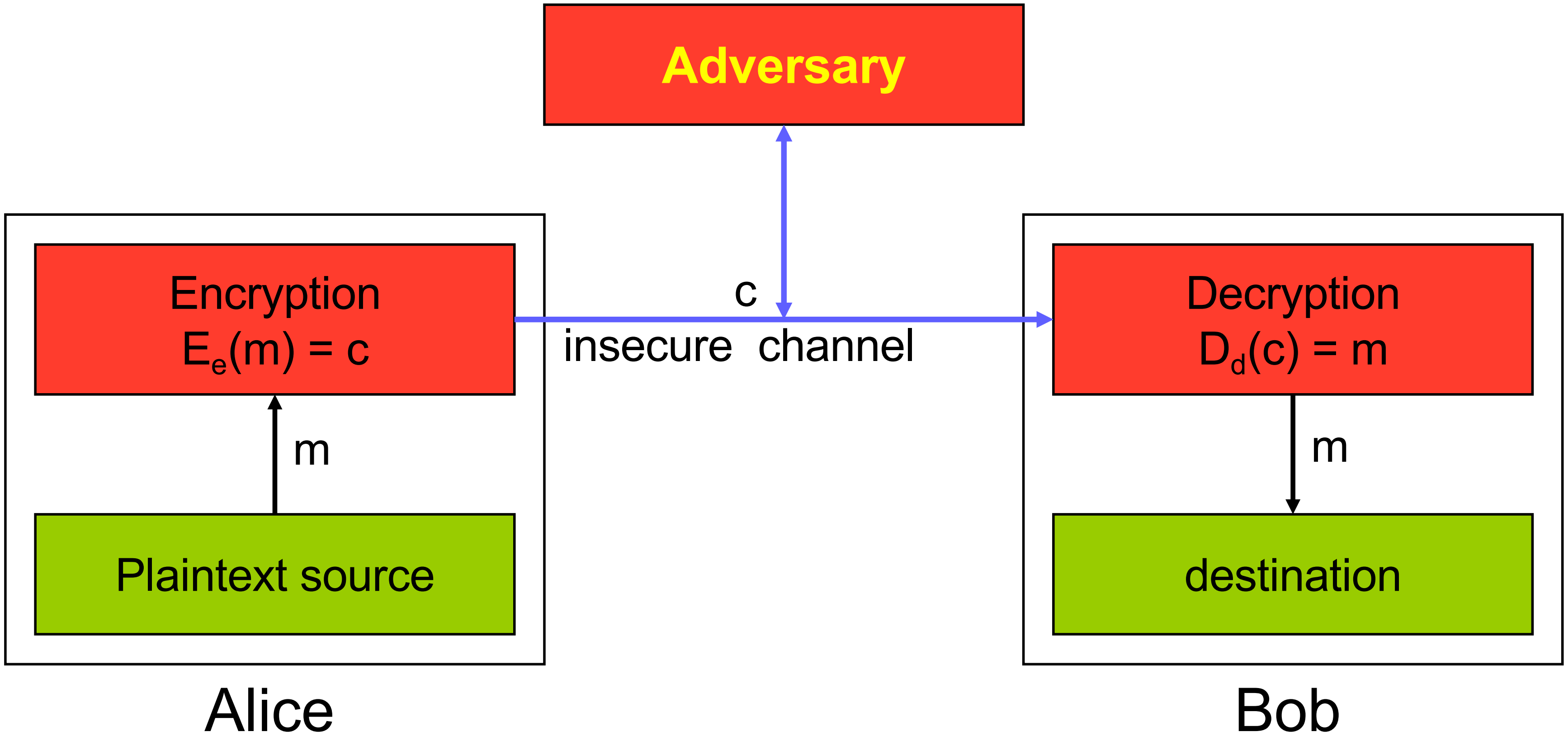
IS511
Introduction to Information Security
Public Key Cryptography and Key Management

