UNIVERSITY OF RAJSHAHI

Rajshahi, BANGLADESH.

Course Code:

ICE-4221

Course Title :

Cryptography and Network security

Public Key Management

Key Management and Elliptic Curve Cryptography (ECC):

Key management,

Diffie-Hellman key exchange,

Elliptic curve arithmetic,

ECC-key exchange using ECC,

Elliptic curve encryption/decryption.

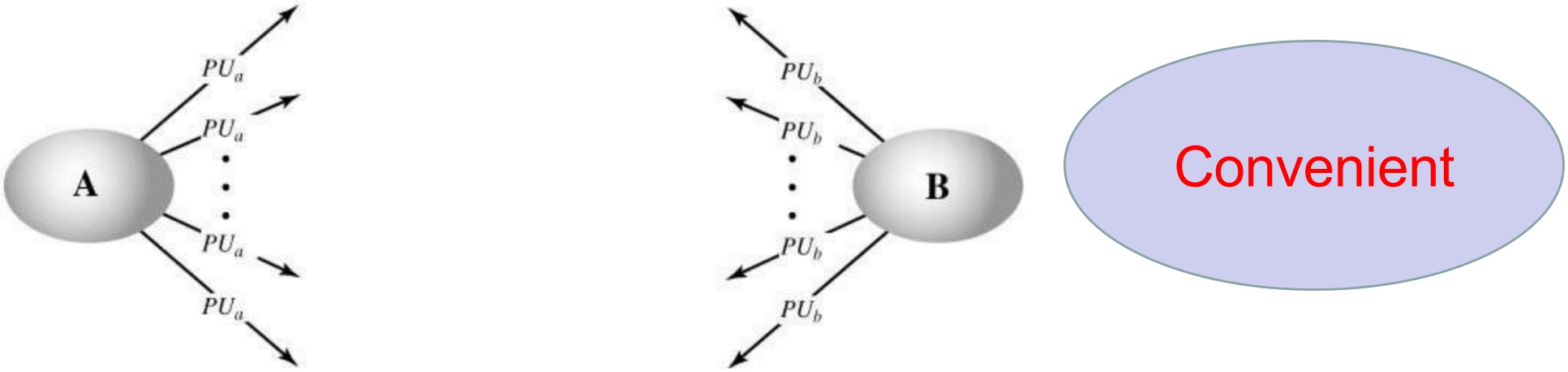
Key Management

- One of the major roles of public-key encryption has been to address the problem of key distribution.
- There are actually two distinct aspects to the use of public-key cryptography in this regard:
  - **Distribution of public keys**
  - **Use of public-key encryption to distribute secret keys**

Distribution of Public Keys

- Several techniques have been proposed for the distribution of public keys.
- Virtually all these proposals can be grouped into the following general schemes:
  - **Public announcement**
  - **Publicly available directory**
  - **Public-key authority**
  - **Public-key certificates**

Public Announcement of Public Key

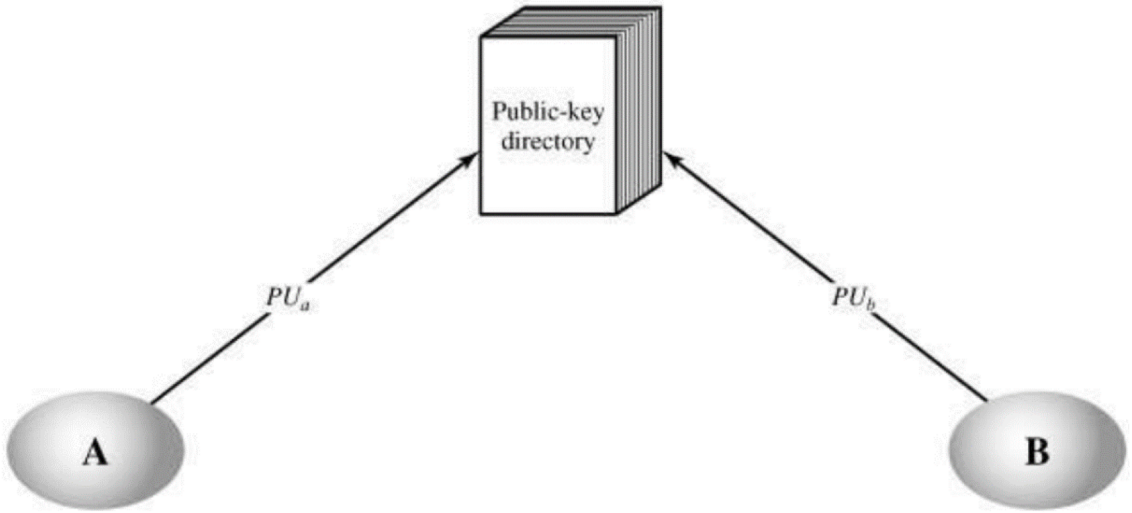


- On the face of it, the point of public-key encryption is that the public key is public.
- Thus, if there is some broadly accepted public-key algorithm, such as RSA, any participant can send his or her public key to any other participant or broadcast the key to the community at large

Public Announcement of Public Key

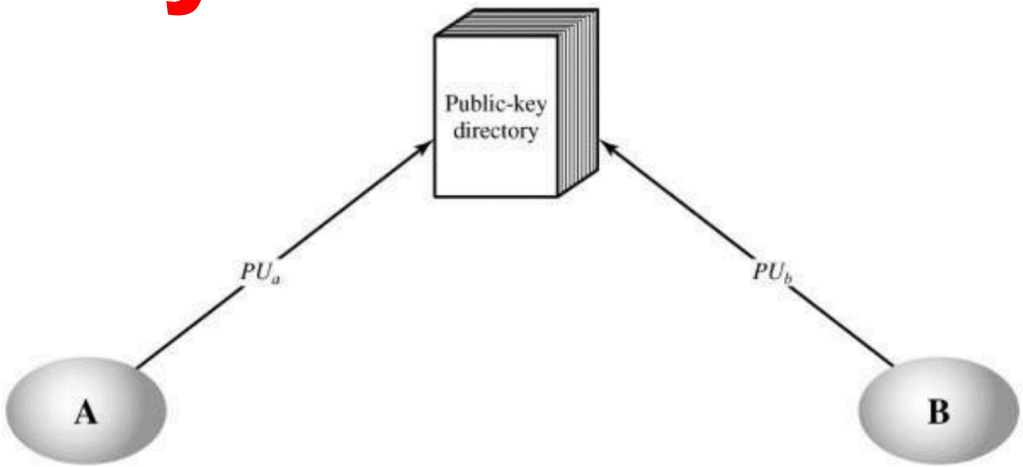
- Major weakness is forgery
  - Anyone can create a key claiming to be someone else and broadcast it
  - Until forgery is discovered can masquerade as claimed user for authentication

