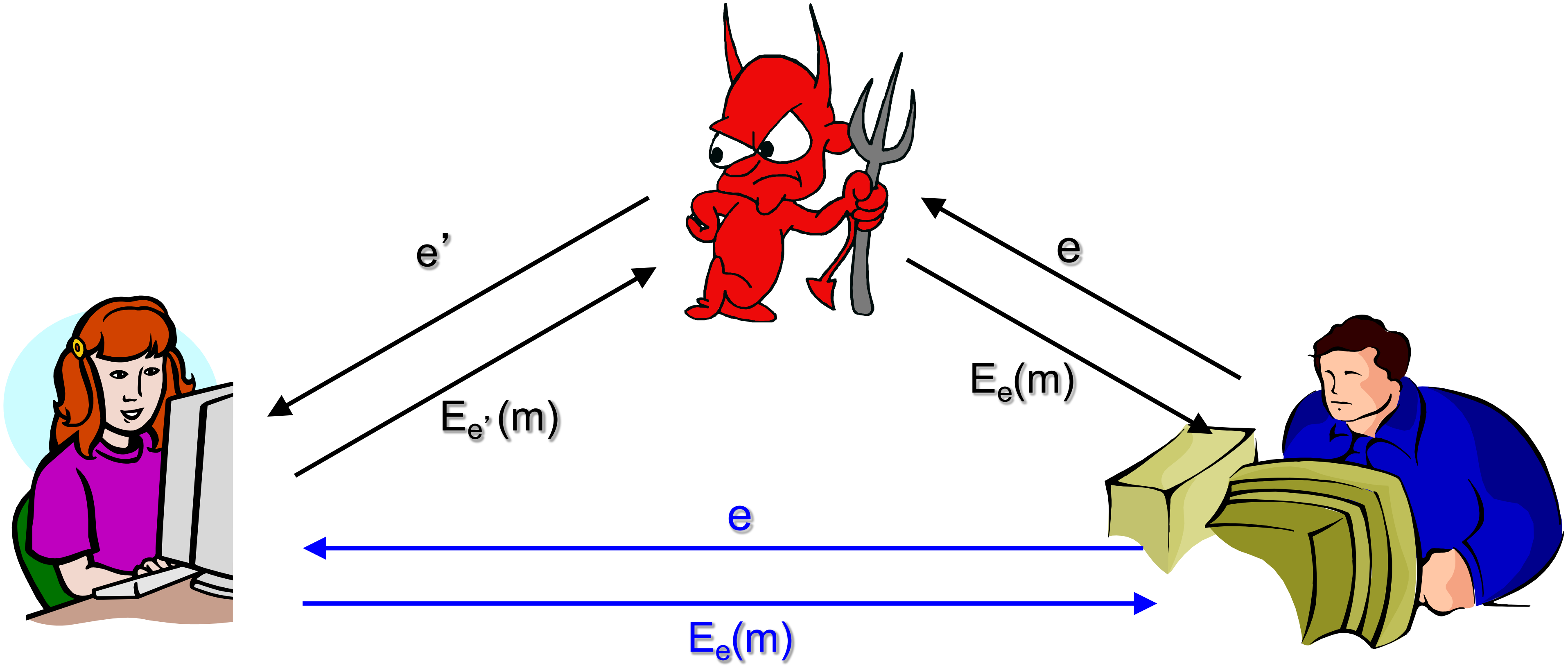
SKE with Secure channel



PKE with insecure channel



Public key should be authentic!



