Information Security

Ch10: Key management & Other Public Key Cryptosystems

Howon Kim

2022.5

- 정보보호 및 사물지능(AIoT) 연구실 http://infosec.pusan.ac.kr
- 블록체인 플랫폼 연구센터 http://itrc.pusan.ac.kr
- 지능형 융합보안대학원 http://aisec.pusan.ac.kr

정보보호 및 사물지능 연구실
Information Security & AIoT (AI of Things)

블록체인 플랫폼 연구센터
BLOCKCHAIN PLATFORM RESEARCH CENTER

부산대학교 융합보안대학원
PNU GRADUATE SCHOOL OF CONVERGENCE SECURITY

Agenda

- Key Management
- Diffie-Hellman Key Exchange

Key Management

- Public-key encryption helps address key distribution problems
  - Here, there are two aspects to use the PKC in this regard:
    - The distribution of public keys
    - The use of public-key encryption to distribute secret keys

Distribution of Public Keys

- can be considered as using one of:
  - public announcement
  - publicly available directory
  - public-key authority
  - public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
  - eg. append PGP keys to email messages or post to news groups, email list
- major weakness is forgery
  - Anyone can forge such a public announcement
  - That is, some user could pretend to be user A and send a public key to another participant or broadcast such a public key
  - Until such time as user A discovers the forgery and alerts other participants, the forger is able to read all encrypted messages intended for A and can use the forged keys for authentication

Public Announcement

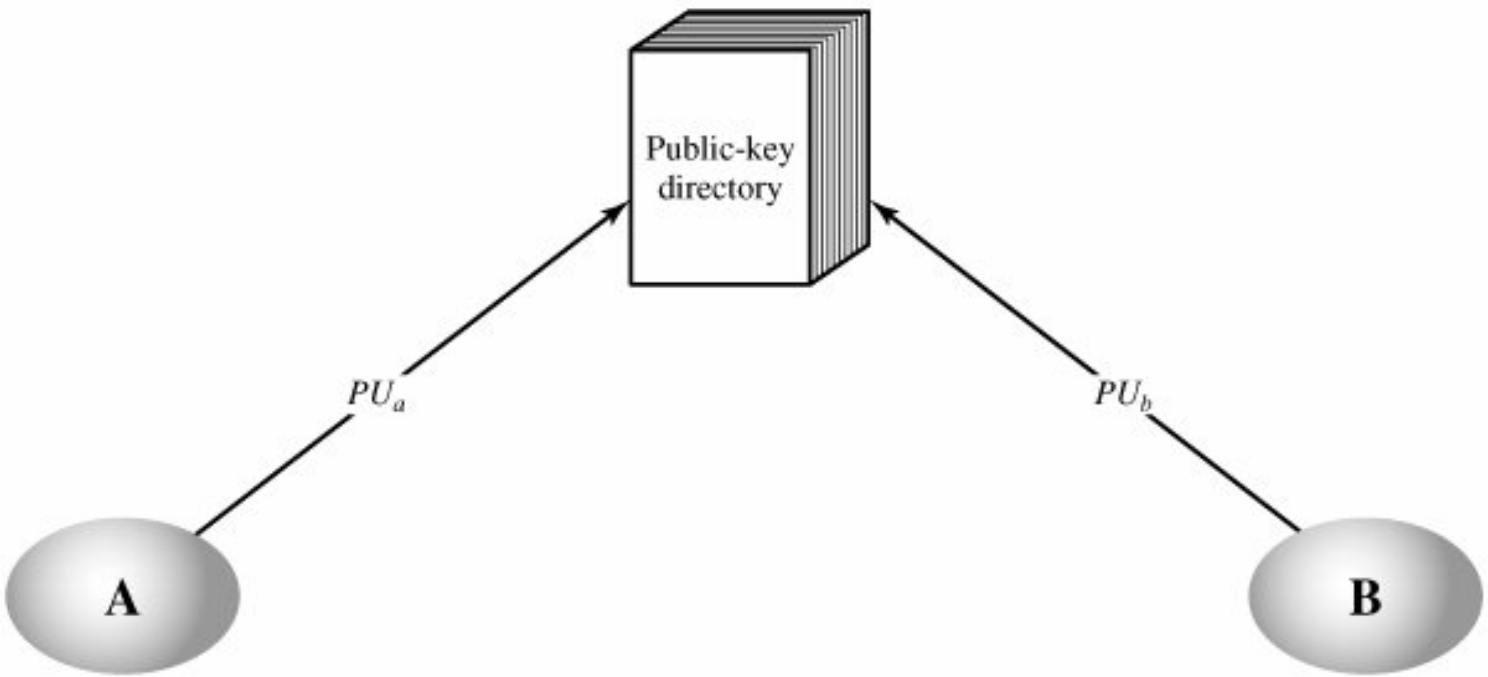
- Uncontrolled key distribution:

Publicly Available Directory

- We can obtain greater security by registering public keys with a public directory
- directory must be trusted with properties:
  - contains {name,public-key} entries
  - participants register securely with directory
  - participants can replace key at any time
  - directory is periodically published
  - directory can be accessed electronically
- still vulnerable to tampering or forgery

Publicly Available Directory

- Public key publication

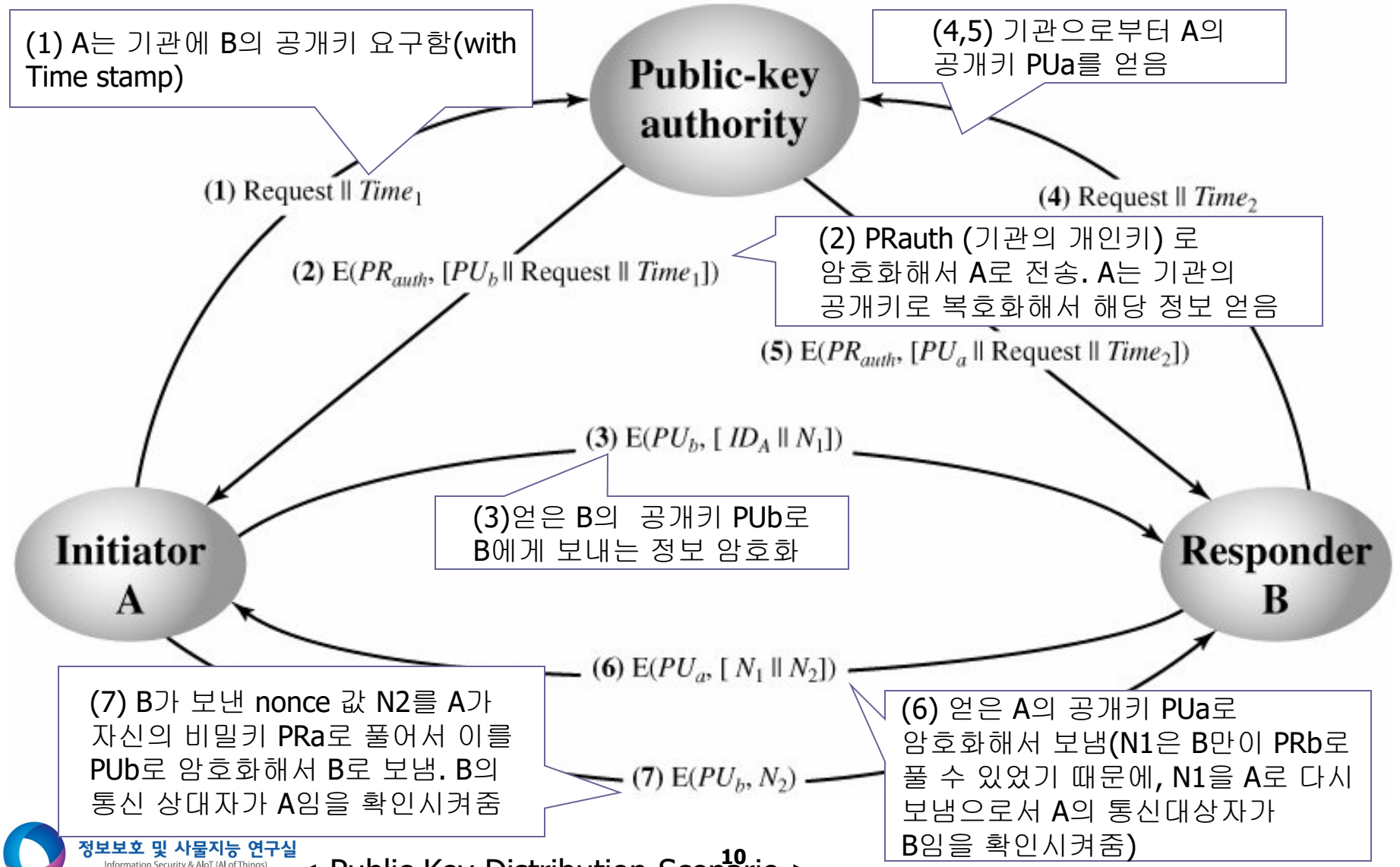


< Public Key Publication>

Public-Key Authority

- It improves security by tightening control over distribution of keys from directory
