EE515
Security of Emerging Systems

Yongdae Kim

SysSec@KAIST

Admin

- Homepage
  - http://security101.kr
- Survey
  - Paper presentation survey (will be sent this week)
  - Find your group members and discuss about projects

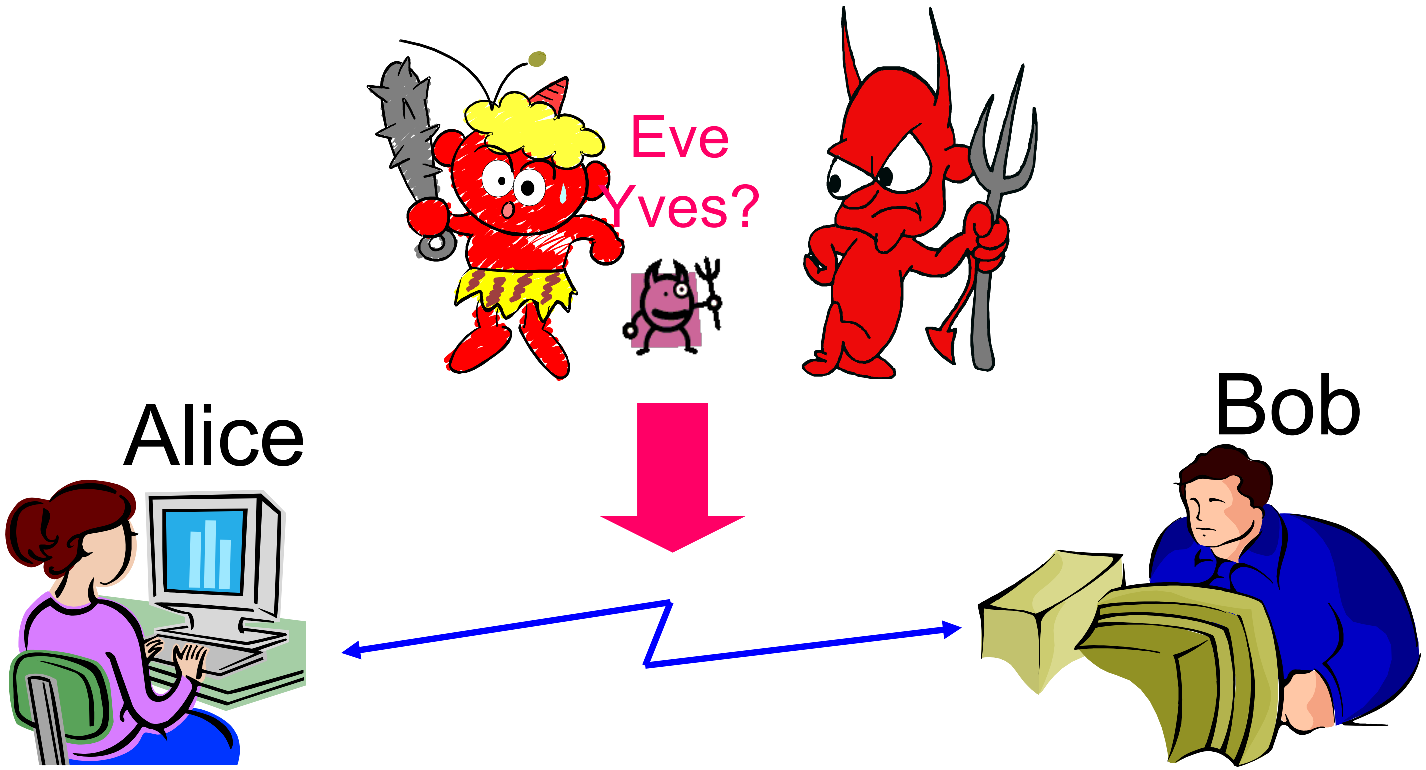
Security theater is the practice of

- investing in countermeasures intended to provide the feeling of improved security
- while doing little or nothing to actually achieve it

