Information Systems Security

Dr. Ayman Abdel-Hamid

College of Computing and Information Technology

Arab Academy for Science & Technology and
Maritime Transport

Ch14 & Section 10.1

Key Management in public-key Cryptography

Outline

- Key management in public-key cryptography
  - Distribution of public keys
  - use of public-key encryption to distribute secret keys
  - Diffie-Hellman Key Exchange

Key Management

- public-key encryption helps address key distribution problems
- Two aspects
  - distribution of public keys
  - use of public-key encryption to distribute secret keys

Distribution of Public keys

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
  - e.g., append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
  - anyone can create a key claiming to be someone else and broadcast it
  - until forgery is discovered can masquerade as claimed user

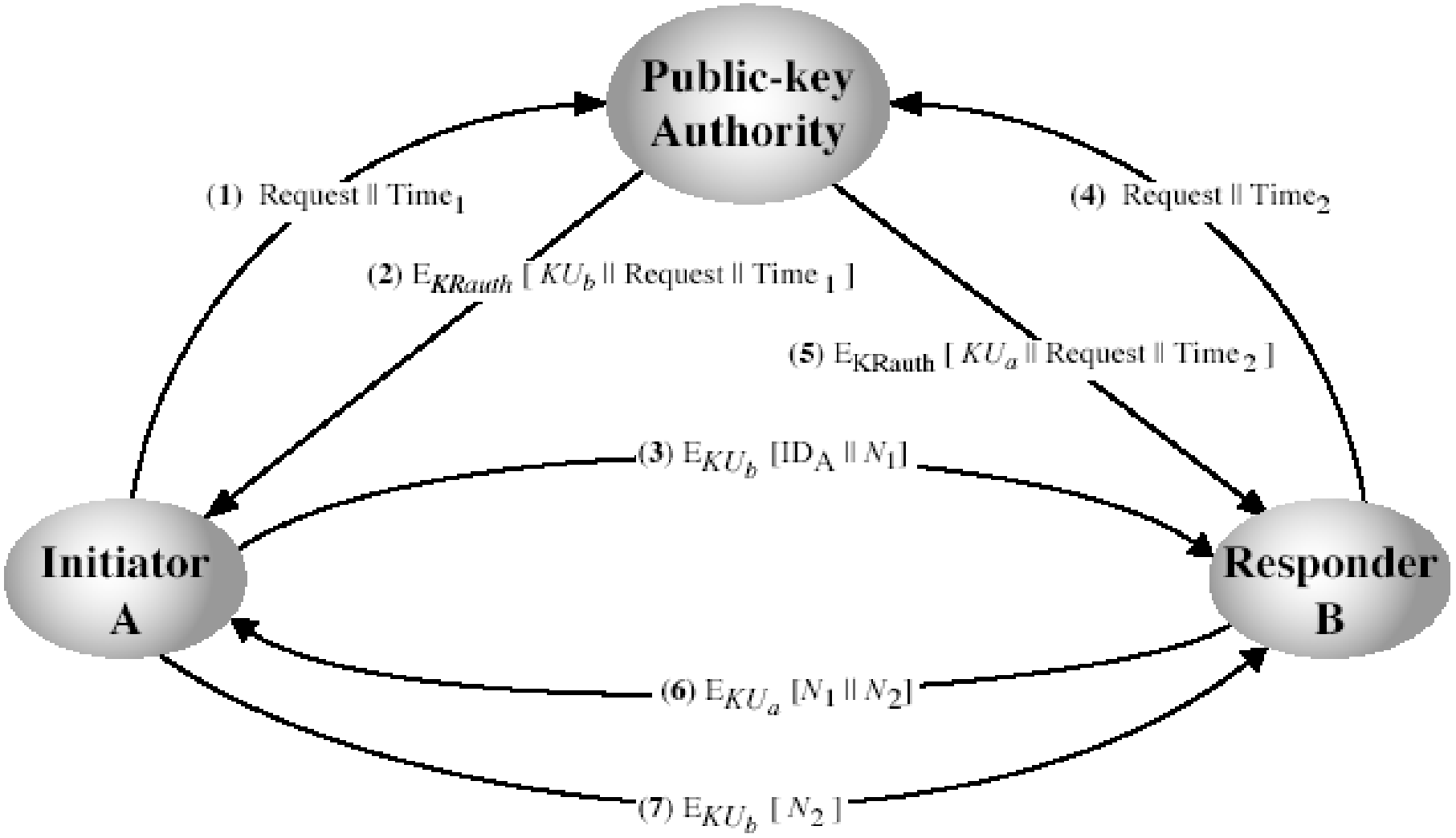
Publicly Available Directory

- can obtain greater security by registering 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
  - Known private key of directory authority
  - Tamper with records of directory authority

Public-Key Authority

- Improve security by tightening control over distribution of keys from directory
- Has properties of directory
- *Requires users to know public key for the authority (only authority knows 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 (can be a bottleneck)
  - Still target for tampering

Public-Key Authority



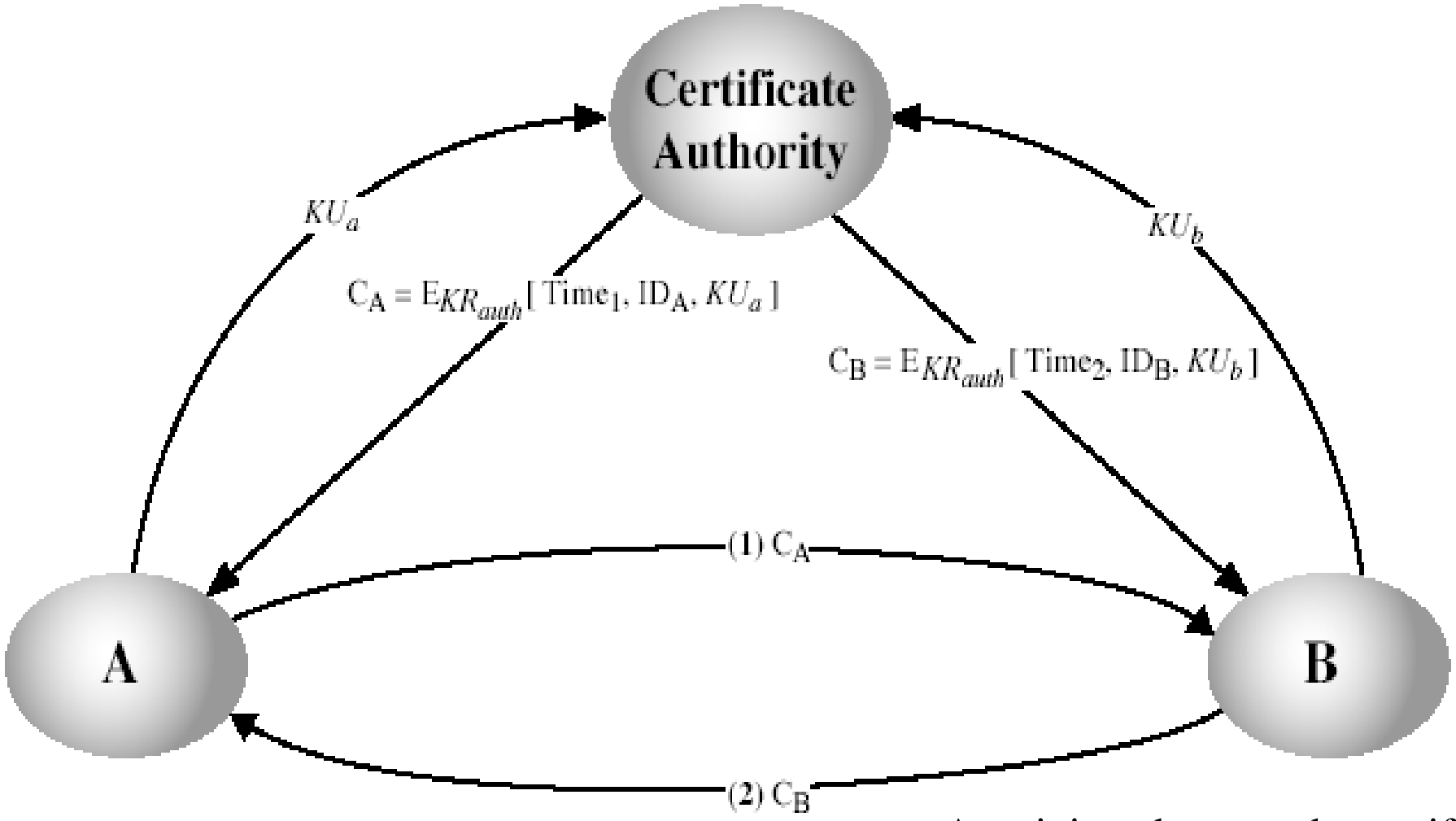
Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
  - usually with other info 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 authorities' public-key*