- Need to authenticate public keys

KAIST

Digital Signatures

- Primitive in authentication and non-repudiation
- Signature
  - Process of transforming the message and some secret information into a tag
- Nomenclature
  - M is set of messages
  - S is set of signatures
  - S_A: Signature generation algorithm
  - V_A is verification transformation from M to S for A, publicly known

Key Establishment, Management

- Key establishment
  - Process to whereby a shared secret key becomes available to two or more parties
  - Subdivided into key agreement and key transport.
- Key management
  - The set of processes and mechanisms which support key establishment
  - The maintenance of ongoing keying relationships between parties

Symmetric vs. Public key

	Pros	Cons
SKE	High data throughput Relatively short key size	The key must remain secret at both ends $O(n^2)$ keys to be managed Relatively short lifetime of the key
PKE	O(n) keys Only the private key must be kept secret longer key life time digital signature	Low data throughput Much larger key sizes

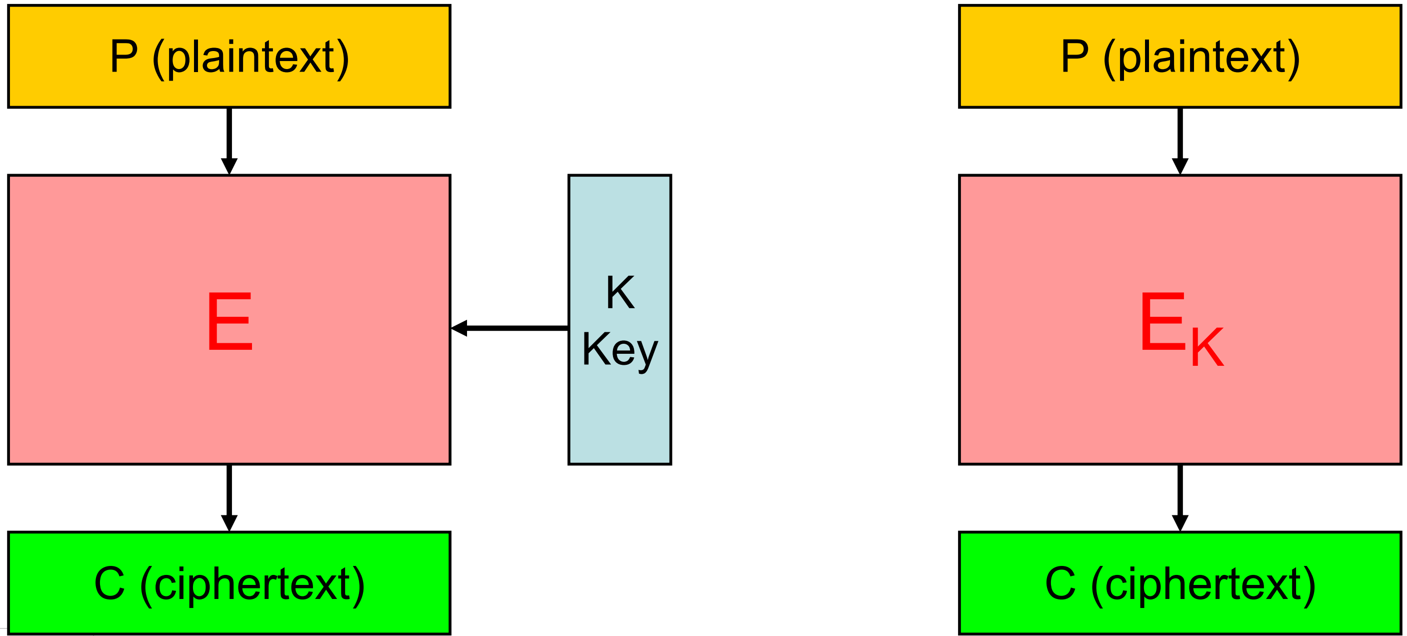
KAIST

Symmetric key Encryption

- Symmetric key encryption
  - if for each (e,d) it is easy computationally easy to compute e knowing d and d knowing e
  - Usually e = d
- Block cipher
  - breaks up the plaintext messages to be transmitted into *blocks* of a fixed length, and encrypts one block at a time
- Stream cipher
  - encrypt individual characters of plaintext message one at a time, using encryption transformation which varies with time