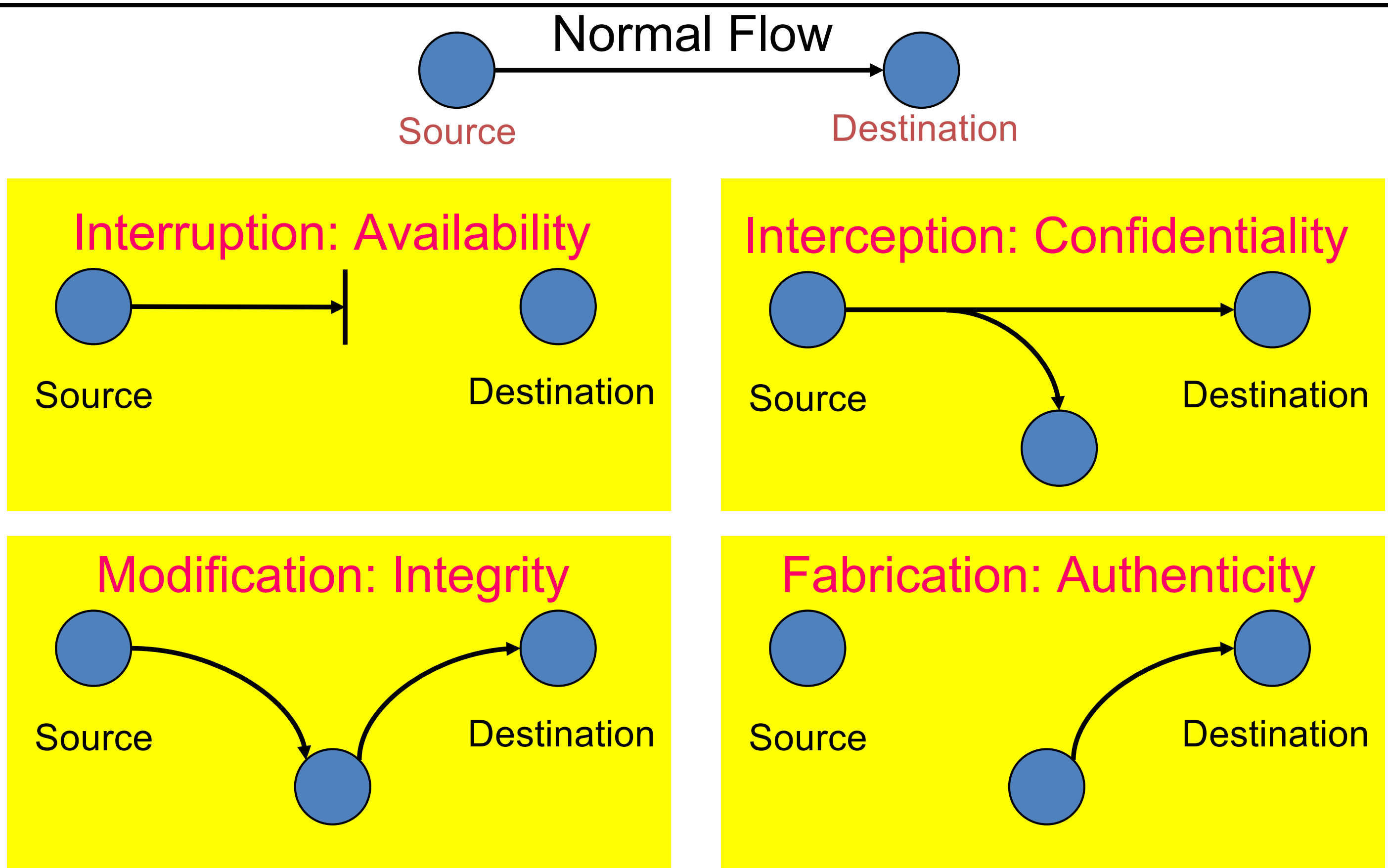
- Bruce Schneier

Basic Cryptography

The Main Players



Attacks

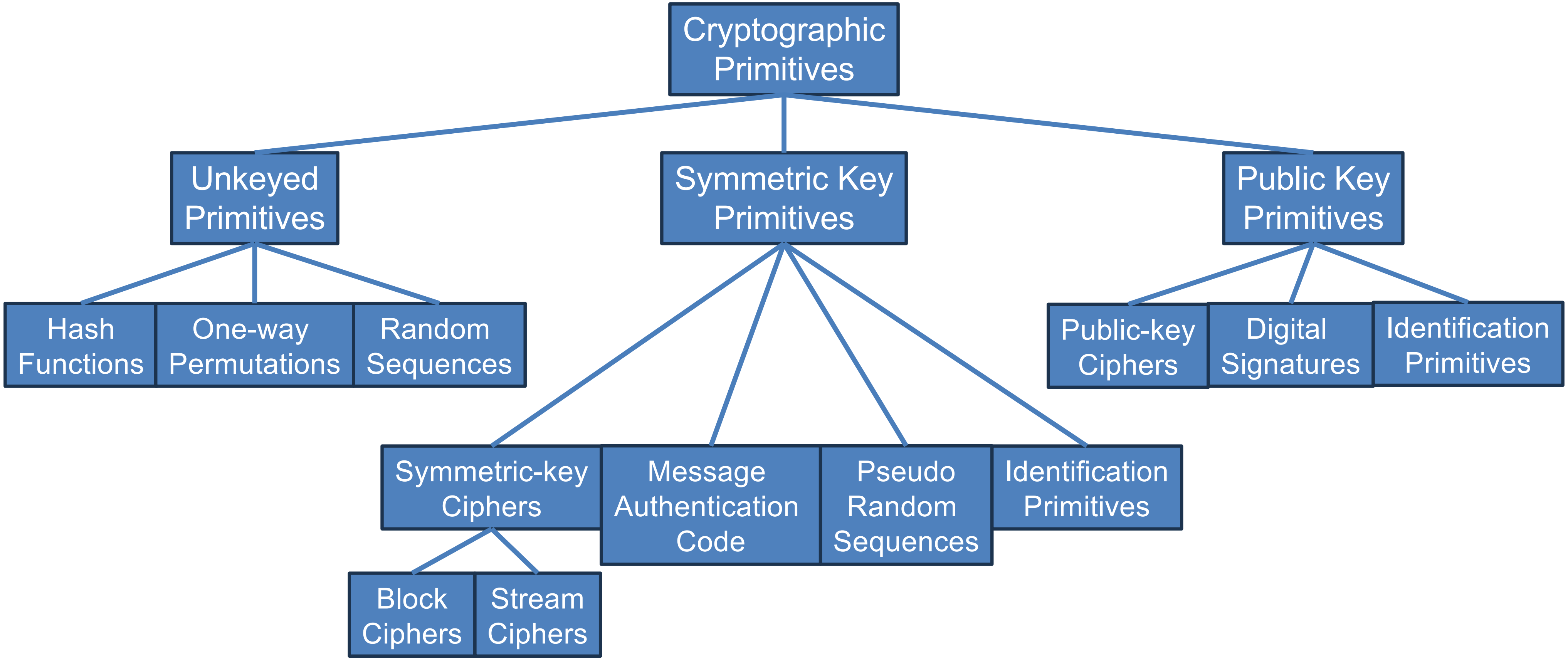

Normal Flow
Source
Destination
Interruption: Availability
Source
Destination
Interception: Confidentiality
Source
Destination
Modification: Integrity
Source
Destination
Fabrication: Authenticity
Source
Destination