Publicly Available Directory



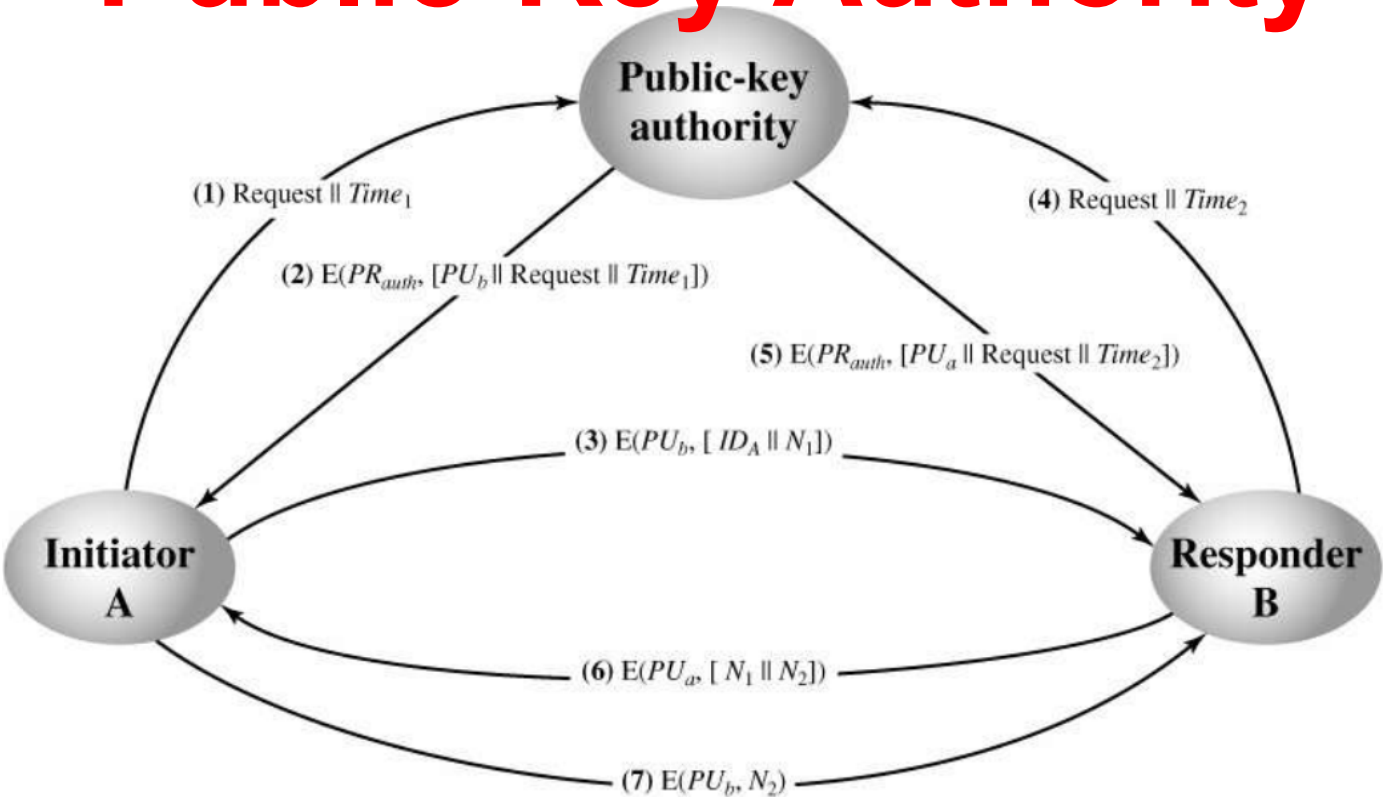
- A greater degree of security can be achieved by maintaining a publicly available dynamic directory of public keys.
- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization.

Publicly Available Directory



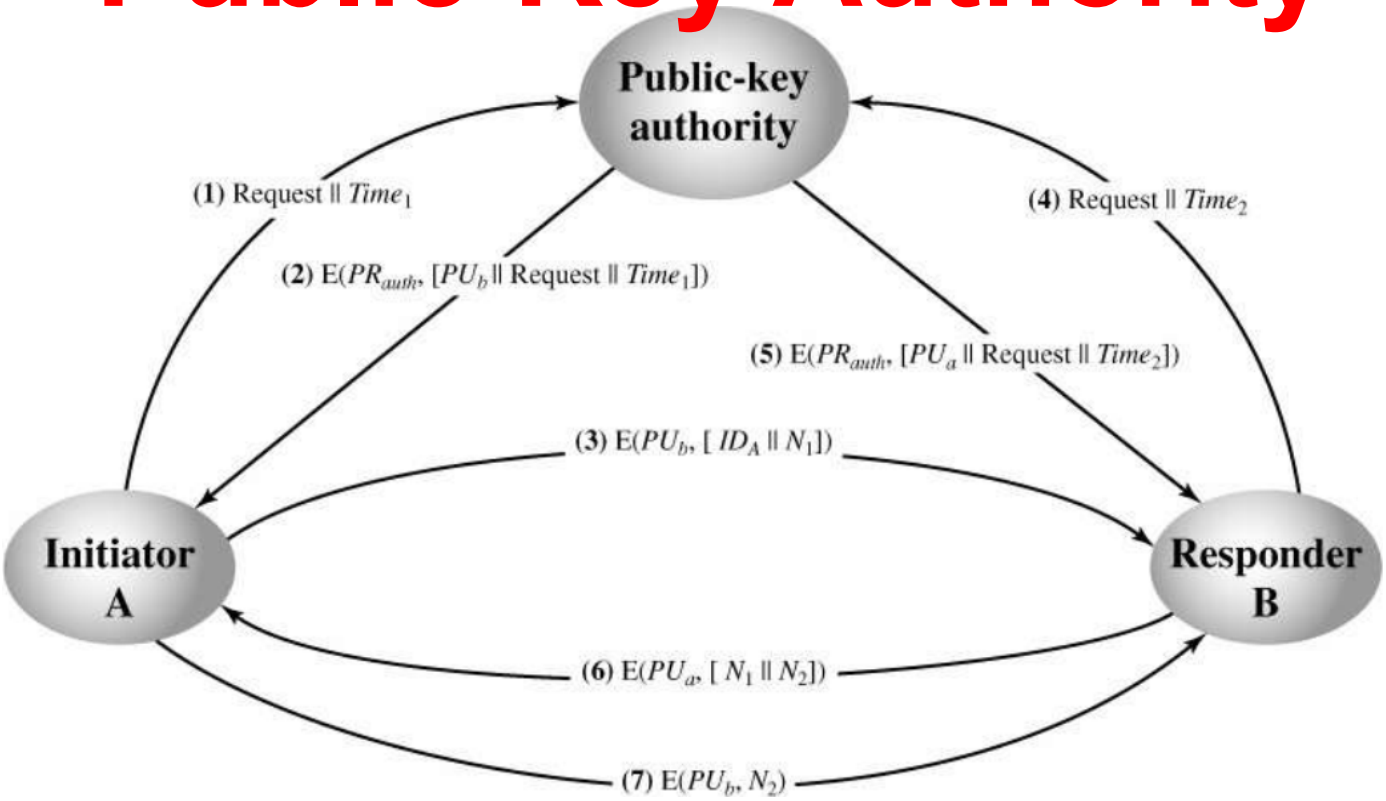
- Directory must be trusted with properties:
  - Maintain a directory with a {name, public-key} entries for each participants
  - Each participant registers securely with directory (How?).
  - Participants can replace (why?) key at any time.
  - Directory is periodically published.
  - Directory can be accessed electronically (Who? How?)
- This scheme is clearly more secure than individual public announcements
- Still vulnerable to tampering or forgery (How?)