Public-Key Certificates

- Requirements
  - Any participant can read a certificate to determine name and public key of certificate's owner
  - Any participant can verify the certificate is originated from the CA
  - Only the certificate authority can create and update certificates
  - Any participant can verify the currency of the certificate

Public-Key Certificates



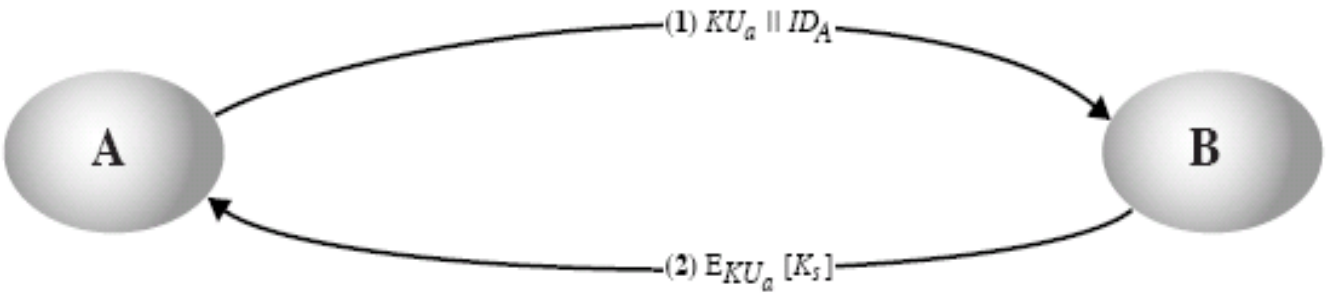
A recipient decrypts the certificate using the CA's public key

Public-key Distribution of Secret Keys

- Use previous methods to obtain public-key
- Can use for secrecy or authentication
- But public-key algorithms are slow
- Usually want to use private-key encryption to protect message contents
- Hence need a session key
- Have several alternatives for negotiating a suitable session

Simple Secret Key Distribution

- proposed by Merkle in 1979
  - A generates a new temporary public key pair
  - A sends B the public key and its identity
  - B generates a session key K (secret key) sends it to A encrypted using the supplied public key
  - A decrypts the session key and both use