Taxonomy of Attacks

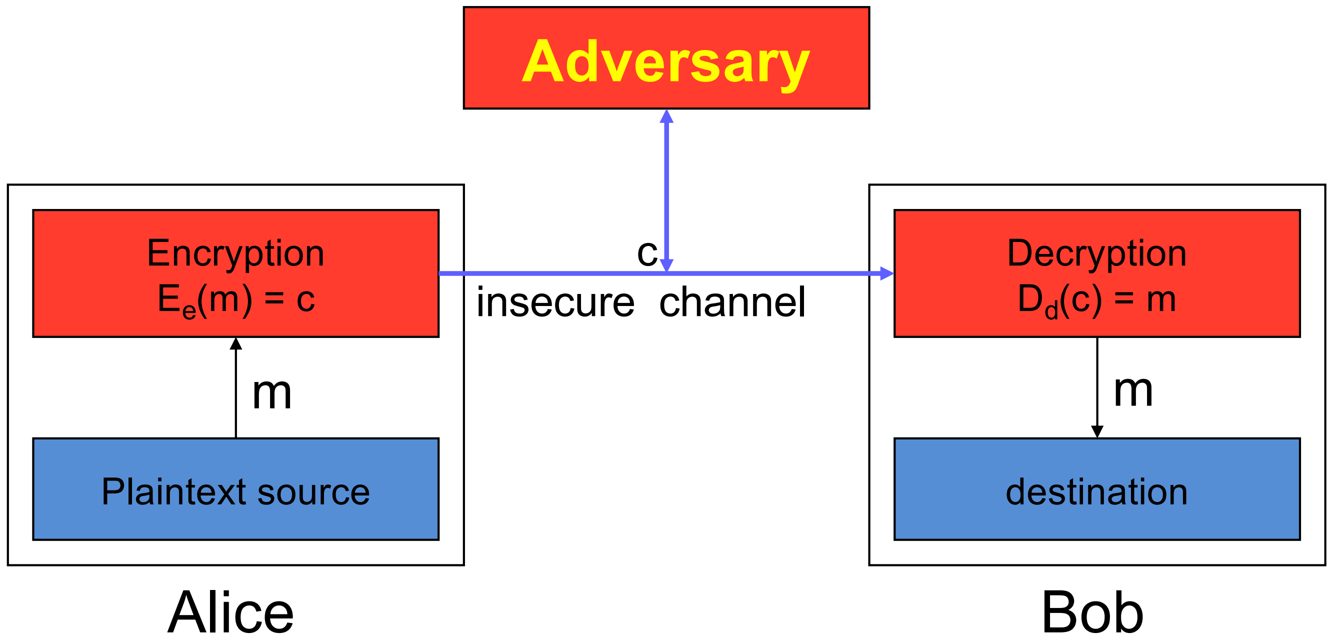
- Passive attacks
  - Eavesdropping
  - Traffic analysis