Yongdae Kim

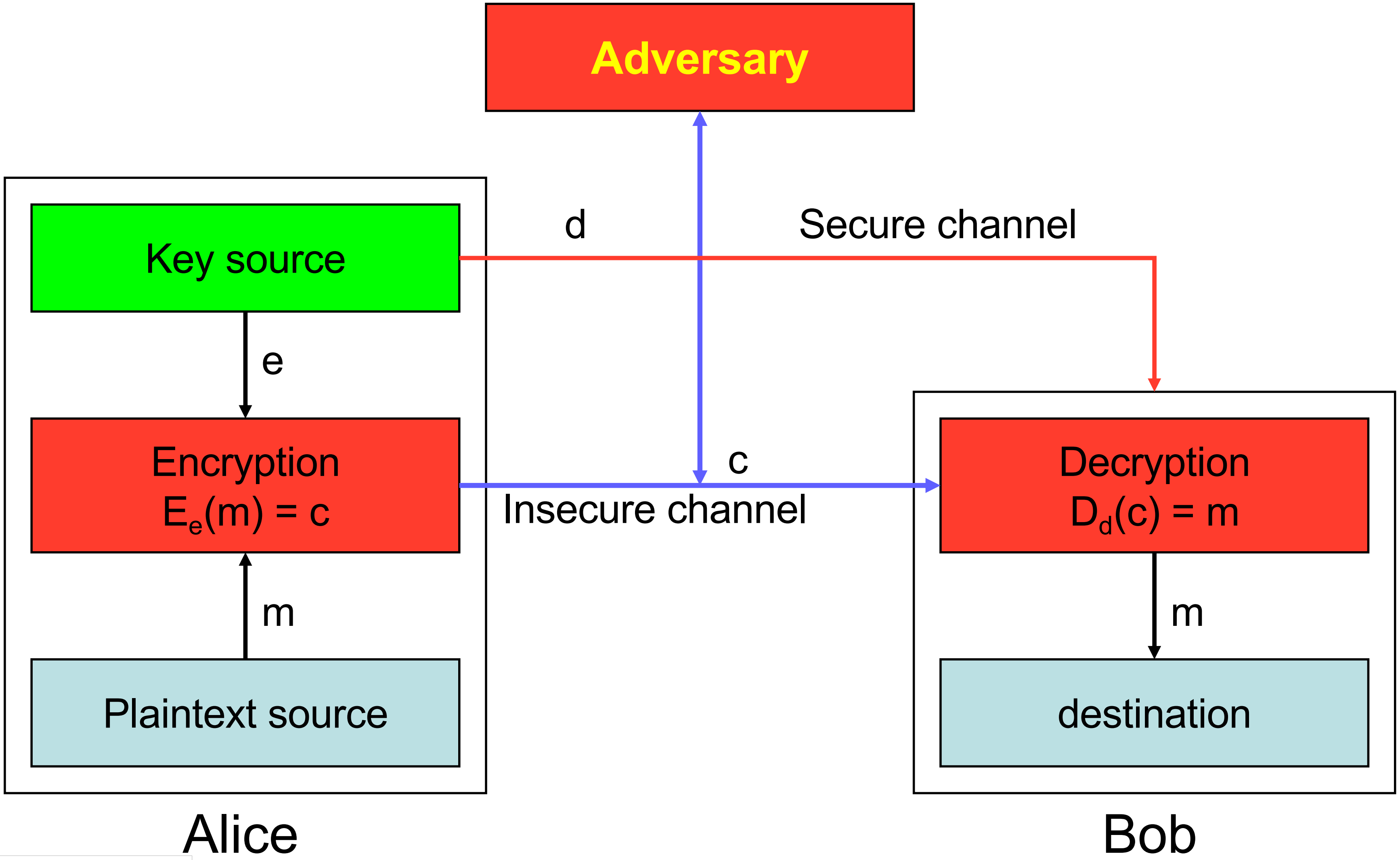
KAIST

Encryption

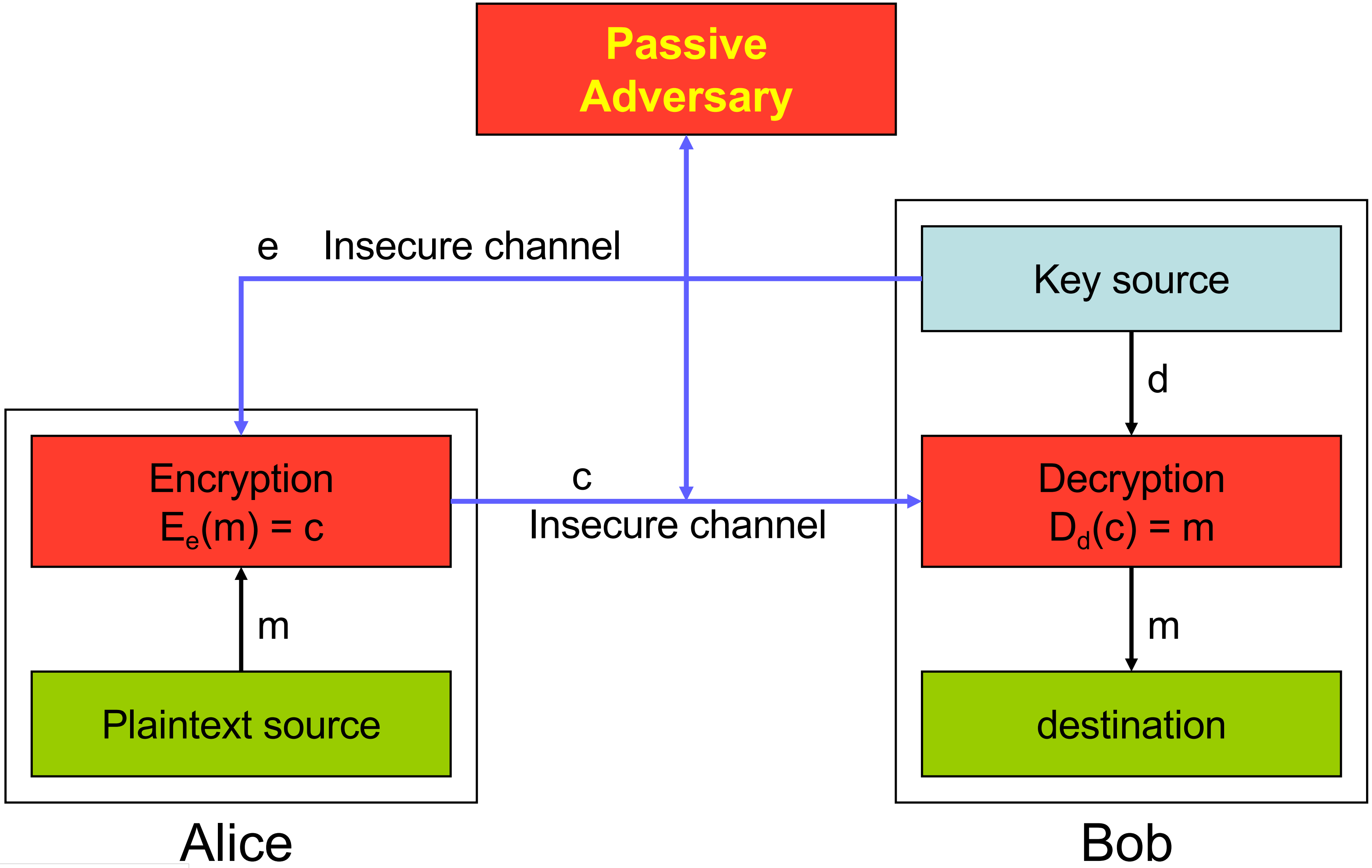


- Why do we use key?
  - Or why not use just a shared encryption function?