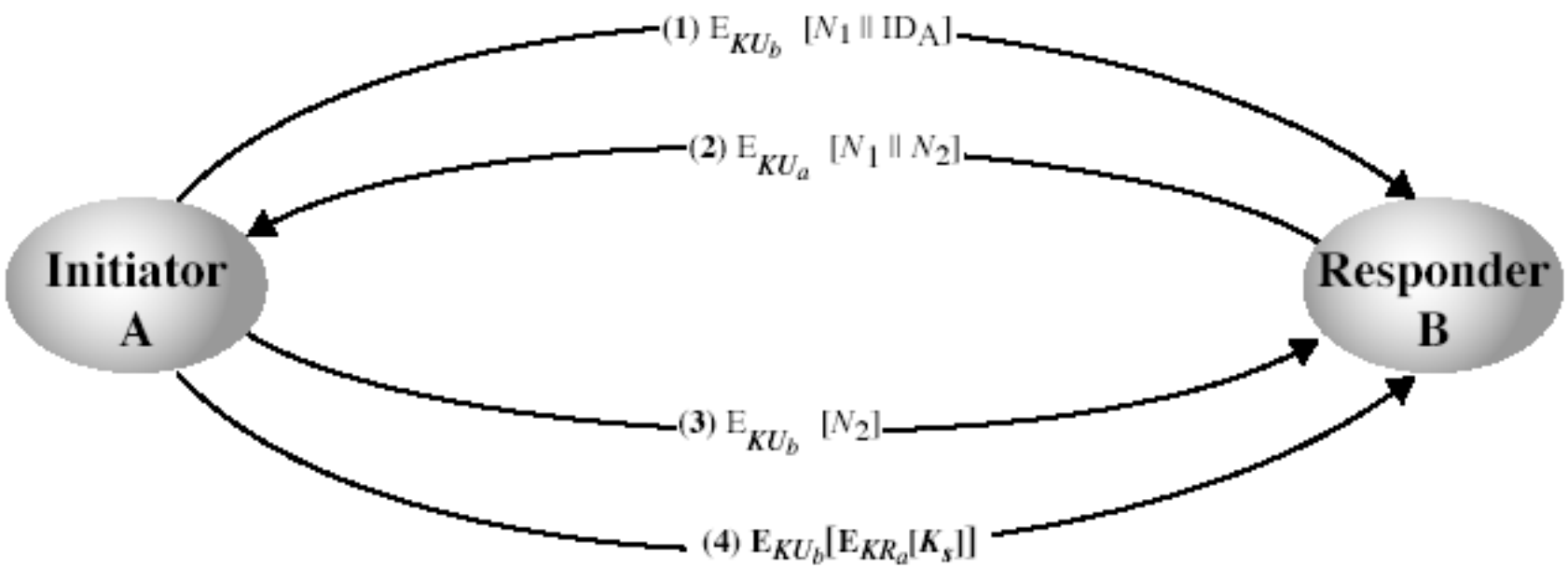


Simple Secret Key Distribution

- problem is that an opponent can intercept and impersonate both halves of protocol (active attack)
  - A generates public/private key pair and transmits a message for B including public key and identity $\{KU_a \| ID_A\}$
  - E intercepts message, creates its own public/private key pair and transmits $\{KU_e \| ID_A\}$ to B
  - B generates session key and transmits $E_{KUe}[K_s]$
  - E intercepts the message and learns K_s
  - E transmits $E_{KUa}[Ks]$ to A
- E can decrypt all messages thereafter!

Distribution with Confidentiality and Authentication

- NEED in 1978
  - Provides protection against active and passive attacks
  - Assume A and B have securely exchanged public keys



A hybrid Scheme for Key Distribution

- Used on IBM Mainframes
- Retains the idea of a KDC
- Distributes secret session keys encrypted with the master keys
- *A public key scheme is used to distribute the master keys*
- Rationale
  - Session keys can change frequently
  - Backward compatibility (easily overlay on an existing KDC scheme)

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 James Ellis (UK CESG) 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 modulo a prime
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – very difficult problem

Diffie-Hellman Key Exchange

- Define a *primitive root* of a prime number *p* as one whose powers generate all integers from 1 to *p*-1

$$a \bmod p,\ a^2 \bmod p,\ ..,\ a^{p-1} \bmod p$$

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 p, can find a unique exponent *i* such that

$$b = a^i \bmod p \text{ where } 0 <= i <= (p-1)$$

- Exponent *i* referred to as the discrete logarithm, or index of *b* to the base *a*