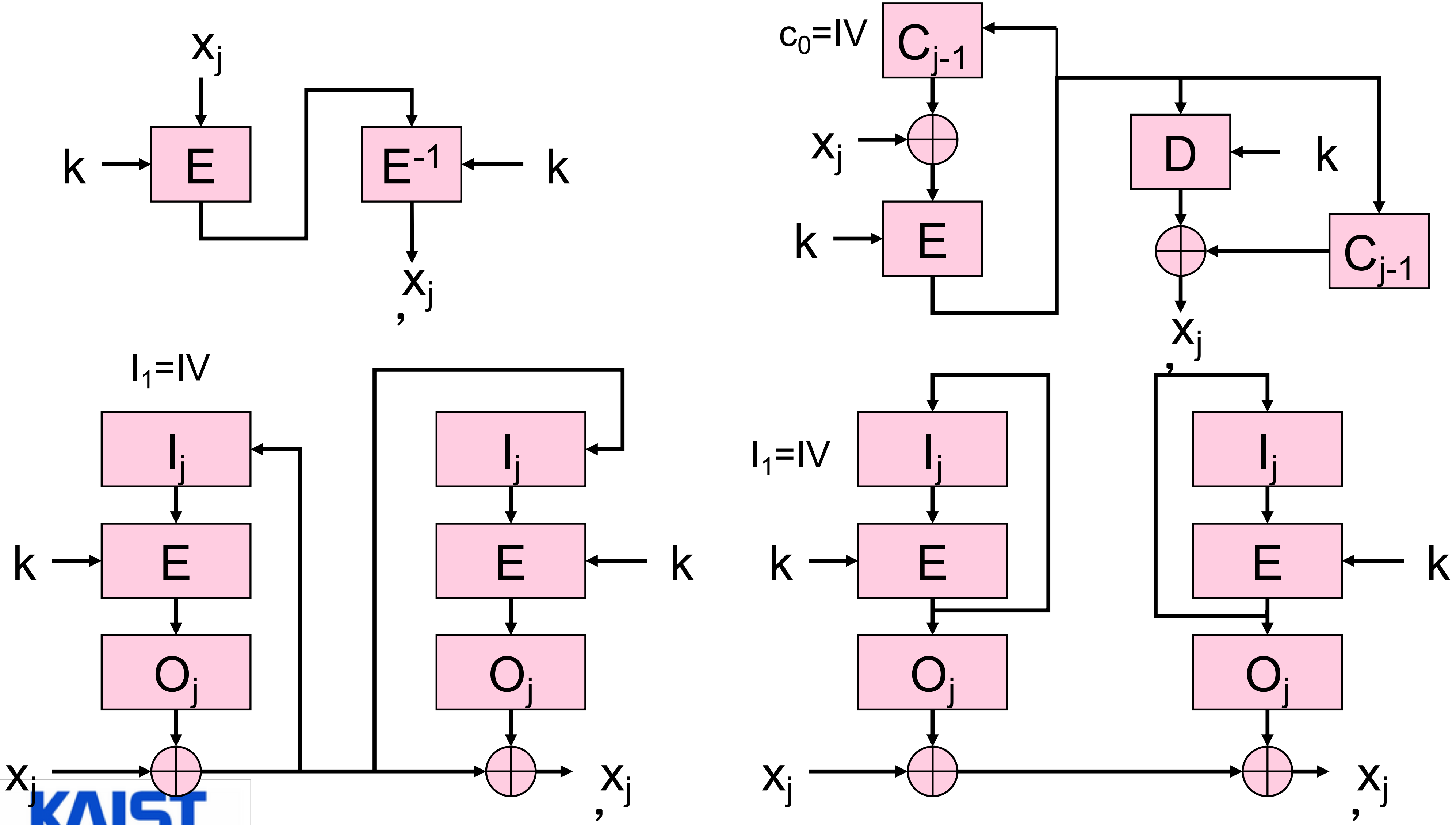
Block Cipher

- E: $V_n \times K \to V_n$
  - $V_n = \{0,1\}^n$, $K = \{0, 1\}^k$, n is called block length, k is called key size
  - E(P, K) = C for $K \in K$ and P, $C \in V_n$
  - E(P, K) = $E_K(P)$ is invertible mapping from V_n to V_n
    - E_K: encryption function
  - D(C, K) = $D_K(C)$ is the inverse of E_K
    - D_k: decryption function