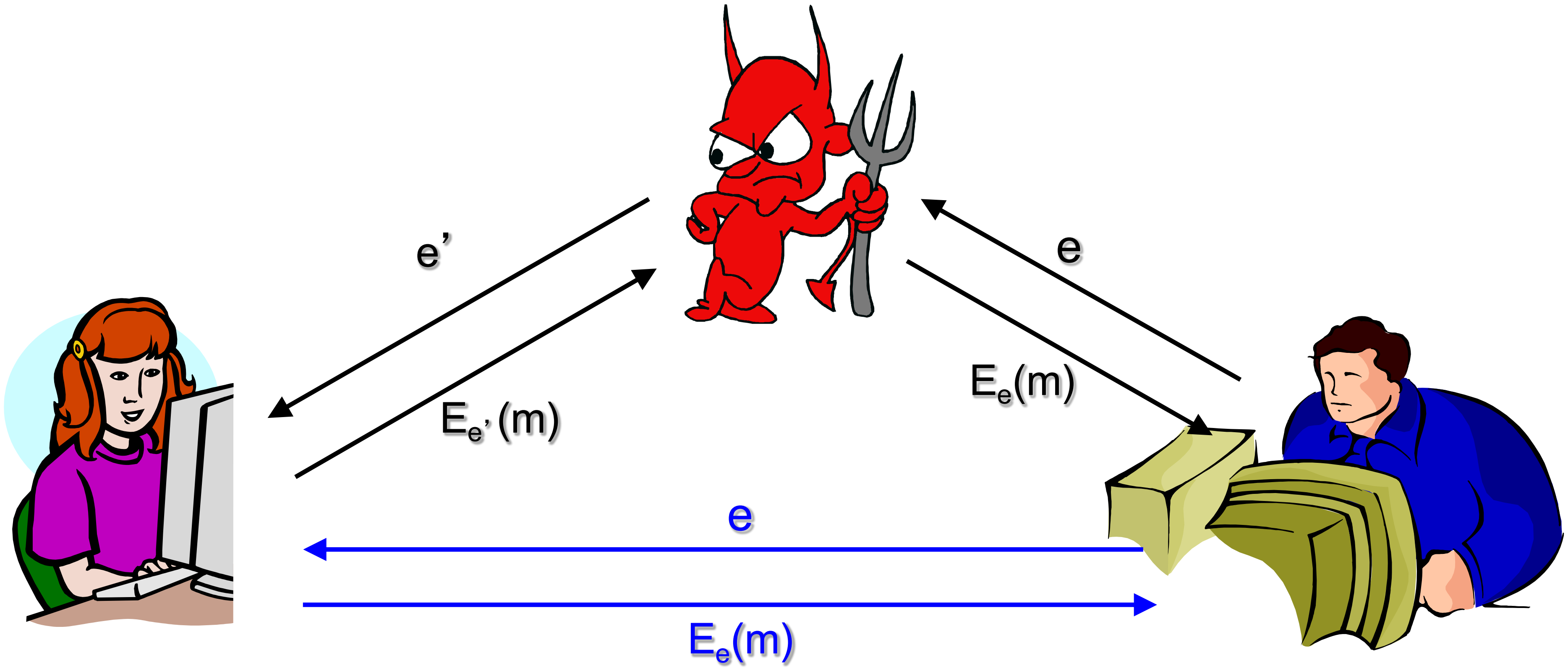
SKE with Secure channel



PKE with insecure channel



Public key should be authentic!



- Need to authenticate public keys

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: M ! S for A, kept private
  - 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

Digital Signature

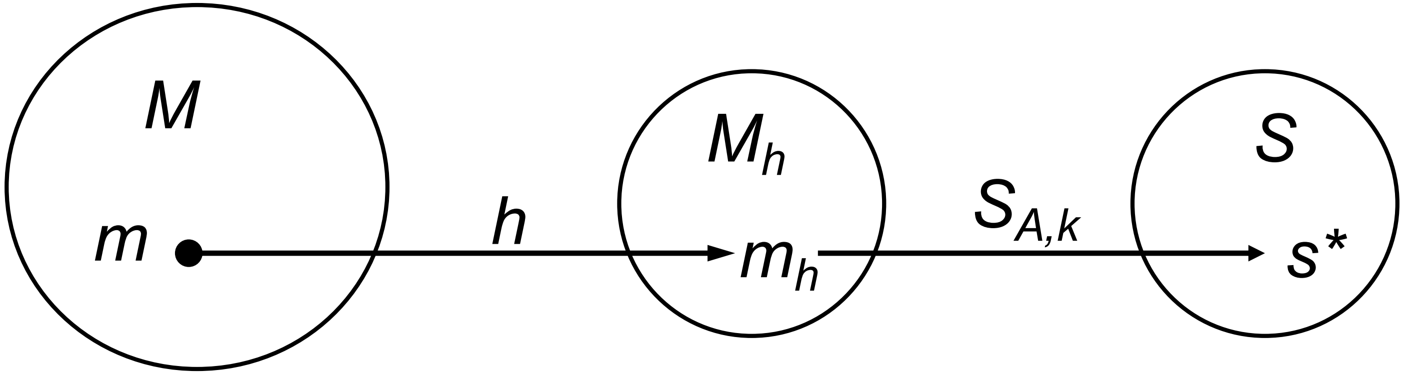
- Integrity
- Authentication
- Non-repudiation