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

Cryptographic Primitives

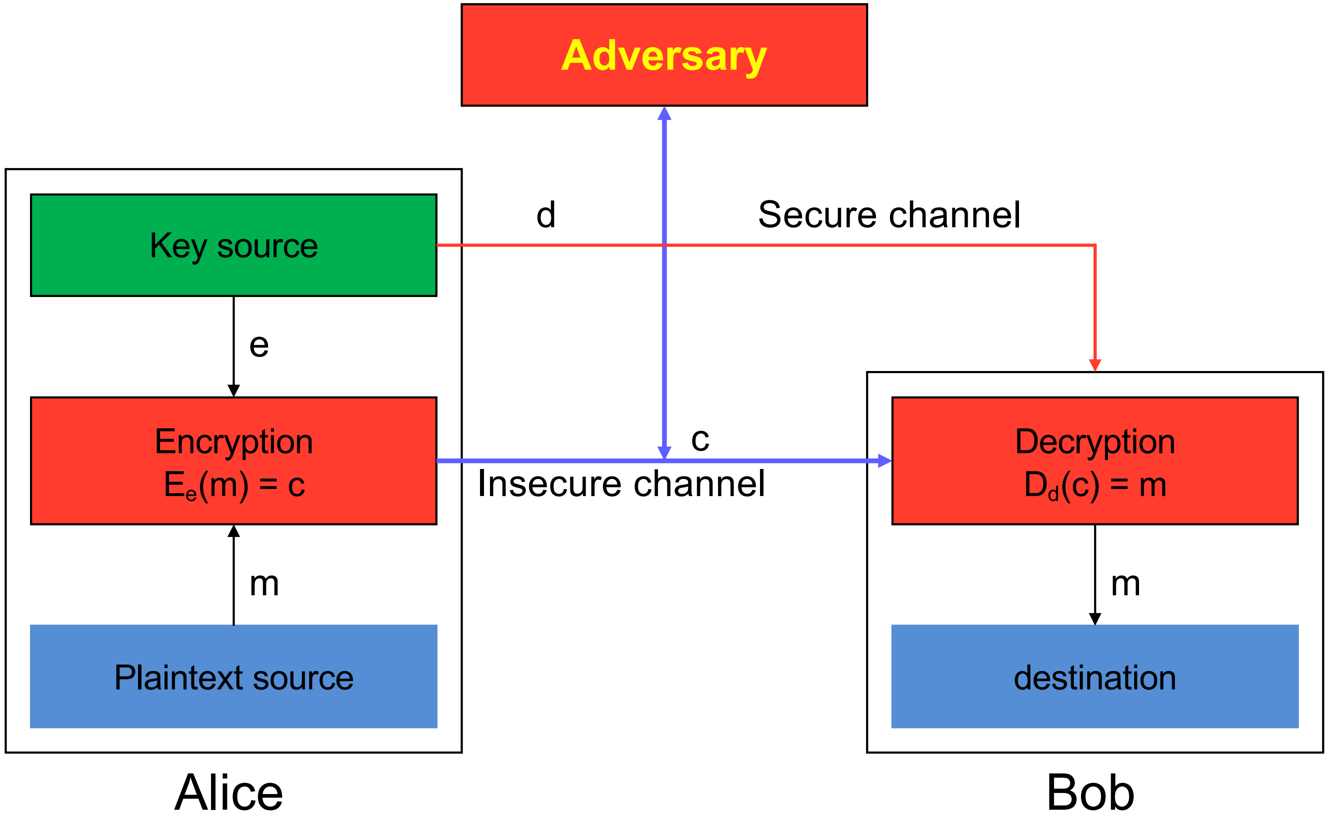


Encryption

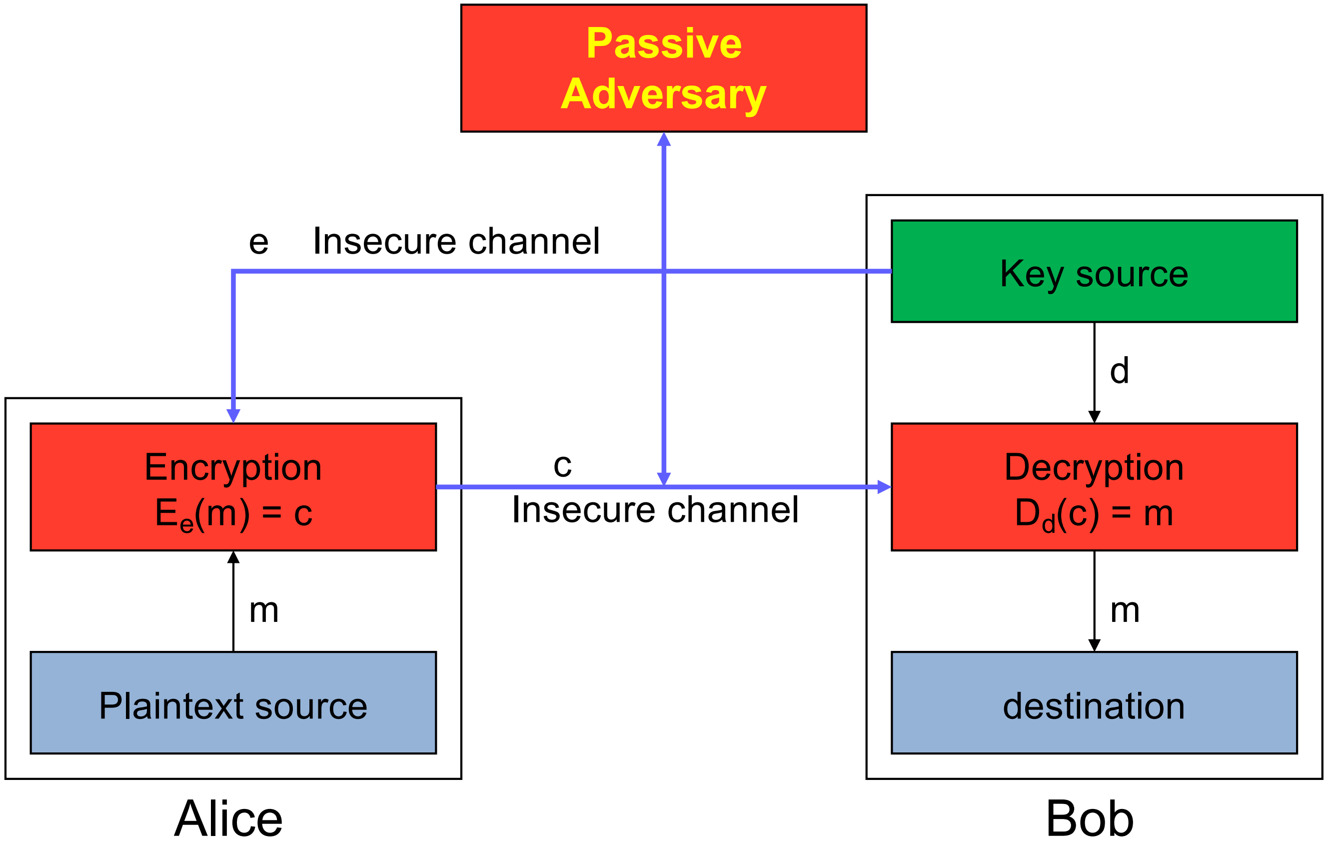


- Why do we use key?
- Or why not use just a shared encryption function?