- It assumes that a central authority maintains a dynamic directory of public keys of all participants
  - In addition, each participant reliably knows the public key of the authority and only the authority knows the corresponding private key.
- Users interact with directory to obtain any desired public key securely
  - does require real-time access to directory when keys are needed

Public-Key Authority



< Public Key Distribution Scenario >

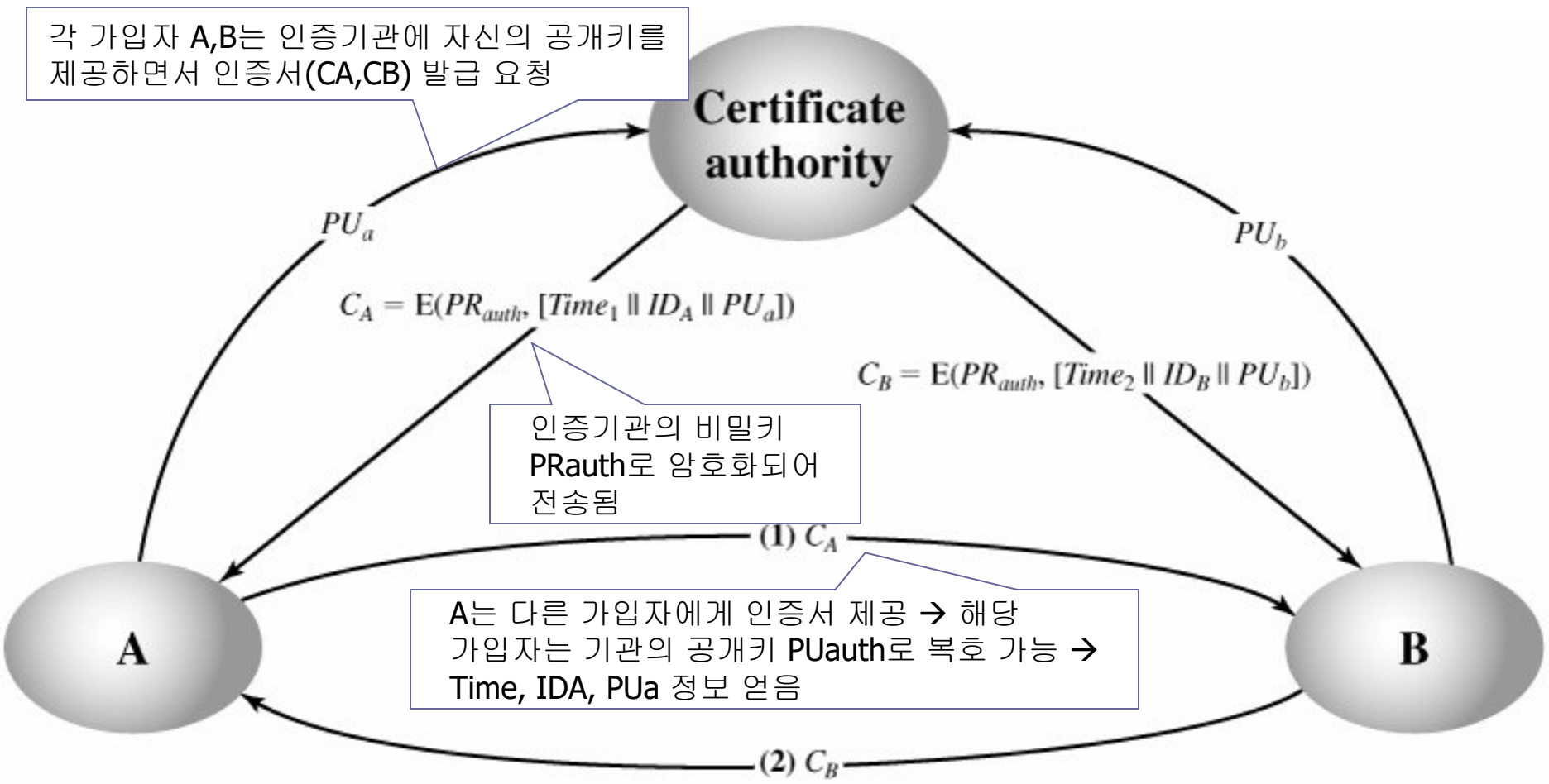
Public-Key Certificates

- Weakness of the public key authority:
  - Public key authority can be a bottleneck in the system
  - Also, the directory of names and public keys maintained by the authority is vulnerable to tampering.