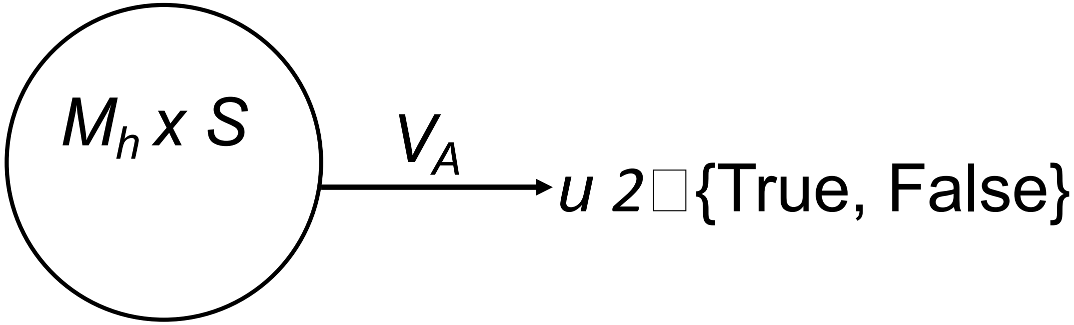
KAIST

Digital Signature with Appendix

- Schemes with appendix
  - Requires the message as input to verification algorithm
  - Rely on cryptographic hash functions rather than customized redundancy functions
  - DSA, ElGamal, Schnorr etc.

Digital Signature with Appendix





$s^* = S_{A,k}(m_h)$

$u = V_A(m_h, s^*)$

Authentication

- How to prove your identity?
  - Prove that you know a secret information
- When key K is shared between A and Server
  - A ➔ S: $HMAC_K(M)$ where M can provide freshness
  - Why freshness?
- Digital signature?
  - A ➔ S: $Sig_{SK}(M)$ where M can provide freshness
- Comparison?

Encryption and Authentication

- $E_K(M)$

- Redundancy-then-Encrypt: $E_K(M, R(M))$
- Hash-then-Encrypt: $E_K(M, h(M))$
- Hash and Encrypt: $E_K(M), h(M)$
- MAC and Encrypt: $E_{h1(K)}(M), HMAC_{h2(K)}(M)$
- MAC-then-Encrypt: $E_{h1(K)}(M, HMAC_{h2(K)}(M))$
- Encrypt-then-MAC: $E_{h1(K)}(M)$, $HMAC_{h2(K)}(E_{h1(K)}(M))$

Challenge-response authentication

- Alice is identified by a *secret* she possesses
  - *Bob* needs to know that Alice does indeed possess this secret
  - *Alice* provides ***response*** to a time-variant ***challenge***
  - Response depends on ***both*** secret and challenge

- Using
  - Symmetric encryption
  - One way functions

KAIST

Challenge Response using SKE

- Alice and Bob share a key K
- Taxonomy
  - ***Unidirectional*** authentication using ***timestamps***
  - ***Unidirectional*** authentication using ***random numbers***
  - ***Mutual*** authentication using ***random numbers***
- Unilateral authentication using timestamps
  - Alice $\rightarrow$ Bob: $E_K(t_A, B)$
  - Bob decrypts and verified that timestamp is OK
  - Parameter B prevents replay of same message in B $\rightarrow$ A direction

Challenge Response using SKE

- Unilateral authentication using random numbers
  - Bob → Alice: r_b
  - Alice → Bob: $E_K(r_b, B)$
  - Bob checks to see if r_b is the one it sent out
    - Also checks "*B*" - prevents reflection attack
  - r_b must be ***non-repeating***
- Mutual authentication using random numbers
  - Bob → Alice: r_b
  - Alice → Bob: $E_K(r_a, r_b, B)$
  - Bob → Alice: $E_K(r_a, r_b)$
  - Alice checks that r_a, r_b are the ones used earlier

Challenge-response using OWF

- Instead of encryption, used keyed MAC h_K
- Check: compute MAC from *known quantities,* and check with message
- SKID3
  - Bob $\rightarrow$ Alice: r_b
  - Alice $\rightarrow$ Bob: r_a, $h_K(r_a, r_b, B)$
  - Bob $\rightarrow$ Alice: $h_K(r_a, r_b, A)$