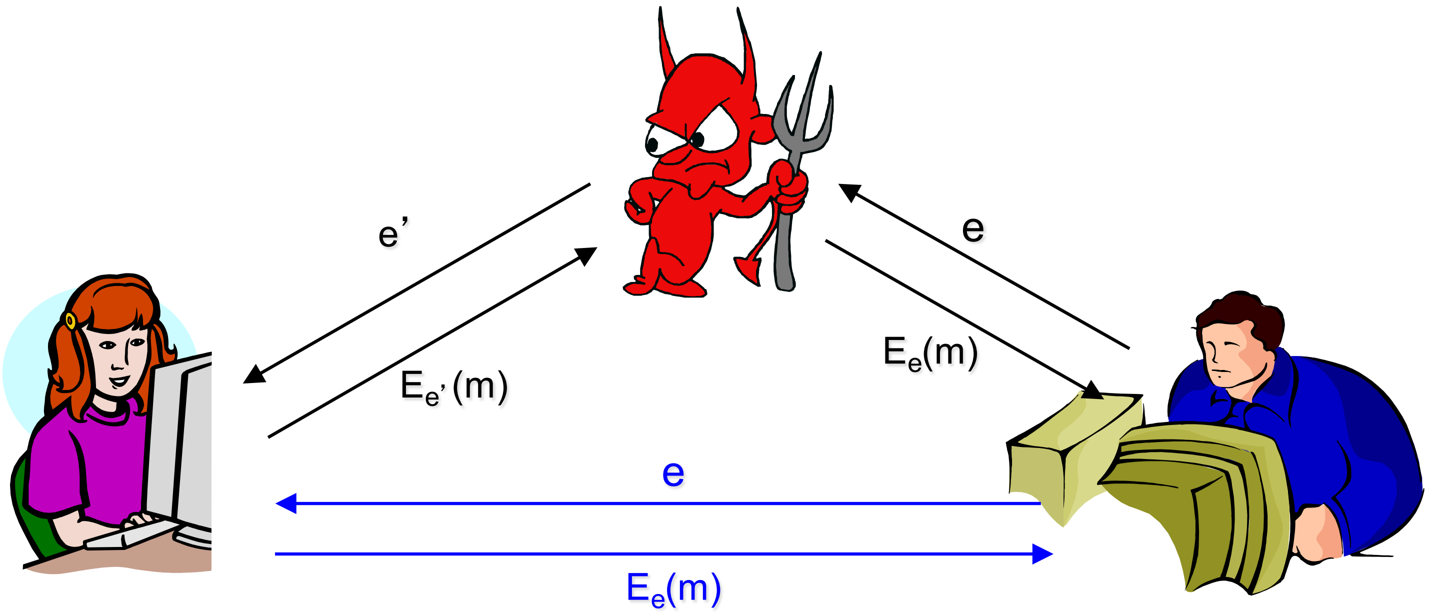
Symmetric Key Encryption



Public Key Encryption



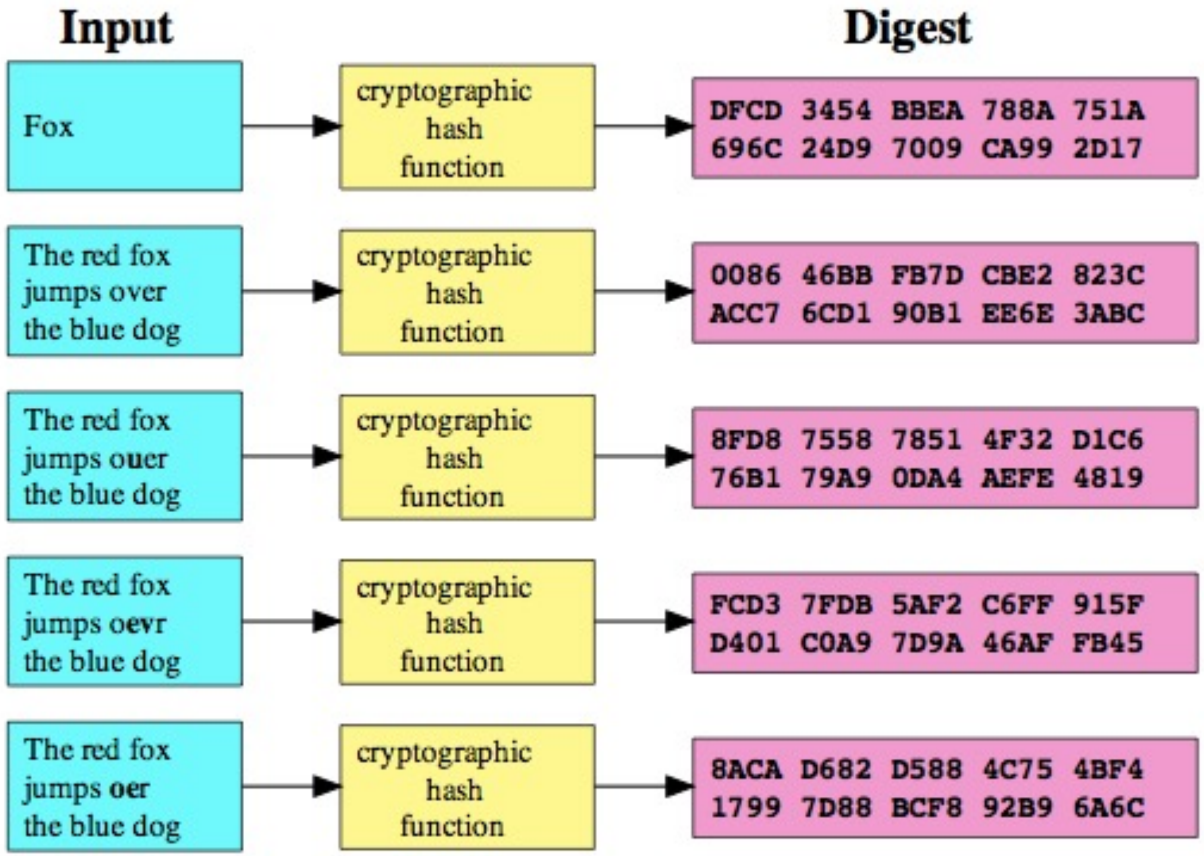
Public Key should be authentic!