Public-Key Certificates

- Certificates allow key exchange without real-time access to public-key authority
- A certificate binds **identity** to **public key**
  - usually with other information such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- can be verified by anyone who knows the public-key authoritie's public-key

Public-Key Certificates



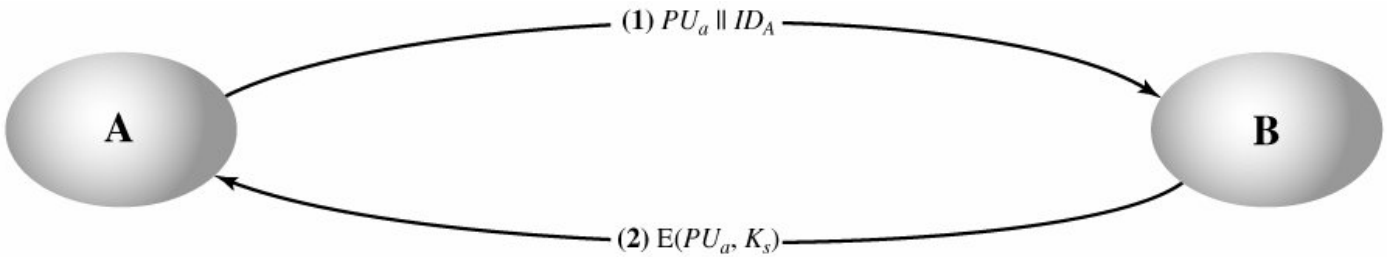
< Exchange of Public Key Certificates >

Distribution of Secret Keys Using PKC

- For channel encryption, few people use the PKC for their secure communications
  - Because public-key algorithms are slow
  - so usually want to use private-key encryption to protect message contents
- So public-key encryption provides for the distribution of secret keys to be used for conventional encryption

Simple Secret Key Distribution

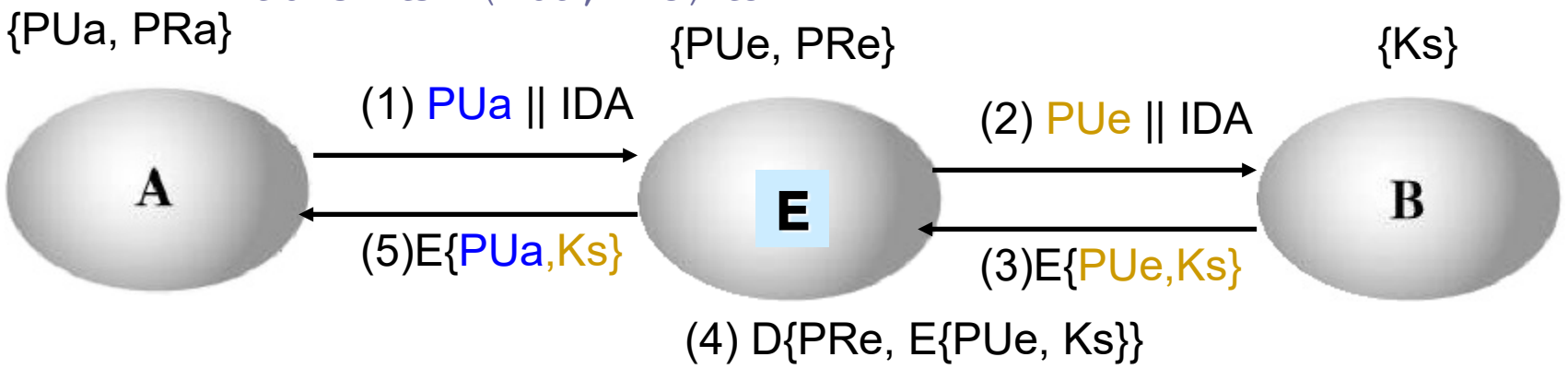
- proposed by Merkle in 1979
  - A generates a new temporary public key pair {PUa, PRa}
  - A sends B the public key and their identity
  - B generates a session key Ks sends it to A encrypted using the supplied public key
  - A decrypts the session key and both (A,B) can use this key
- This protocol is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message (man-in-the-middle attack)