Public-Key Authority



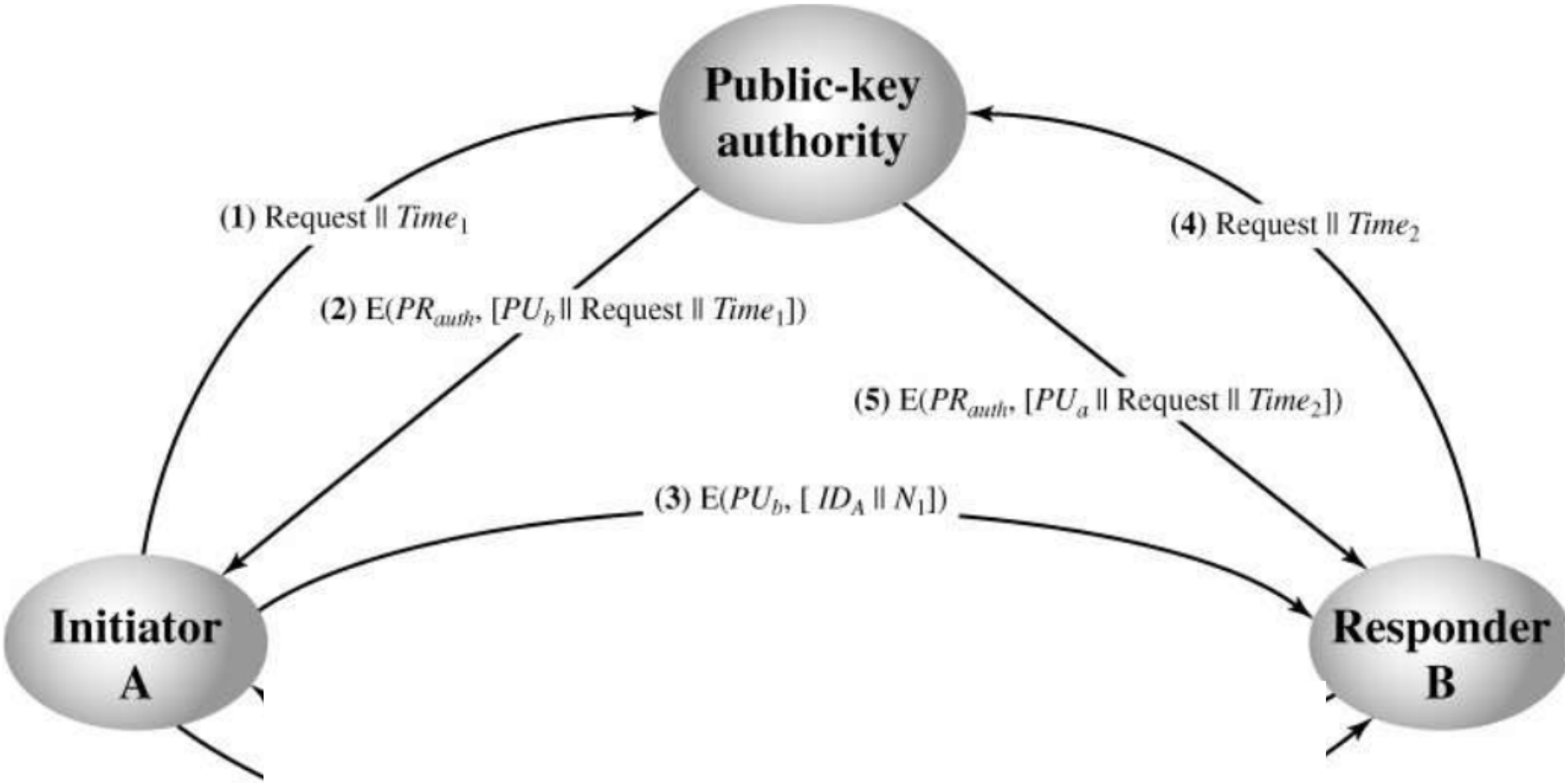
- ✓Stronger security for public-key distribution can be achieved by providing tighter control over the distribution of public keys from the directory.
- ✓As before, this scenario assumes that a central authority maintain a dynamic directory of all participants.

Public-Key Authority

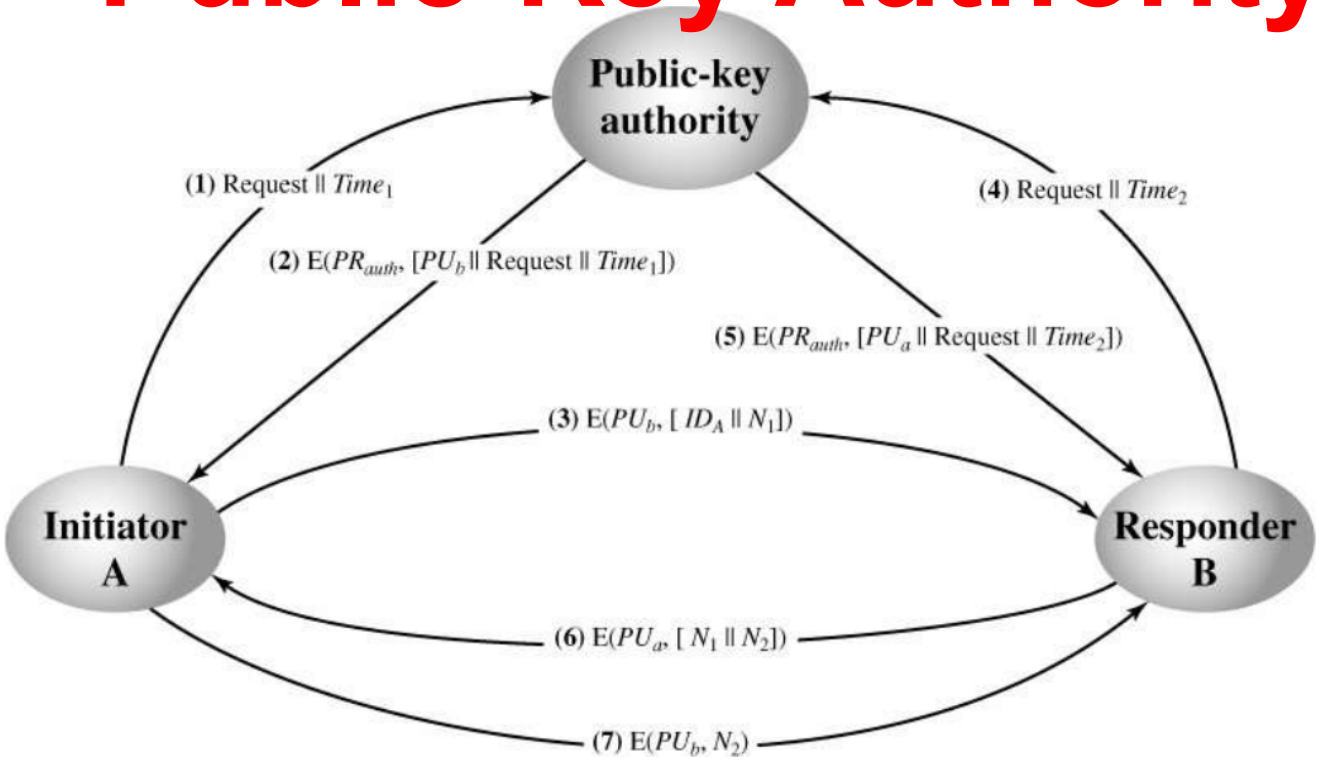


- ✓In addition, each participant reliably apply knows a public key for the authority , with only the authority knowing the corresponding private key.

Public-Key Authority



Public-Key Authority



- Thus, a total of **SEVEN** messages are required.
- However, the initial **FOUR** messages need be used only **INFREQUENTLY** because both A and B can save the other's public key for future use, a technique known as caching.
- Periodically, a user should request fresh copies of the public keys of its correspondents to ensure currency.

Public-Key Authority

- Improve security by tightening control over distribution of keys from directory
- Requires users to know public key for the directory
- Then 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

- The explained scenario is attractive, yet it has some drawbacks.
- The public-key authority could be somewhat of a bottleneck in the system, for a user must appeal to the authority for a public key for every other user that it wishes to contact.
- As before, the directory of names and public keys maintained by the authority is vulnerable to tampering.

Public-Key Certificates

- Use certificates to exchange keys without contacting a public-key authority.
  - It is reliable in a same way that keys were obtained directly from a public-key authority.
- A certificate consists of :
  - A public key
  - An identifier of the key owner.
- With the whole block signed by a **TRUSTED THIRD PARTY(?).**