Hash Function

- A hash function is a function h satisfying
  - h:$\{0, 1\}^*$ ➔ $\{0, 1\}^k$ (Compression)
- A cryptographic hash function is a hash function satisfying
  - It is easy to compute y=h(x) (ease of computation)
  - For a given y, it is hard to find x′ such that h(x′)=y. (onewayness)
  - It is hard to find x and x′ such that h(x)=h(x′) (collision resistance)
- Examples: SHA-1, MD-5, SHA-256, …

Randomness of a Hash Function



Applications of Hash Functions

❖ 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 WalkerNews.net

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

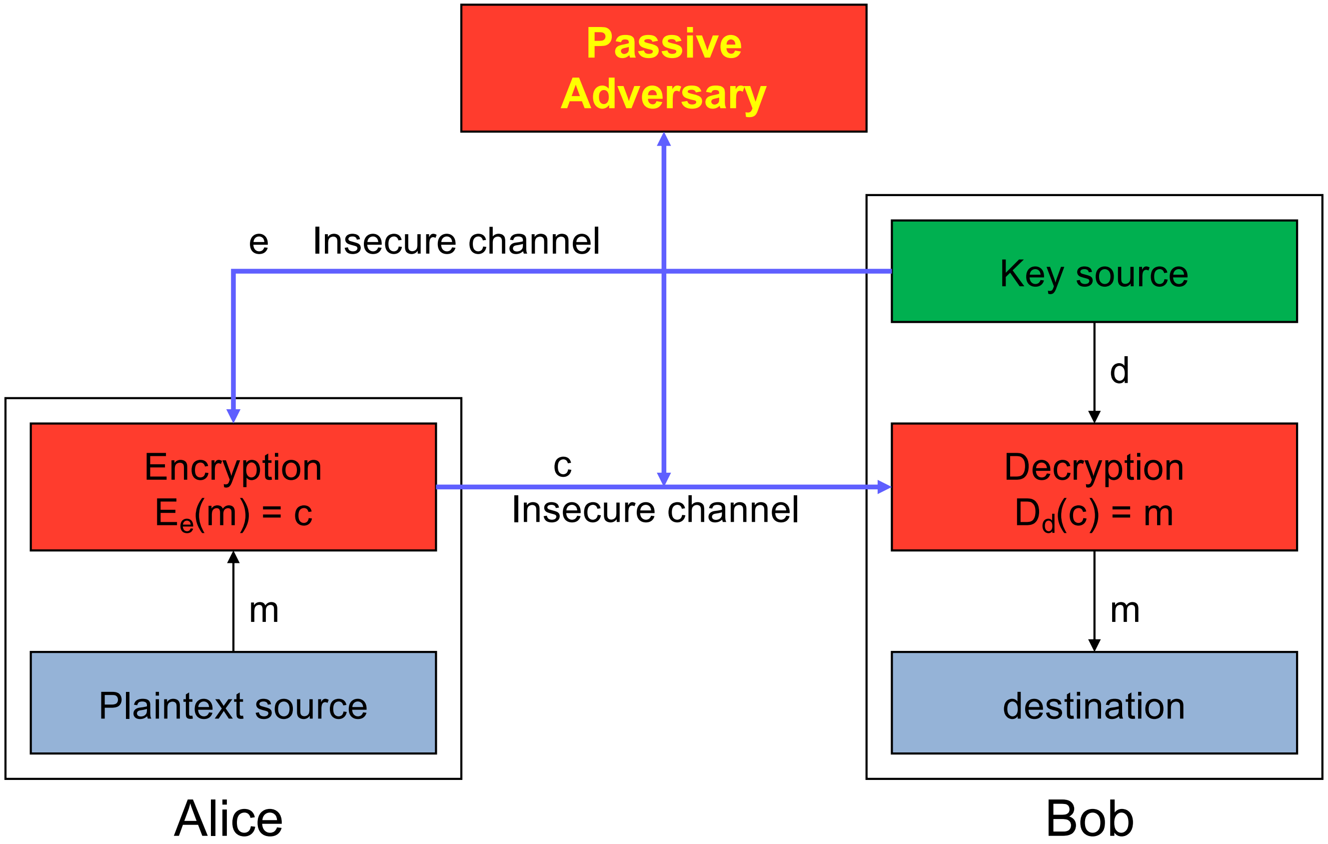
MAC construction from Hash Function

- 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 $M = H_k(x)$

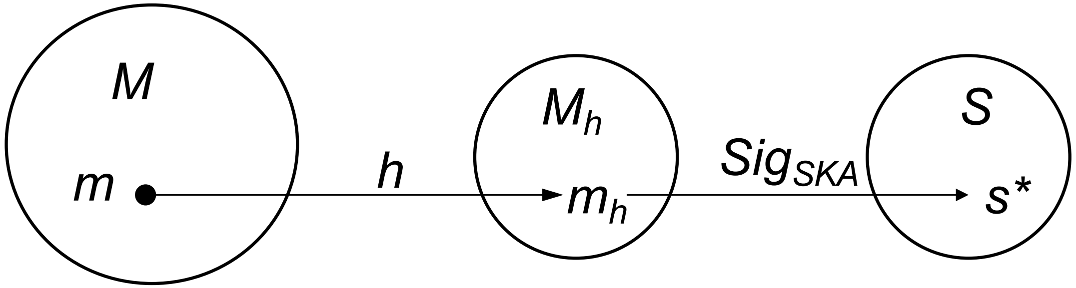
Public Key Encryption

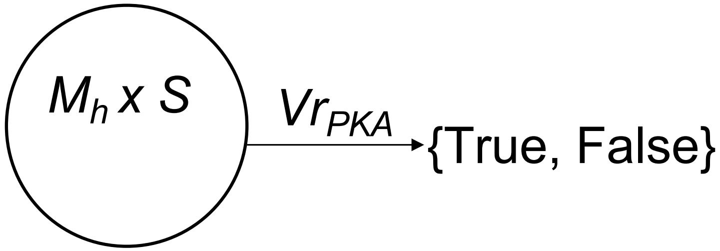


Digital Signatures

- Unforgeability
- Integrity
- Authentication
- Non-repudiation

Digital Signature with Appendix





$s^* = Sig_{SKA}(m_h)$

$u = Vr_{PKA}(m_h, s^*)$

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: $C, HMAC_{h2(K)}(C)$, where $C=E_{h1(K)}(M)$

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 key encryption