Modes of Operation

- A block cipher encrypts plaintext in fixed-size n-bit blocks (often n =128). What happens if your message is greater than the block size?

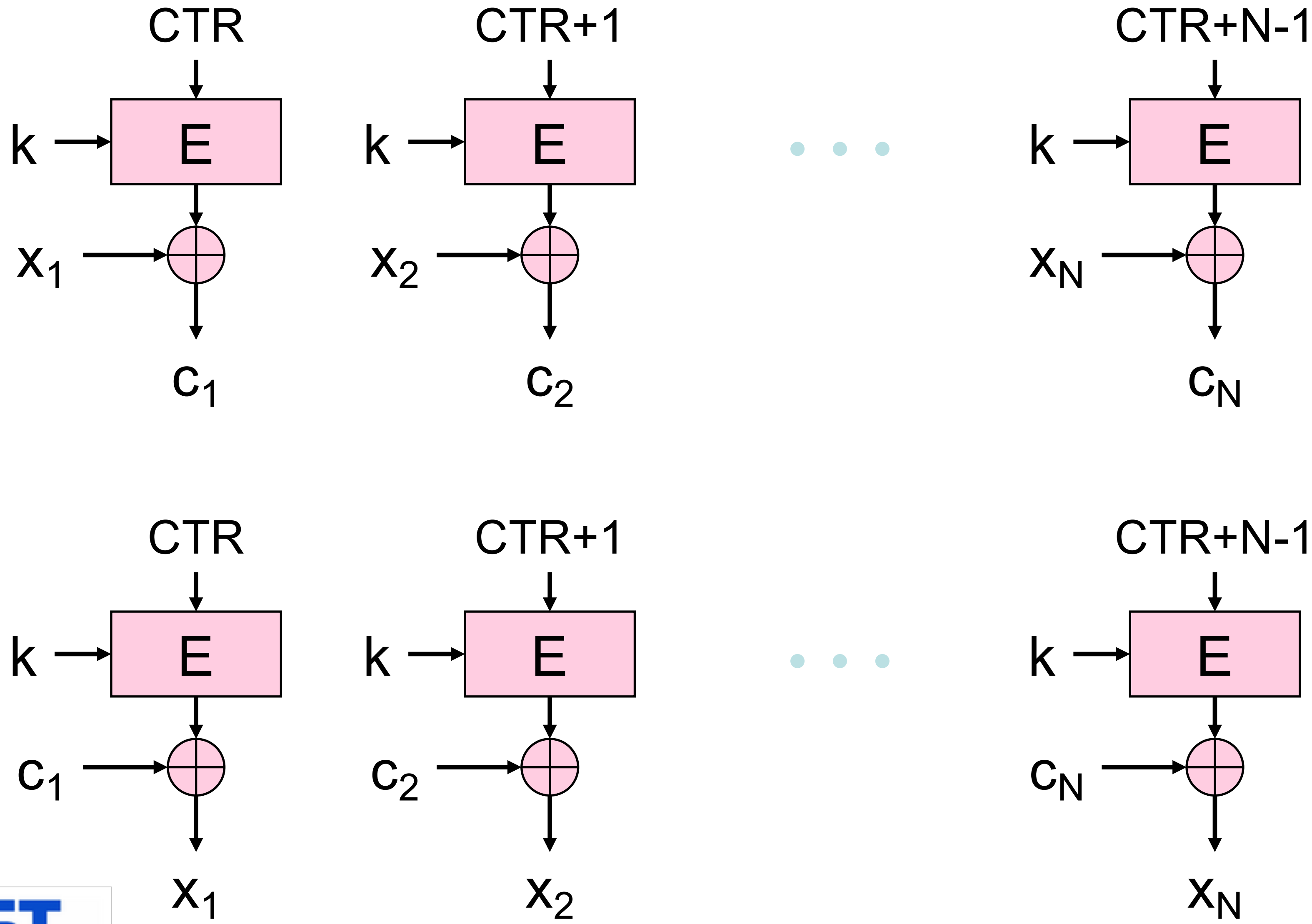


Modes of Operation

- ECB
  - Encryption: $c_j \leftarrow E_K(x_j)$
  - Decryption: $x_j \leftarrow E^{-1}{}_K(c_j)$
- CBC
  - Encryption: $c_0 \leftarrow IV$, $c_j \leftarrow E_K(c_{j-1} \oplus x_j)$
  - Decryption: $c_0 \leftarrow IV$, $x_j \leftarrow c_{j-1} \oplus E^{-1}{}_K(c_j)$
- CFB
  - Encryption: $I_1 \leftarrow IV$, $c_j \leftarrow x_j \oplus E_K(I_j)$, $I_{j+1} = c_j$
  - Decryption: $I_1 \leftarrow IV$, $x_j \leftarrow c_j \oplus E_K(I_j)$, $I_{j+1} = c_j$
- OFB
  - Encryption: $I_1 \leftarrow IV$, $o_j = E_K(I_j)$, $c_j \leftarrow x_j \oplus o_j$, $I_{j+1} = o_j$