< Simple use public-key encryption to establish a session key >

Man-in-the-middle Attack

- A generates a public/private key pair {PUa, PRa} and transmits a message intended for B consisting of PUa and an identifier of A, ID_A.
- E intercepts the message, creates its own public/private key pair {PUe, PRe} and transmits PUe||ID_A to B.
- B generates a secret key, Ks, and transmits E(PUe, Ks).
- E intercepts the message, and learns Ks by computing D(PRe, E(PUe, Ks)).
- E transmits E(PUa, Ks) to A



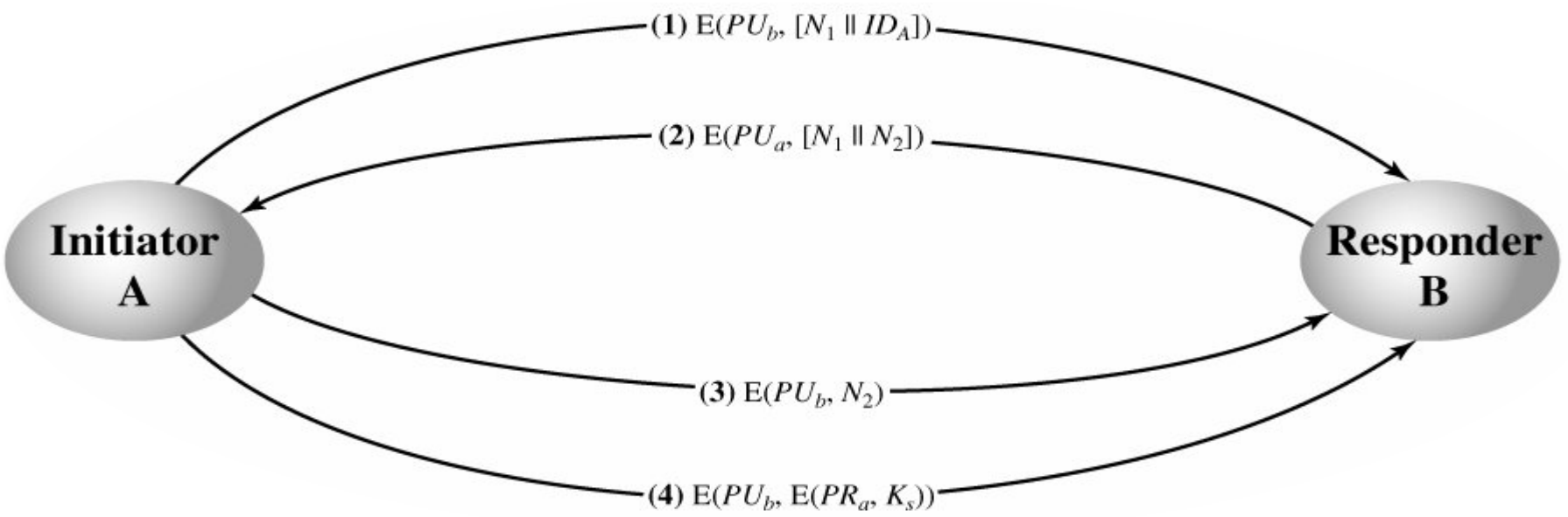
< Man-in-the-middle attack>

* A & B know the Ks and are unaware that Ks has also revealed to E

Secret Key Distribution with Confidentiality and Authentication

(1) A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N1), which is used to identify this transaction uniquely.

(2) B sends a message to A encrypted with PUa and containing A's nonce (N1) & a new nonce generated by B (N2) because only B could have decrypted message(1). The presence of N1 in message (2) assures A that the correspondent is B.