Public-Key Certificates

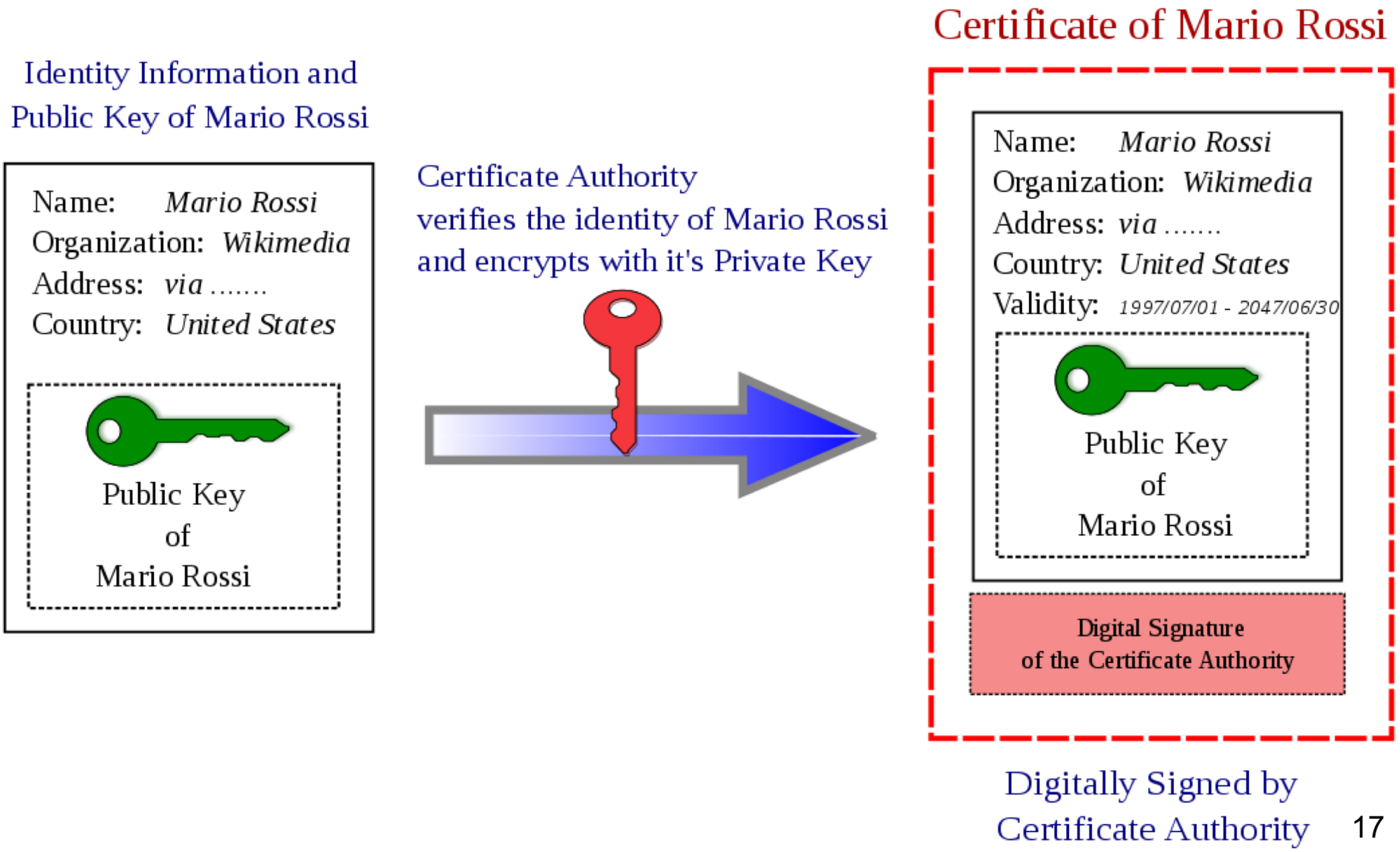
The trusted third party is a Certificate Authority, such as

- Government agency
- Financial Institution

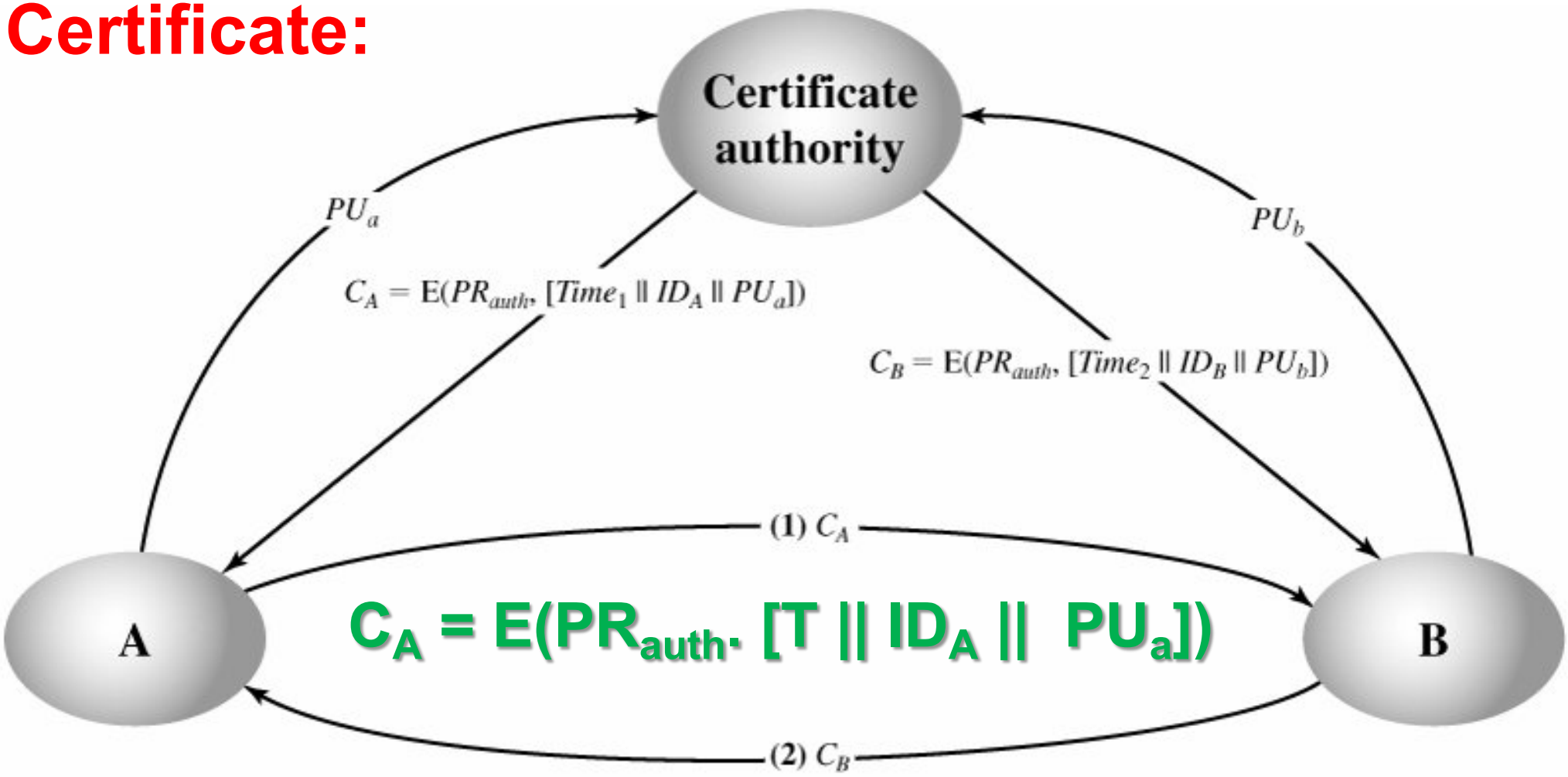
Trusted by

The User Community

Public-Key Certificates



Steps to Generate and Exchange Public-Key Certificate:

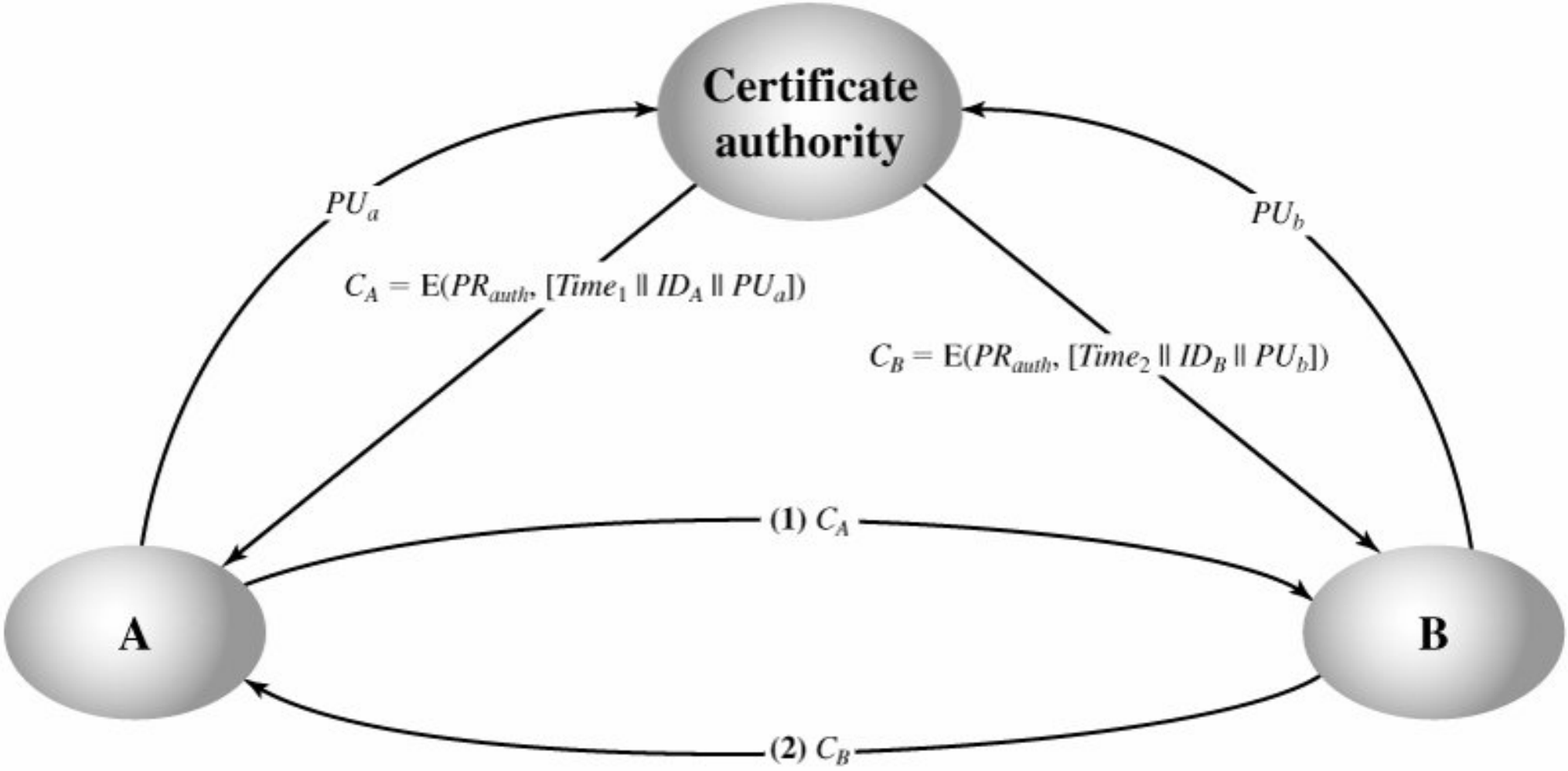


$$C_A = E(PR_{auth}.\ [T \| ID_A \| PU_a])$$

$$D\ (PU_{auth}, C_A) = D\ (PU_{auth}.\ E(PR_{auth}.\ [T \| ID_A \| PU_a])$$

$$= (T \| ID_A \| PUa)$$

Vulnerabilities of Public-Key Certificate



- If A's private key is learned by an adversary. …

Public-Key Certificates

Requirements

- Any participant can read a certificate to determine the name and public key of the certificate's owner.
- Any participant can verify that the certificate originated from the certificate authority and is not counterfeit.
- Only the certificate authority can create and update certificates.
- Any participant can verify the currency of the certificate.

Why Timestamp is required in Certificate?

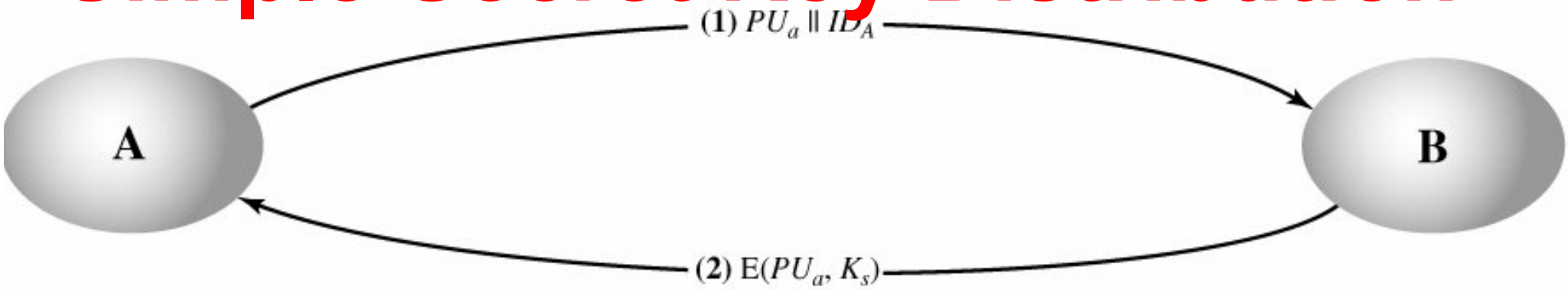
- The timestamp T validates the currency of the certificate.
- The timestamp counters the following sensitive scenario:
    - A's private key is learned by an adversary.
    - A generate a new private/public key pair and applies to the certificate authority for anew certificate.
    - Meanwhile, the adversary replays to the old certificate to B.
    - If B then encrypts messages using the compromised old public key of A
    - The adversary can read those message.

Distribution of Secret Keys using PKC

Why?

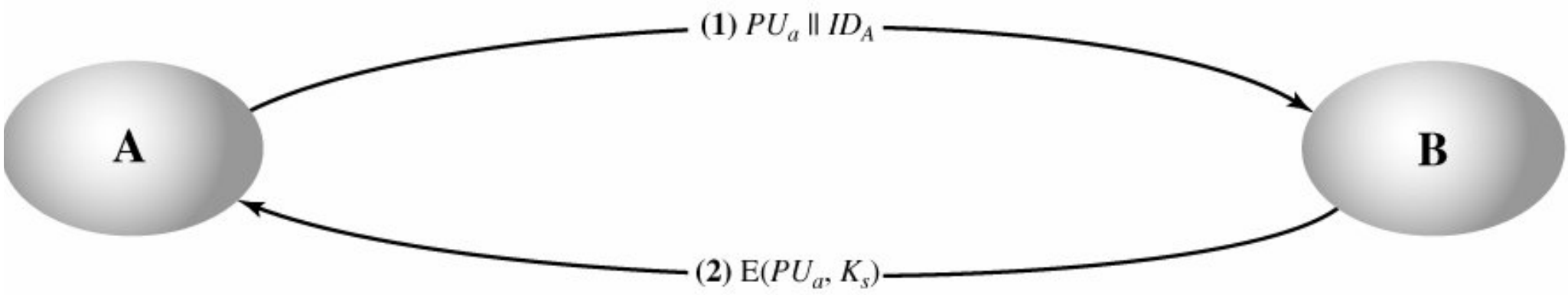
- Public-key algorithms are relatively slow.
- So, usually want to use private-key encryption to protect message contents
- Hence, need a session key (?)
- Have several alternatives for negotiating a suitable session using public-key

Simple Secret Key Distribution



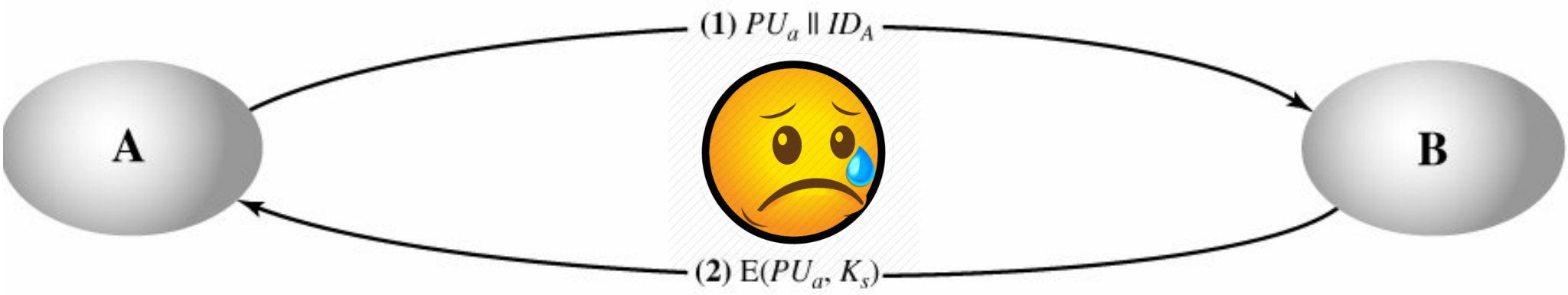
- Proposed by Merkle in 1979
  - A generates a new temporary public key pair (Pu_a, Pr_a)
  - A sends the public key Pu_a and his/her identity, Id_a to B.
  - B generates a session key Ks, sends it to A, encrypted using the supplied public key.
  - A computes D(Pra, E(Pua, Ks)) to recover the secret key
  - Both now can use conventional encryption and the session keys.
  - A discards PU_a and PR_a and B discards PU_a.