  - Decryption: $I_1 \leftarrow IV$, $o_j = E_K(I_j)$, $x_j \leftarrow c_j \oplus o_j$, $I_{j+1} = o_j$

KAIST

Modes of Operation (CTR)



KAIST

CTR advantages

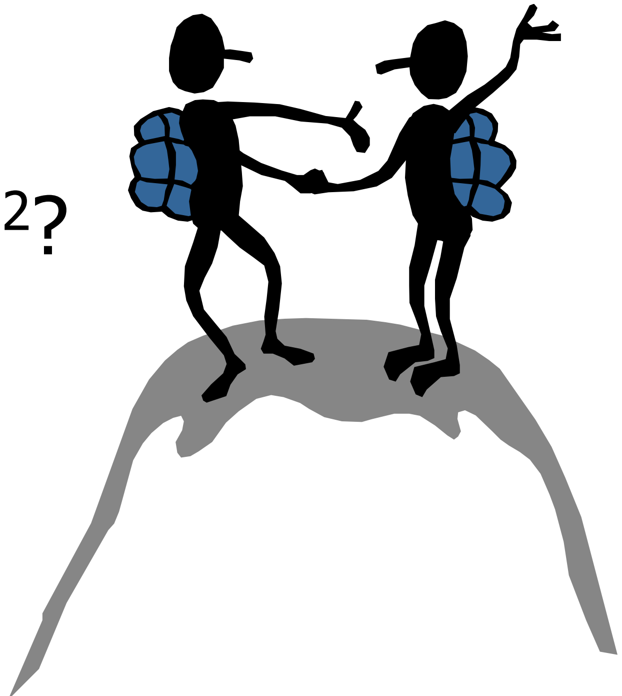
- Hardware efficiency
  - Parallelizable
- Software efficiency
  - Similar, modern processors support parallel computation
- Preprocessing
  - Pad can be computed earlier
- Random-access
  - Each ciphertext block can be encrypted independently
  - important in applications like hard-disk encryption
- Provable security
  - no worse than what one gets for CBC encryption
- Simplicity
  - No decryption algorithm and key scheduling

Double DES

- $C = E_{K2}[E_{K1} [P]]$
- $P = D_{K1}[D_{K2}[C]]$

- Reduction to single stage?
  - $E_{K2}[E_{K1} [P]] =? E_{K3}[P]$
  - It was proven that it does not hold