KAIST

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

Kerberos vs. PKI vs. IBE

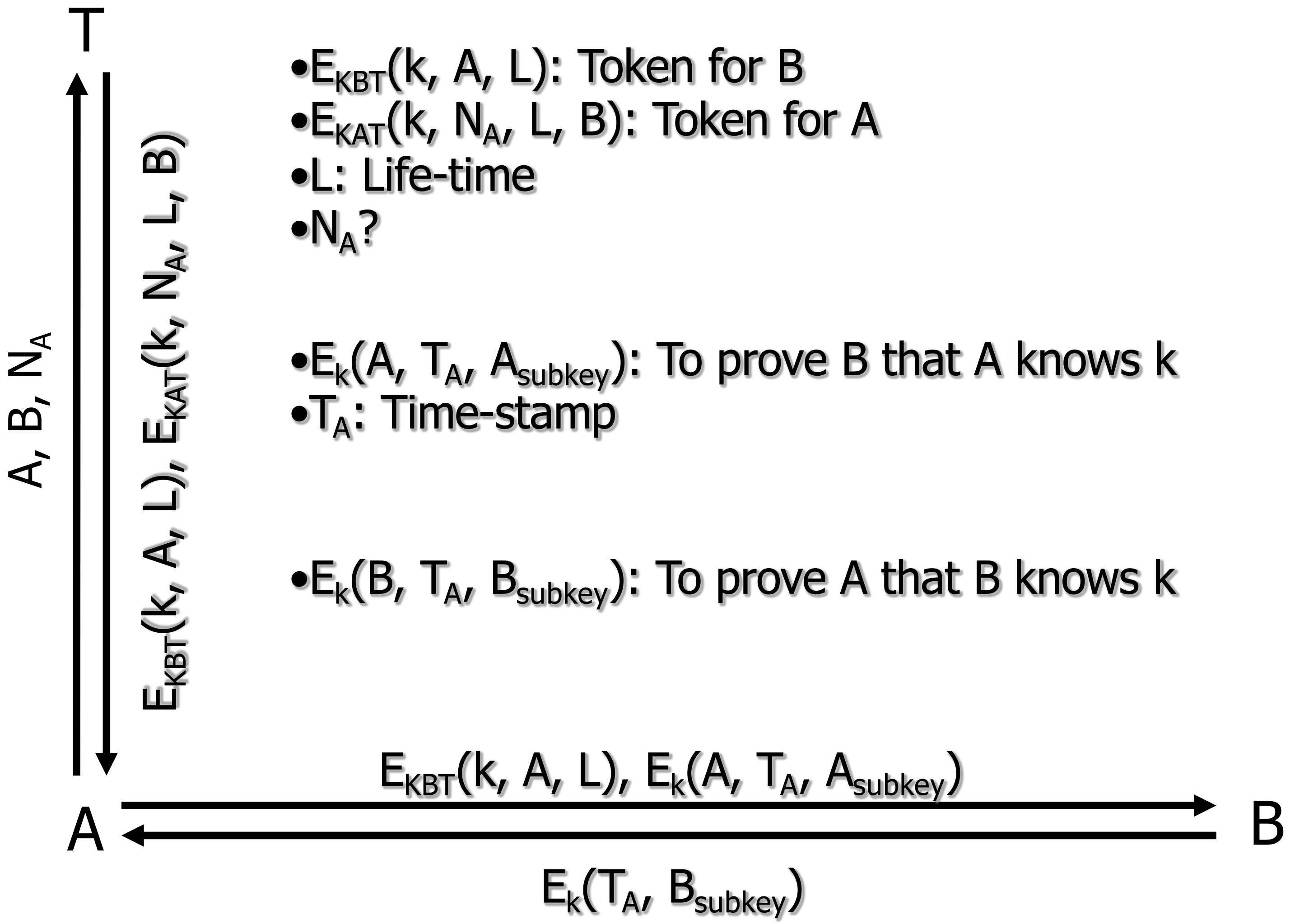
- Still debating ☺
- Let’s see one by one!

Kerberos (cnt.)