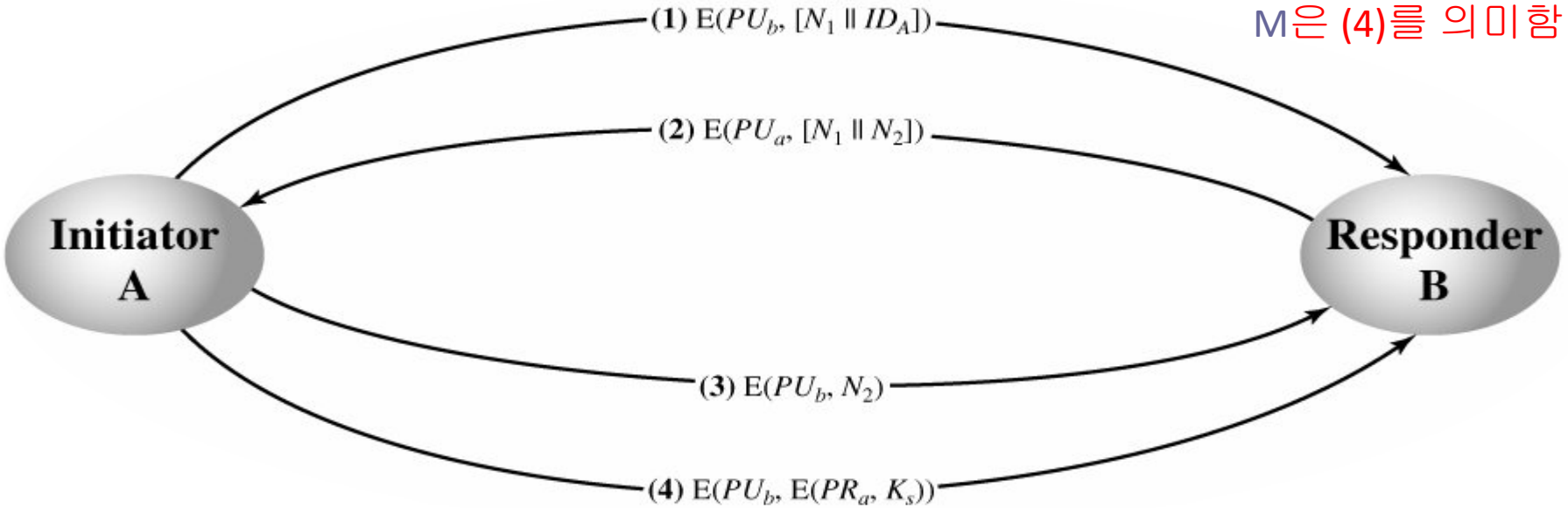
Secret Key Distribution with Confidentiality and Authentication

(3) A returns N2 encrypted using B's public key, to assure B that its correspondent is A

(4) A selects a secret key Ks and sends M = E(PUb, E(PRa, Ks)) to B. Encryption of this message with B's public key ensures that only B can read it; encryption with A's private key ensures that only A could have sent it.

* B computes D(PUa, D(PRb, **M**)) to recover the secret key.

M은 (4)를 의미함



Hybrid Key Distribution

- Another way to use public-key encryption to distribute secret keys is a hybrid approach : The use of KDC(Key Distribution Center) (used on IBM mainframe computer)
- It shares secret master key with each user
  - public-key used to distribute master keys
  - especially useful with widely distributed users
- distributes session key using master key
- Feature of this scheme
  - Performance : The session key exchange by public-key encryption could degrade overall system. With this scheme, the public-key encryption is used only to update the master key b/w a user and the KDC
  - Backward compatibility : this hybrid scheme is easily overlaid on existing KDC scheme.

Agenda

- Key Management
- Diffie-Hellman Key Exchange

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that Williamson (UK) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Key Exchange

- a public-key distribution scheme
  - cannot be used to exchange an arbitrary message
  - rather it can establish a common key
  - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Discrete Logarithm

- We can define the discrete logarithm in the following way
- First, we define a primitive root of a prime number p as one whose powers modulo p generate all the integers from 1 to $p-1$
  - That is, if a is a primitive root of the prime number p, then the numbers $a \bmod p,\ a^2 \bmod p, \ldots, a^{p-1} \bmod p$ are distinct and consist of the integers from 1 through $p-1$ in some permutation.
- For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that $b = a^i \pmod p$ where $0 \le i \le (p-1)$
- The exponent i is referred to as the discrete logarithm of b for the base $a, \bmod p$
- $\mathrm{dlog}_{a,p}(b)$

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$
- each user makes his key $(e.g.\ y_A)$ in public