Meet-in-the-middle Attack



- Diffie 1977
- Exhaustively cracking it requires 2^{112}?
- $C = E_{K2}[E_{K1}[P]]$
  - $X = E_{K1}[P] = D_{K2}[C]$
- Given a known pair, (P, C)
  - Encrypt P with all possible 2^{56} values of K_1
  - Store this results and sort by X
  - Decrypt C with all possible 2^{56} K_2, and check table
  - If same, accept it as the correct key
- Are we done? &&#@!#(

Meet-in-the-middle Attack

- Little statistics
  - For any P, there are 2^{64} possible C
  - DDES uses 112 bit key, so 2^{112} keys
  - Given C, there are $2^{112}/2^{64} = 2^{48}$ possible P
    - So there are 2^{48} false alarms
  - If one more (P’ , C’) pair, we can reduce it to 2^{-16}
- So using two (plaintext, ciphertext) pairs, we can break DDES c * 2^{56} encryption/decryption

- $C = E_{K2}[D_{K1} [P]]$ different?

KAIST

Triple DES with two keys

- Obvious counter to DDES: Use three keys
  - Complexity?
  - 168 bit key

- Triple DES = EDE = encrypt-decrypt-encrypt
  - $C = E_{K1}[D_{K2}[E_{K1}[P]]]$

- Attacks?
  - No practical one so far

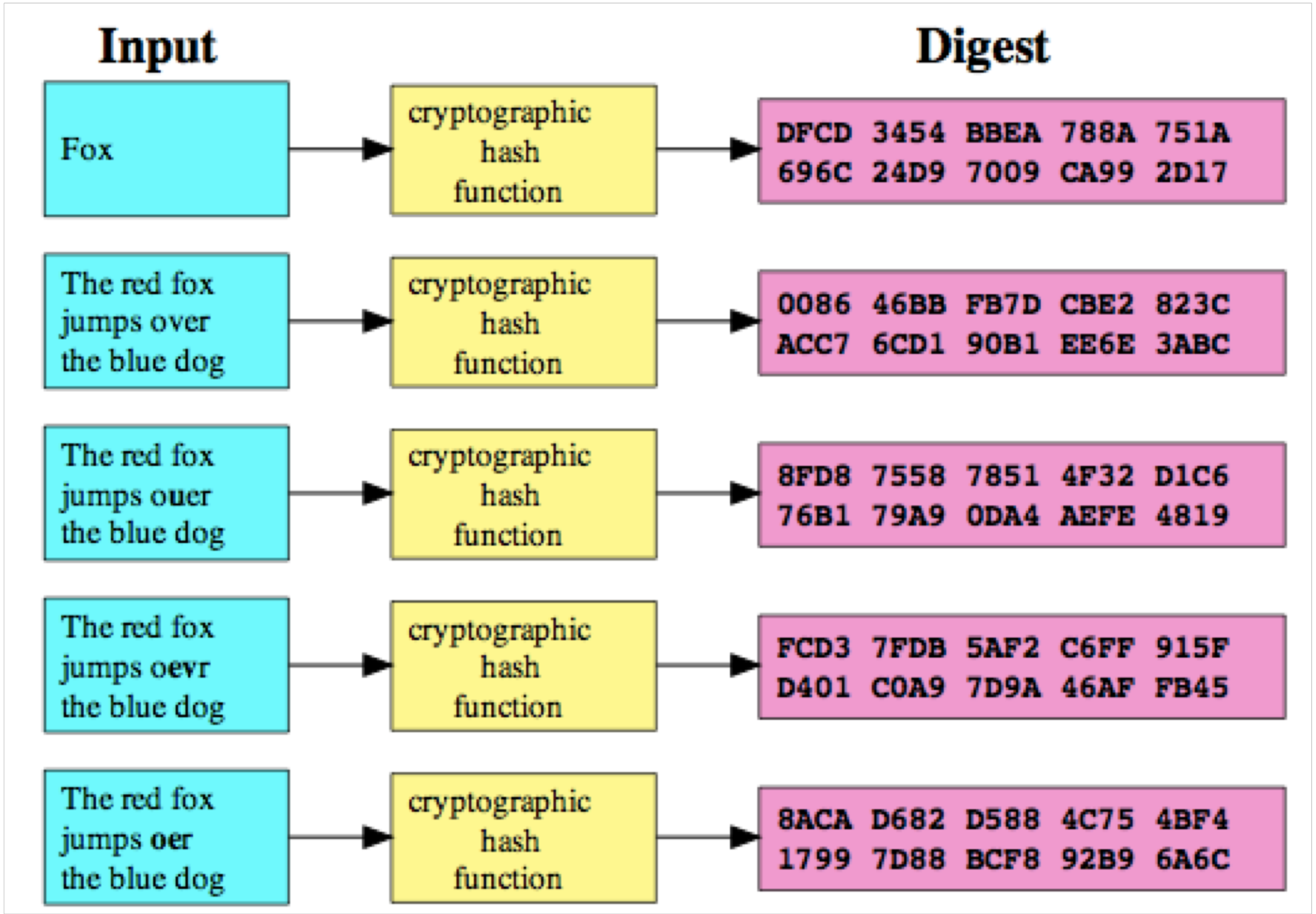
KAIST

Hash function and MAC

- A hash function is a function h
  - compression
  - ease of computation
  - Properties
    - one-way: for a given y, find x' such that h(x') = y
    - collision resistance: find x and x' such that h(x) = h(x')
  - Examples: SHA-1, MD-5

- MAC (message authentication codes)
  - both authentication and integrity
  - MAC is a family of functions h_k
    - ease of computation (if k is known !!)
    - compression, x is of arbitrary length, $h_k(x)$ has fixed length
    - computation resistance
  - Example: HMAC

How Random is the Hash function?



KAIST

Applications of Hash Function