Diffie-Hellman Key Setup

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

Diffie-Hellman Key Exchange

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

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

- 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 (infeasible for large primes)

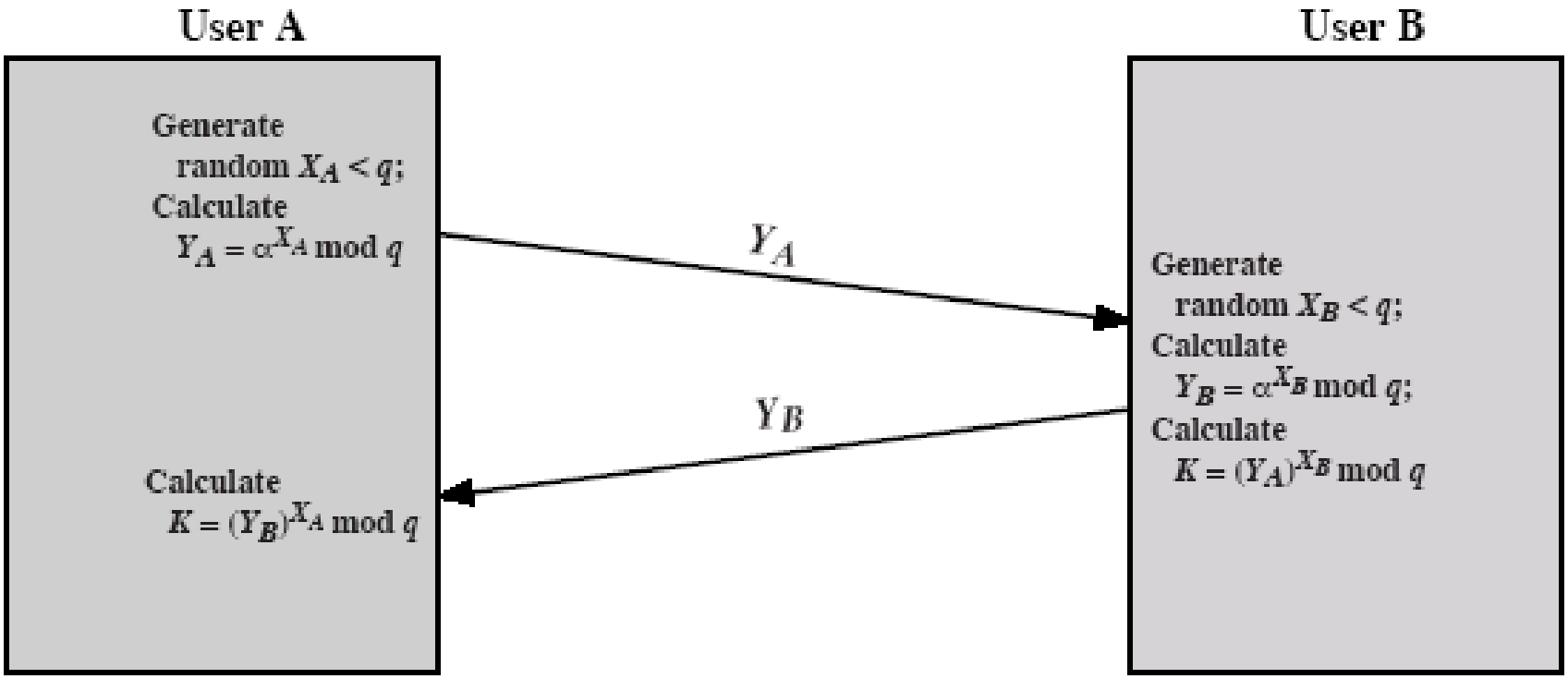
Diffie-Hellman Example

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

$K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)

$K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Using Diffie-Hellman in a simple Protocol



q and α could be known ahead of time or A picks the values and include in first message