Diffie-Hellman Key Exchange

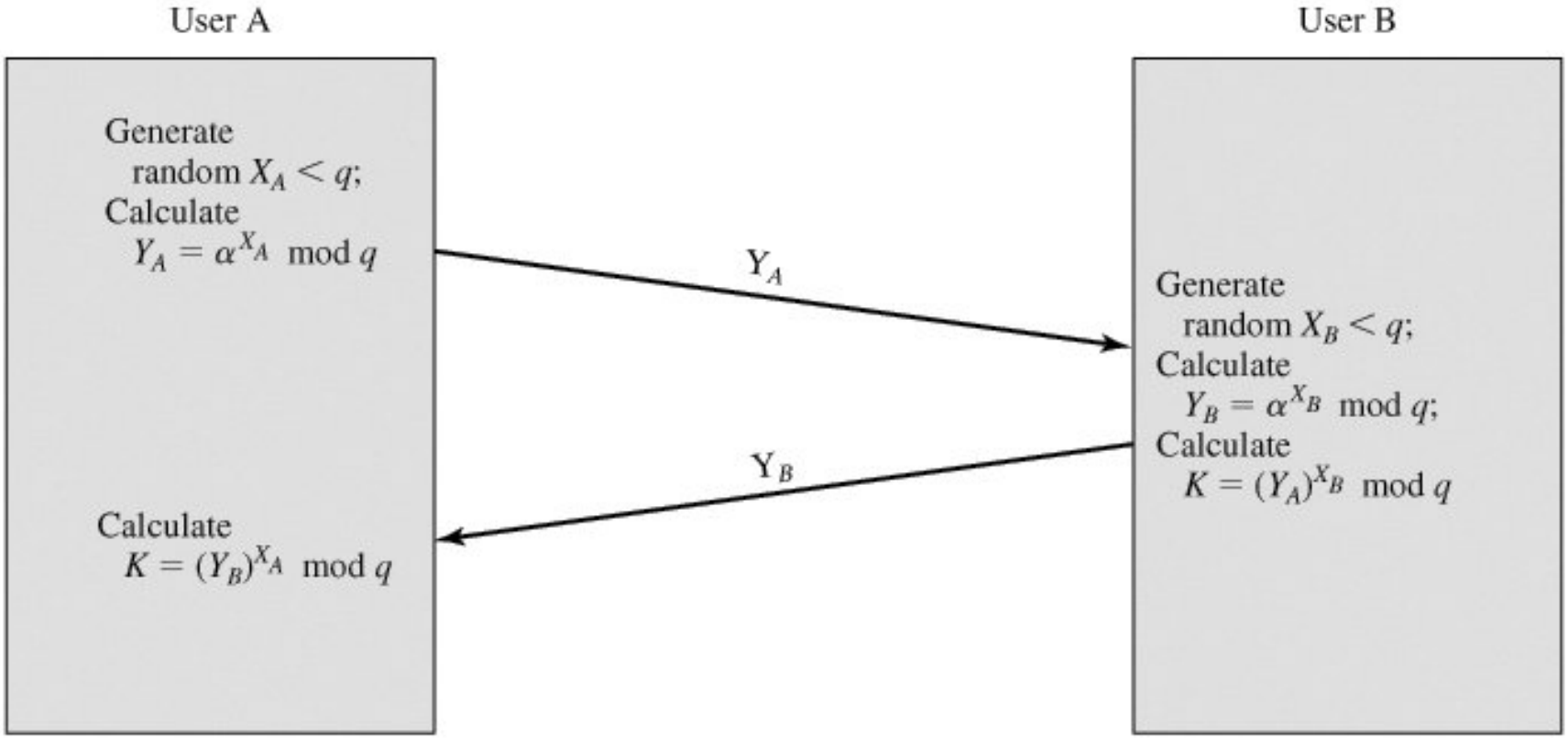
- shared session key for users A & B is K_{AB}:

$$K_{AB} = a^{x_A . x_B} \bmod q$$
$$= y_A^{x_B} \bmod q \quad \text{(which } \mathbf{B} \text{ can compute)}$$
$$= y_B^{x_A} \bmod q \quad \text{(which } \mathbf{A} \text{ can compute)}$$

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x, must solve discrete log

Diffie-Hellman Key Exchange

- Suppose that user A wishes to set up a connection with user B and use a secret key to encrypt messages on that connection