  - Public key encryption
  - MAC
  - Digital signatures

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 → Bob: $E_K(t_A, B)$
  - Bob decrypts and verified that timestamp is OK
  - Parameter B prevents replay of same message in B → 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 MAC

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

Challenge-Response using PKE and DS

- Mutual Authentication based on PK decryption
  - Alice → Bob: $P_B(r_A, B)$
  - Bob → Alice: $P_A(r_A, r_B)$
  - Alice → Bob: r_B
- Timestamp-based unilateral authentication using DS
  - Alice → Bob: $cert_A, t_A, B, S_A(t_A, B)$
  - Bob checks:
    - Timestamp OK
    - Identifier "B" is its own
    - Signature is valid (after getting public key of Alice using certificate)
- Mutual Authentication using DS
  - Bob → Alice: r_B
  - Alice → Bob: $cert_A, r_A, B, S_A(r_A, r_B, B)$
  - Bob → Alice: $cert_B, A, S_B(r_A, r_B, A)$

Key Establishment, Management

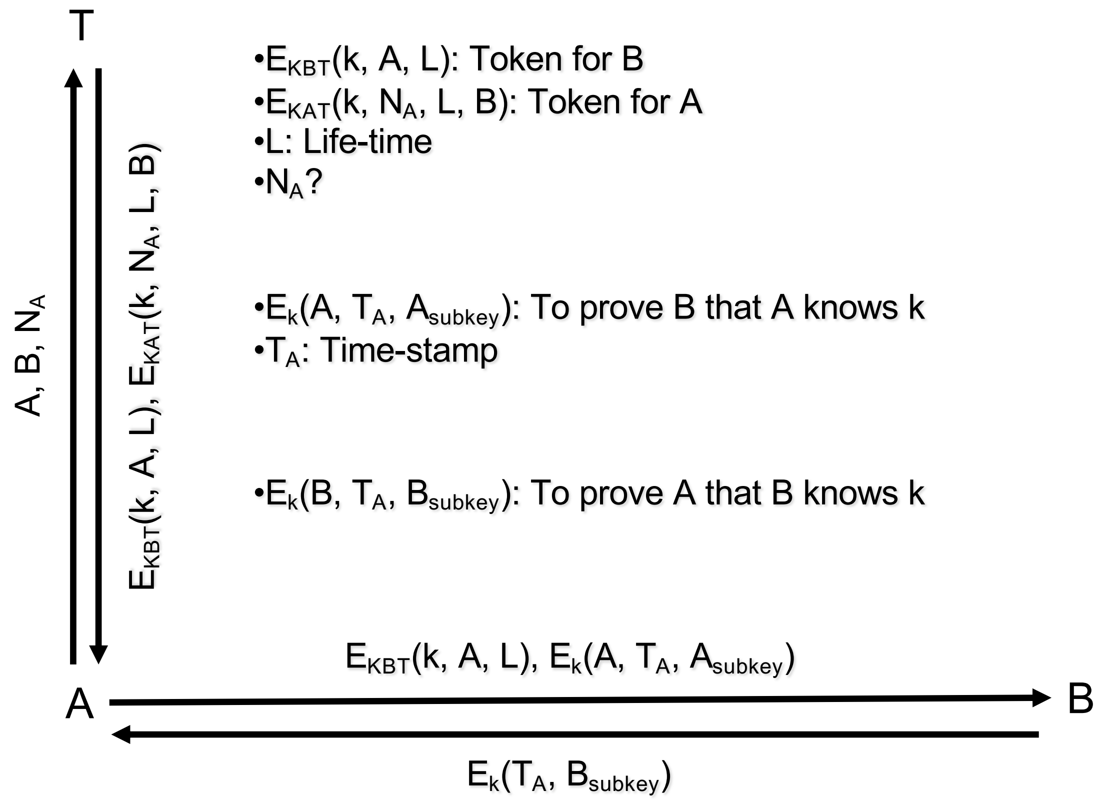
- Key establishment
  - Process to whereby a shared secret key becomes available to two or more parties
  - Subdivided into key agreement and key transport.