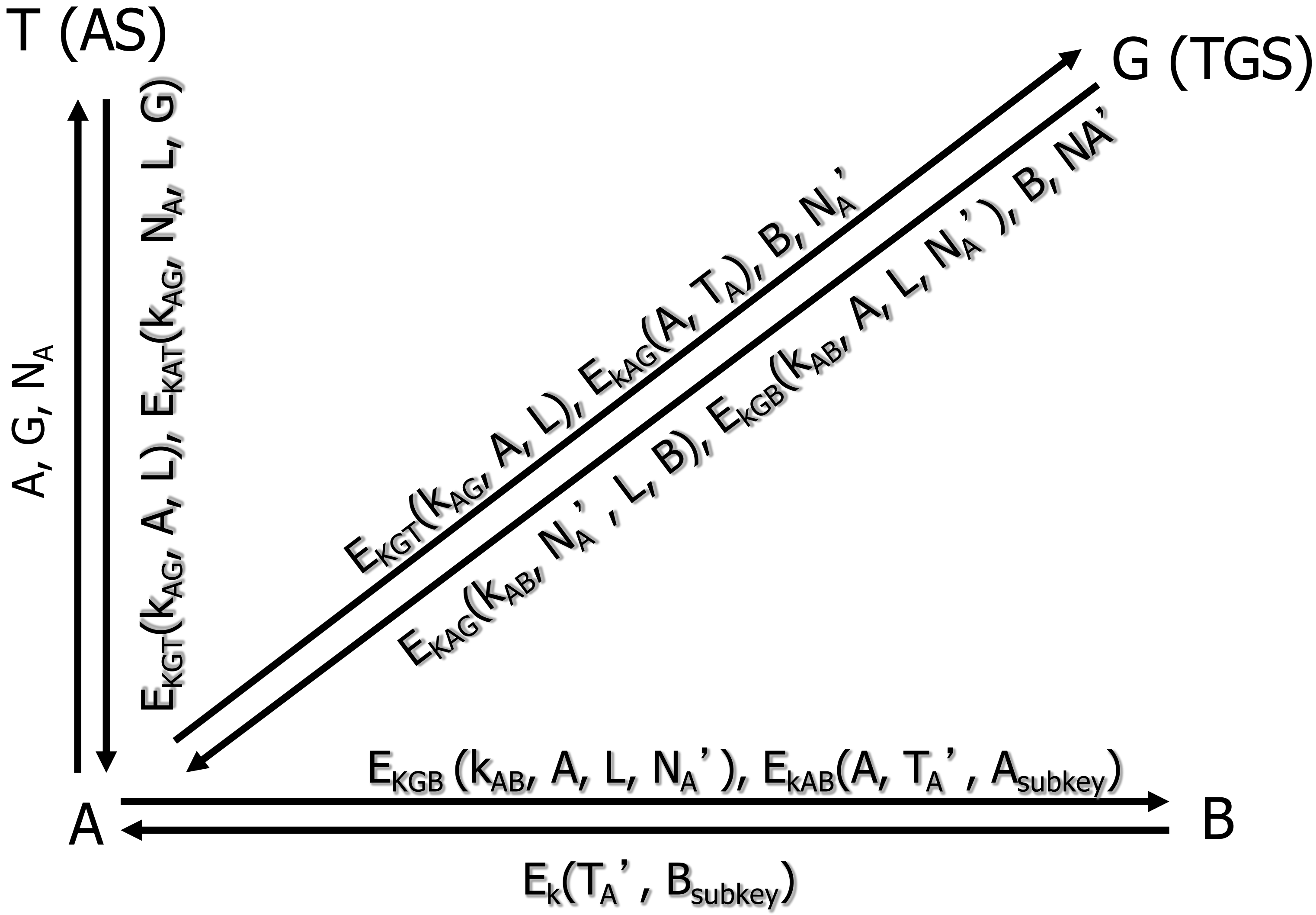
- $E_{KBT}(k, A, L)$: Token for B
- $E_{KAT}(k, N_A, L, B)$: Token for A
- L: Life-time
- N_A?

- $E_k(A, T_A, A_{subkey})$: To prove B that A knows k
- T_A: Time-stamp

- $E_k(B, T_A, B_{subkey})$: To prove A that B knows k



Kerberos (Scalable)



Public Key Certificate

- Public-key certificates are a vehicle
  - public keys may be stored, distributed or forwarded over unsecured media
- The objective
  - make one entity’ s public key available to others such that its authenticity and validity are verifiable.
- A public-key certificate is a data structure
  - data part
    - cleartext data including a public key and a string identifying the party (subject entity) to be associated therewith.
  - signature part
    - digital signature of a certification authority over the data part
    - binding the subject entity’ s identity to the specified public key.

CA

- a trusted third party whose signature on the certificate vouches for the authenticity of the public key bound to the subject entity
  - The significance of this binding must be provided by additional means, such as an attribute certificate or policy statement.
- the subject entity must be a unique name within the system (distinguished name)
- The CA requires its own signature key pair, the authentic public key.
- Can be off-line!

ID-based Cryptography

- No public key
- Public key = ID (email, name, etc.)
- PKG
  - Private key generation center
  - SK_{ID} = $PKG_S(ID)$
  - PKG’s public key is public.
  - distributes private key associated with the ID
- Encryption: C= $E_{ID}(M)$
- Decryption: $D_{SK}(C)$ = M

KAIST

Discussion (PKI vs. Kerberos vs. IBE)

- On-line vs. off-line TTP
  - Implication?
- Non-reputation?
- Revocation?
- Scalability?
- Trust issue?