Diffie-Hellman Example

- users Alice & Bob who wish to exchange keys:
- agree on prime $q=353$ and $a=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
  - $y_A=3^{97} \mod 353 = 40$ (Alice)
  - $y_B=3^{233} \mod 353 = 248$ (Bob)
- compute shared session key as:
  - $K_{AB}= y_B^{x_A} \mod 353 = 248^{97} = 160$ (Alice)
  - $K_{AB}= y_A^{x_B} \mod 353 = 40^{233} = 160$ (Bob)

DH Key Exchange Protocols

- users could create random private/public DH keys each time they communicate
- users could create a known private/public DH key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a meet-in-the-Middle Attack
- authentication of the keys is needed

Man-in-the-Middle Attack

1. Darth prepares by creating two private / public keys
2. Alice transmits her public key to Bob
3. Darth intercepts this and transmits his first public key to Bob. Darth also calculates a shared key with Alice
4. Bob receives the public key and calculates the shared key (with Darth instead of Alice)
5. Bob transmits his public key to Alice
6. Darth intercepts this and transmits his second public key to Alice. Darth calculates a shared key with Bob
7. Alice receives the key and calculates the shared key (with Darth instead of Bob)

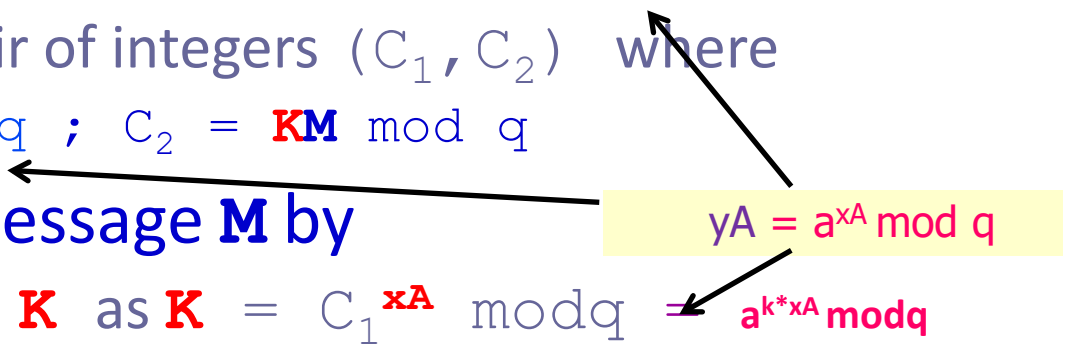
- Darth can then intercept, decrypt, re-encrypt, forward all messages between Alice & Bob

ElGamal Cryptography

- public-key cryptosystem related to DH
- so uses exponentiation in a finite (Galois)
- with security based difficulty of computing discrete logarithms, as in DH
- each user (eg. A) generates their key
  - chooses a secret key (number): $1 < x_A < q-1$
  - compute their **public key**: $y_A = a^{x_A} \bmod q$

ElGamal Message Exchange

- Bob encrypt a message to send to A computing
  - represent message M in range $0 <= M <= q-1$
    - longer messages must be sent as blocks
  - choose random integer k with $1 <= k <= q-1$
  - compute one-time key $K = y_A^k \bmod q = a^{xA*k} \bmod q$
  - encrypt **M** as a pair of integers (C_1, C_2) where
    - $C_1 = a^k \bmod q$; $C_2 = KM \bmod q$
- A then recovers message **M** by
  - recovering key **K** as $K = C_1^{xA} \bmod q = a^{k*xA} \bmod q$
  - computing **M** as $M = C_2 K^{-1} \bmod q$
- a unique k must be used each time
  - otherwise result is insecure



ElGamal Example

- use field GF(19) $q=19$ and $a=10$
- Alice computes her key:
  - A chooses $x_A=5$ & computes $y_A=10^5 \bmod 19 = 3$
- Bob send message $m=17$ as $(11,5)$ by
  - chosing random $k=6$
  - computing $K = y_A^k \bmod q = 3^6 \bmod 19 = 7$
  - computing $C_1 = a^k \bmod q = 10^6 \bmod 19 = 11$;
    $C_2 = KM \bmod q = 7.17 \bmod 19 = 5$
- Alice recovers original message by computing:
  - recover $K = C_1^{xA} \bmod q = 11^5 \bmod 19 = 7$
  - compute inverse $K^{-1} = 7^{-1} = 11$
  - recover $M = C_2\ K^{-1} \bmod q = 5.11 \bmod 19 = 17$

Next…

- We will study on Elliptic Curve Cryptography…


Q&A