Simple Secret Key Distribution



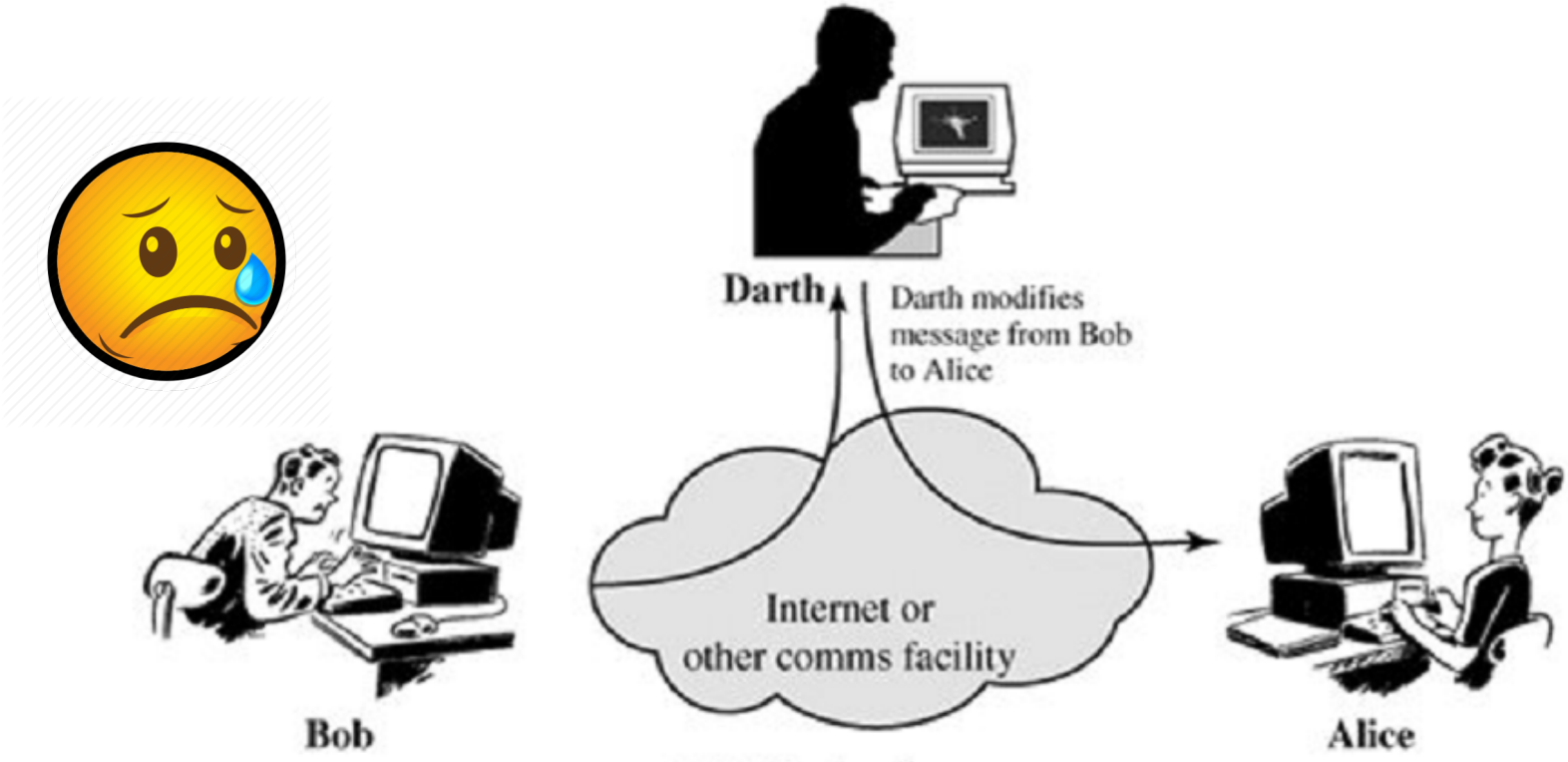
- It is an attractive protocols:
  - No keys exist before the start of the communication and none exist after the completion of communication.
  - Thus, the risk of compromise of the keys is minimal.
  - At the same time, the communication is secure from eavesdropping.

Simple Secret Key Distribution



- It is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message.

Simple Secret Key Distribution



- Such an attack is known as a **Man-in-the-Middle Attack**.

Man-in-the-middle Attack on Simple Secret Key Distribution

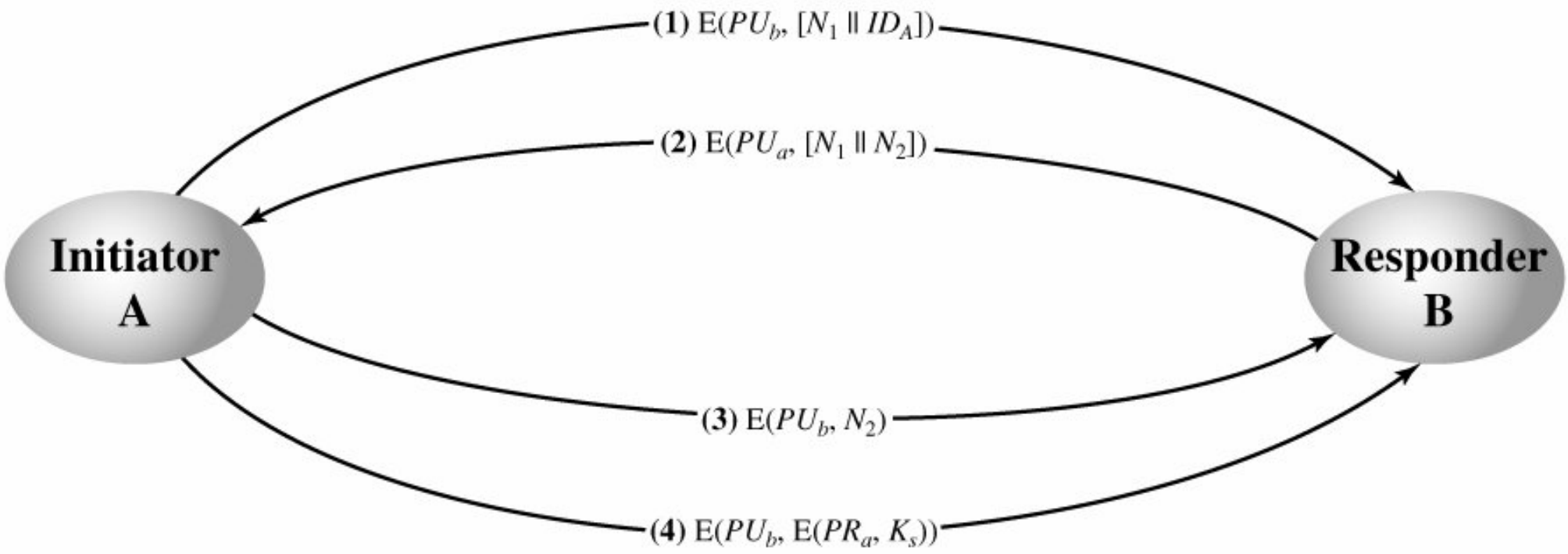
1. **A** generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message intended for **B** consisting of PU_a and an identifier of **A**, ID_A.
2. **E** intercepts the message, creates its own public/private key pair $\{PU_e, PR_e\}$ and transmits $PU_e||ID_A$ to **B**.
3. **B** generates a secret key, K_s, and transmits $E(PU_e, K_s)$.
4. **E** intercepts the message, and learns K_s by computing $D(PR_e, E(PU_e, K_s))$.
5. **E** transmits $E(PU_a, K_s)$ to A.