- 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

- Two people who never met before
  - Can mutually authenticate each other
  - Can share a secret key

Kerberos

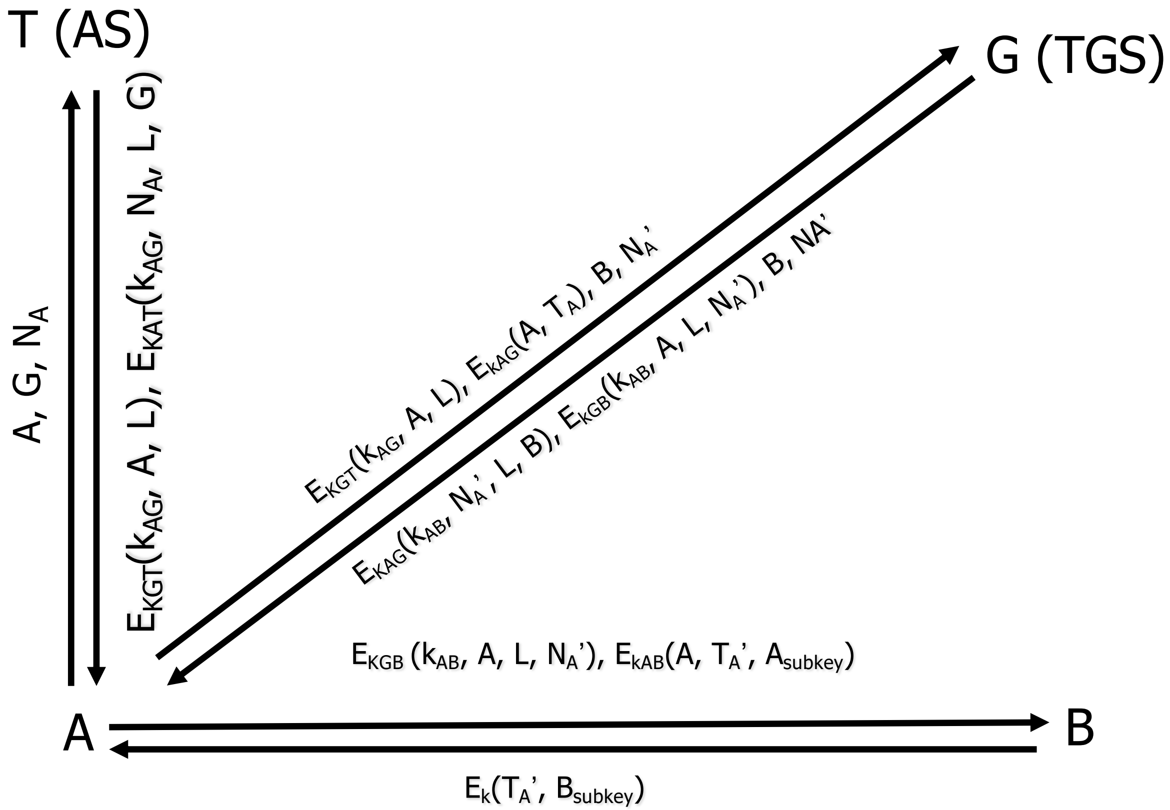


- $E_{KBT}(k, A, L)$: Token for B
- $E_{KAT}(k, N_A, L, B)$: Token for A
- L: Life-time
- N_A?

- $E_k(A, T_A, A_{subkey})$: 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.

Certificate Authority

- 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!

Verifying Public Key Certificate

1. (One-time) acquire the authentic public key of the certification authority.
2. Obtain an identifying string uniquely identifying the intended party A
3. Acquire over some unsecured channel A's public-key certificate and agreeing with the previous identifying string.
4. (a) Verify the current date and time against the validity period (if any) in the certificate, relying on a local trusted time/day-clock;
   (b) Verify the current validity of the CA's public key itself;
   (c) Verify the signature on A's certificate, using the CA's public key;
   (d) Verify that the certificate has not been revoked.
5. If all checks succeed, accept the public key in the certificate as A's authentic key.

X.509 Strong Two-way Authentication

- Let D_A= (t_A, r_A, B, $data_1$*, $P_B(k_1)$*) and D_B=(t_B, r_B, A, r_A, $data_2$ ∗, $P_A(k_2)$∗).

- A → B: $cert_A$, D_A, $S_A(D_A)$
- B → A: $cert_B$, D_B, $S_B(D_B)$

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: $D_{SK}(C) = M$

Discussion (PKI vs. Kerberos vs. IBE)

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

Questions?

- Yongdae Kim
  - email: yongdaek@kaist.ac.kr
  - Home: http://syssec.kaist.ac.kr/~yongdaek
  - Facebook: https://www.facebook.com/y0ngdaek
  - Twitter: https://twitter.com/yongdaek
  - Google "Yongdae Kim"