Point-to-Point Key Update

- Key Transport with one pass
  - $A \rightarrow B: E_K(r_A)$
  - Implicit key authentication
  - Additional field
    - timestamp, sequence number: freshness
    - redundancy: explicit key authentication, message modification
    - target identifier: prevent undetectable message replay
  - Hence $A \rightarrow B: E_K(r_A, t_A, B)$
  - Mutual authentication: $B \rightarrow A: E_K(r_B, t_B, A)$: $K = f(r_A, r_B)$
- Key Transport with challenge-response
  - $B \rightarrow A: n_B$: for freshness
  - $A \rightarrow B: E_K(r_A, n_A, n_B, B)$
  - $B \rightarrow A: E_K(r_B, n_B, n_A, A)$
  - Cannot provide PFS
- Authenticated Key Update Protocol
  - $A \rightarrow B: r_A$
  - $B \rightarrow A: (B, A, r_A, r_B), h_K(B, A, r_A, r_B)$
  - $A \rightarrow B: (A, r_B), h_K(A, r_B)$
  - $W = h'_{K'}(r_B)$

KAIST

Key Transport using PKC

- Needham-Schroeder
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A)$
    - $B \rightarrow A: P_A(k_2, B)$
    - $A \rightarrow B: P_B(k_2)$
  - Properties: Mutual authentication, mutual key transport

- Modified NS
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A, r_1)$
    - $B \rightarrow A: P_A(k_2, r_1, r_2)$
    - $A \rightarrow B: r_2$
  - Removing third encryption

KAIST

Key Transport using PKC

- Needham-Schroeder
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A)$
    - $B \rightarrow A: P_A(k_1, k_2, B)$
    - $A \rightarrow B: P_B(k_2)$
- Modified NS
  - Algorithm
    - $A \rightarrow B: P_B(k_1, A, r_1)$
    - $B \rightarrow A: P_A(k_2, r_1, r_2)$
    - $A \rightarrow B: r_2$
  - Removing third encryption
- Encrypting signed keys
  - $A \rightarrow B: P_B(k, t_A, S_A(B, k, t_A))$
  - Data for encryption is too large
- Encrypting and signing separately
  - $A \rightarrow B: P_B(k, t_A), S_A(B, k, t_A)$
  - Acceptable only if no information regarding plaintext data can be deduced from the signature
- Signing encrypted keys
  - $A \rightarrow B: t_A, P_B(A, k), S_A(B, t_A, P_B(A, k))$
  - Prevent the above problem
  - Can provide mutual authentication

KAIST

Combining PKE and DS

- Assurances of X.509 strong authentication
  - identity of A, and the token received by B was constructed by A
  - the token received by B was specifically intended for B;
  - the token received by B has "freshness"
  - the mutual secrecy of the transferred key.
- X.509 strong authentication
  - $D_A=(t_A, r_A, B, data_1, P_B(k_1)), D_B=(t_B, r_B, A, r_A, data_2, P_A(k_2))$,
  - $A \rightarrow B$: $cert_A, D_A, S_A(D_A)$
  - $B \rightarrow A$: $cert_B, D_B, S_B(D_B)$
- Comments
  - Since protocol does not specify inclusion of an identifier within the scope of the encryption P_B within D_A, one cannot guarantee that the signing party actually knows (or was the source of) plaintext key

Attack strategies and classic flaws

- "man-in-the-middle" attack on unauthenticated DH
- Reflection attack
  - Original protocol

1. $A \rightarrow B : r_A$
2. $B \rightarrow A : E_k(r_A, r_B)$
3. $A \rightarrow B : r_B$

  - Attack

1. $A \rightarrow E : r_A$
2. $E \rightarrow A : r_A$: Starting a new session
3. $A \rightarrow E : E_k(r_A, r_A')$: Reply of (2)
4. $E \rightarrow A : E_k(r_A, r_A')$: Reply of (1)
5. $A \rightarrow E : r_A'$

  - prevented by using different keys for different sessions

KAIST

Attack strategies and classic flaws

- Interleaving attacks
  - To provide freshness and entity authentication
  - Flawed protocol
    1. $A \rightarrow B : r_A$
    2. $B \rightarrow A : r_B, S_B(r_B, r_A, A)$
    3. $A \rightarrow B : r_A', S_A(r_A', r_B, B)$
  - Attack
    1. $E \rightarrow B : r_A$
    2. $B \rightarrow E : r_B, S_B(r_B, r_A, A)$
    3. $E \rightarrow A : r_B$
    4. $A \rightarrow E : r_A', S_A(r_A', r_B, B)$
    5. $E \rightarrow B : r_A', S_A(r_A', r_B, B)$
  - Due to symmetric messages (2), (3)

KAIST