- File integrity

Instructions

The Windows SDK is available as a DVD ISO image file so that you can bu
that you are downloading the correct ISO file, please refer to the table bel
to validate that the file you've downloaded is the correct file.

==

File Name: GRMSDK_EN_DVD.iso

Chip: X86

CRC#: 0xCA4FE79D

SHA1: 0x8695F5E6810D84153181695DA78850988A923F4E

- Digital signature

  $Sign = S_{SK}(h(m))$

- Password verification

stored hash = h(password)

- File identifier
- Hash table
- Generating random numbers

Hash function and MAC

- A hash function is a function h
  - compression
  - ease of computation
  - Properties
    - one-way: for a given y, find x' such that h(x') = y
    - collision resistance: find x and x' such that h(x) = h(x')
  - Examples: SHA-1, MD-5
- MAC (message authentication codes)
  - both authentication and integrity
  - MAC is a family of functions h_k
    - ease of computation (if k is known !!)
    - compression, x is of arbitrary length, $h_k(x)$ has fixed length
    - computation resistance
  - Example: HMAC

MAC construction from Hash

- Prefix
  - M=h(k||x)
  - appending y and deducing h(k||x||y) form h(k||x) without knowing k
- Suffix
  - M=h(x||k)
  - possible a birthday attack, an adversary that can choose x can construct x' for which h(x)=h(x') in $O(2^{n/2})$
- STATE OF THE ART: HMAC (RFC 2104)
  - $HMAC(x)=h(k||p_1||h(k|| p_2||x))$, p1 and p2 are padding
  - The outer hash operates on an input of two blocks
  - Provably secure

How to use MAC?

- A & B share a secret key k
- A sends the message x and the MAC $M \leftarrow H_k(x)$
- B receives x and M from A
- B computes $H_k(x)$ with received M
- B checks if $M = H_k(x)$