The result is that both A and B know K_s and are unaware that K_s has also been revealed to E.

Impact of Man-in-the-middle Attack

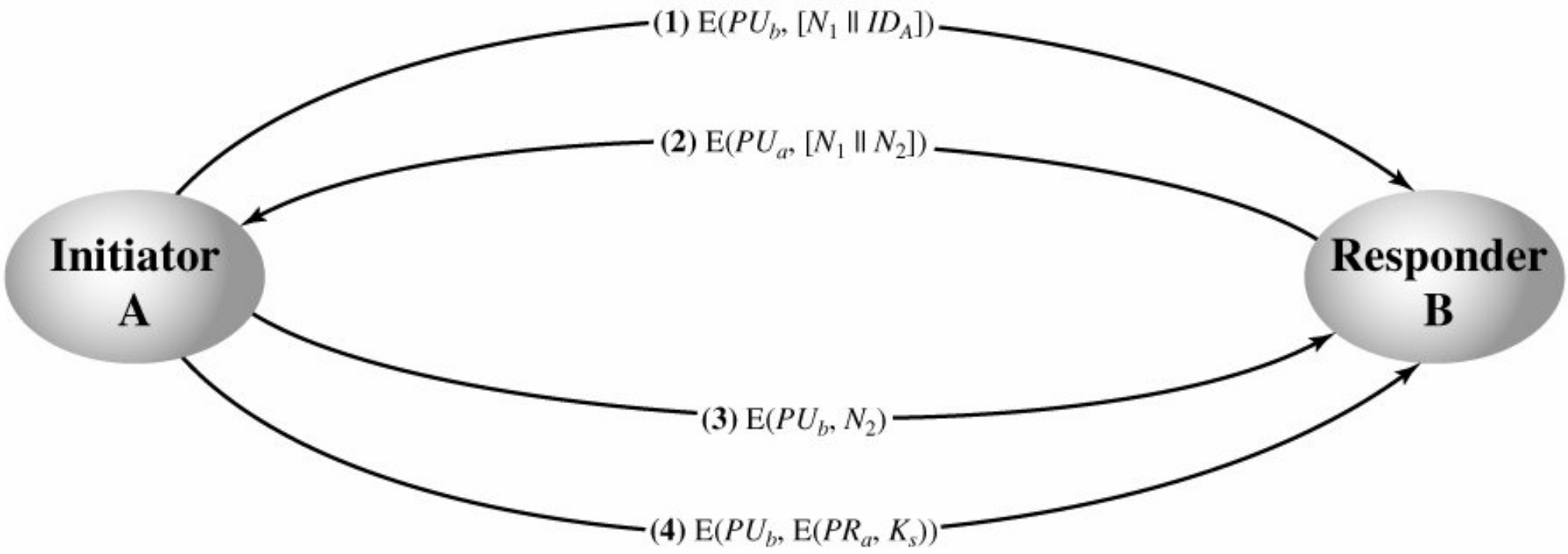
1. Both A and B know K_s and are unaware that K_s has also been revealed to E.
2. E no longer actively interferes with the communication channel but simply eavesdrops.
3. Knowing K_s E can decrypt all messages, and both A and B are unaware of the problem.
4. Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication



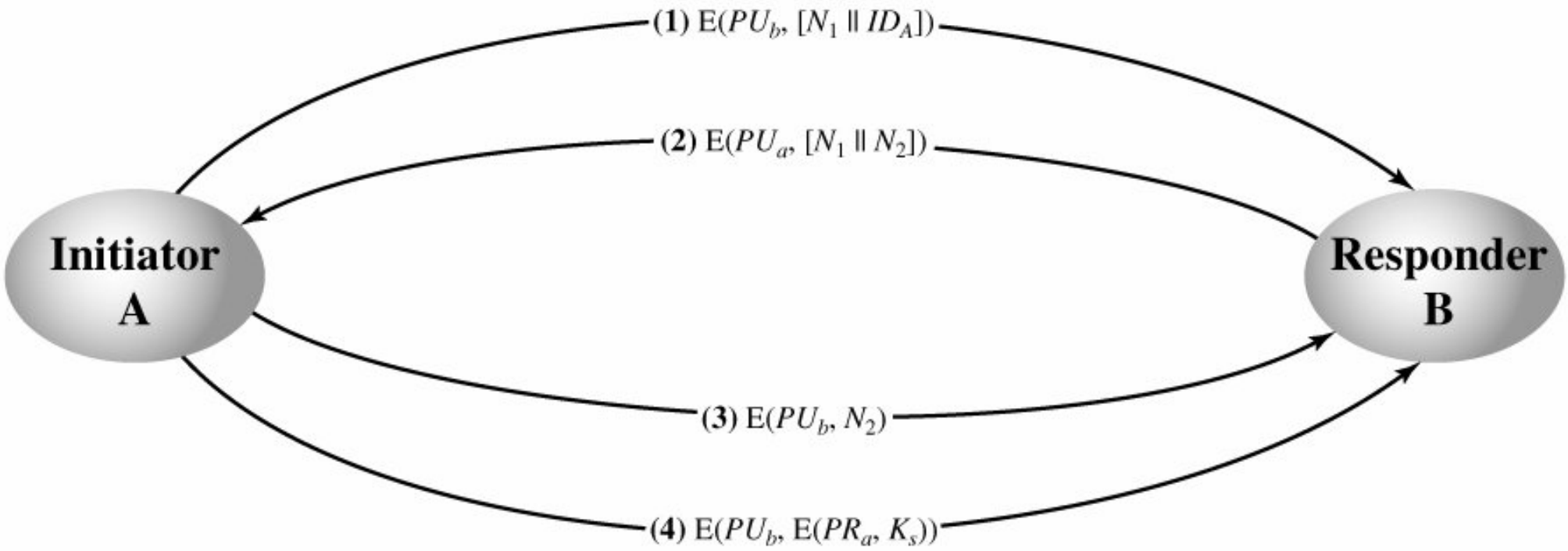
- Nonce is used to identify a transaction uniquely.

Secret Key Distribution with Confidentiality and Authentication



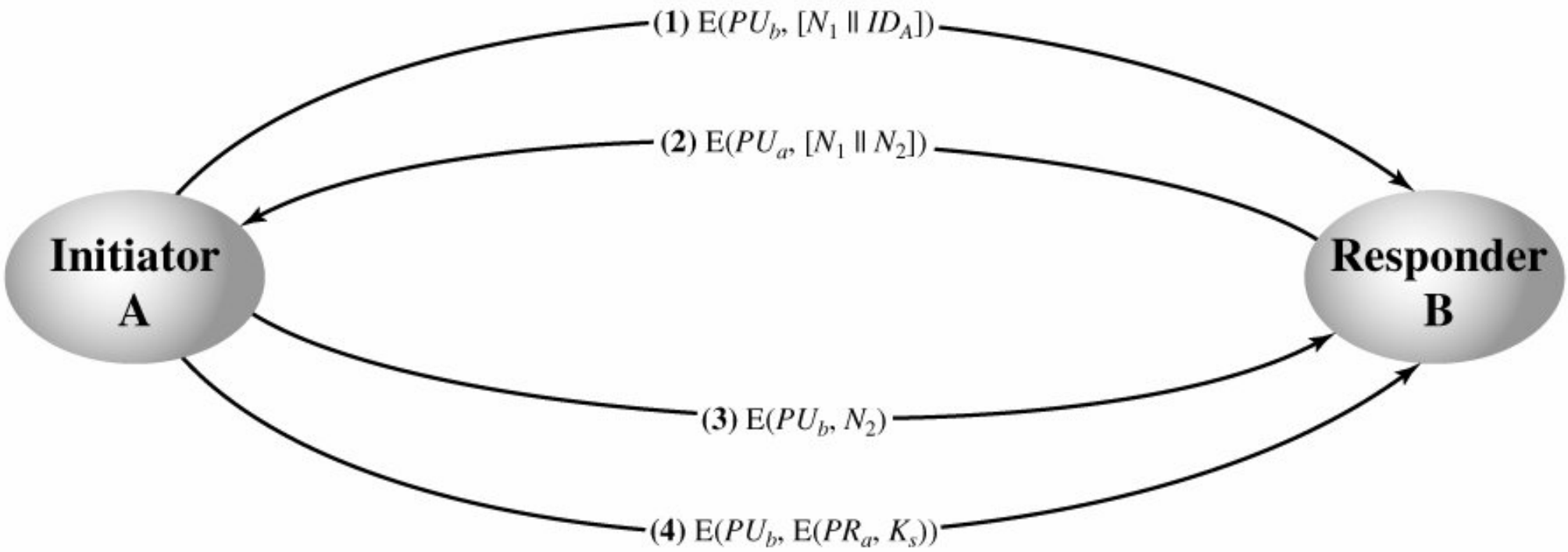
Step 1. A uses B's public key to encrypt a message to B containing an identifier of A (ID_A) and a nonce (N_1), which is used to identify this transaction uniquely.

Secret Key Distribution with Confidentiality and Authentication



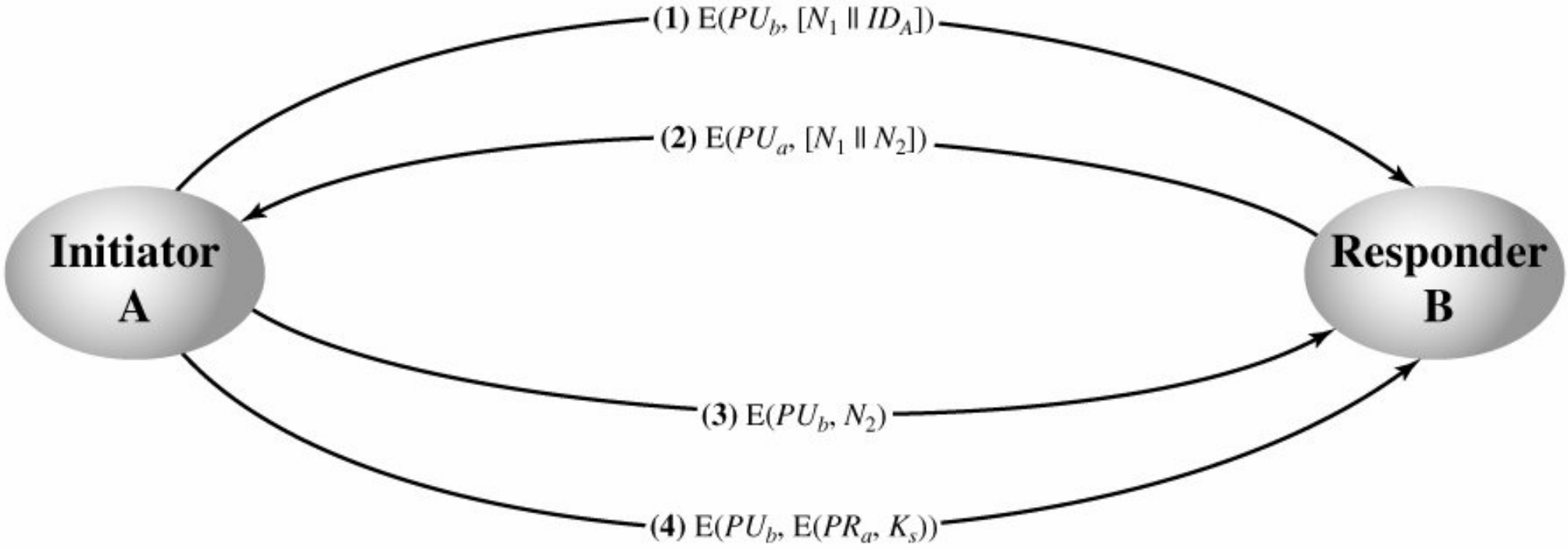
Step 2. B sends a message to A encrypted with PU_a and containing A's nonce (N_1) as well as a new nonce generated by B (N_2) Because only B could have decrypted message (1), the presence of N_1 in message (2) assures A that the correspondent is B.

Secret Key Distribution with Confidentiality and Authentication



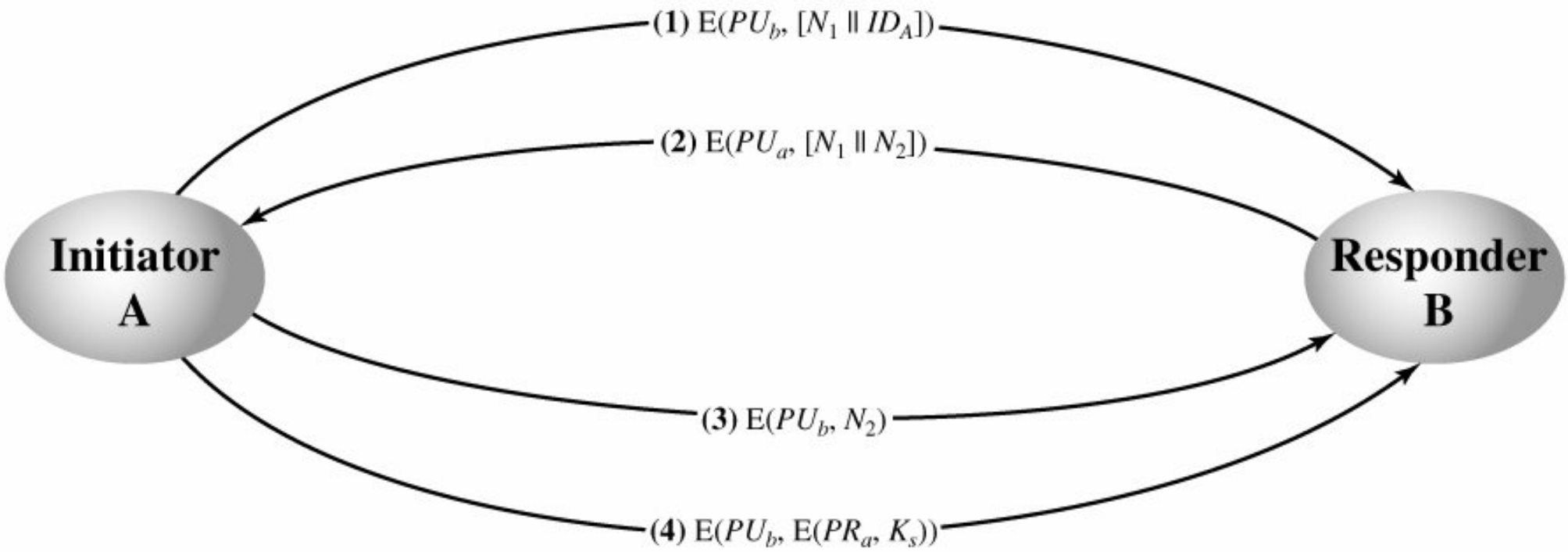
Step 3. A returns N_2 encrypted using B's public key, to assure B that its correspondent is A.

Secret Key Distribution with Confidentiality and Authentication



Step 4. A selects a secret key K_s and sends $M = E(PU_b, E(PR_a, K_s))$ 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.

Secret Key Distribution with Confidentiality and Authentication



Step 5. B computes $D(PU_a, D(PR_b, M))$ to recover the secret key

Diffie-Hellman Key Exchange

- First public-key type scheme proposed
  - For key distribution only
- A practical method for public exchange of a secret key
- Used in a number of commercial products
- 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

Diffie-Hellman Key Exchange Algorithm

Global Public Elements

q	prime number
α	$\alpha < q$ *and* α a primitive root of q

User A Key Generation

Select private X_A	$X_A < q$
Calculate public Y_A	$Y_A = \alpha^{X_A} \bmod q$

User B Key Generation

Select private X_B	$X_B < q$
Calculate public Y_B	$Y_B = \alpha^{X_B} \bmod q$

Calculation of Secret Key by User A

$K = (Y_B)^{X_A} \bmod q$

Calculation of Secret Key by User B

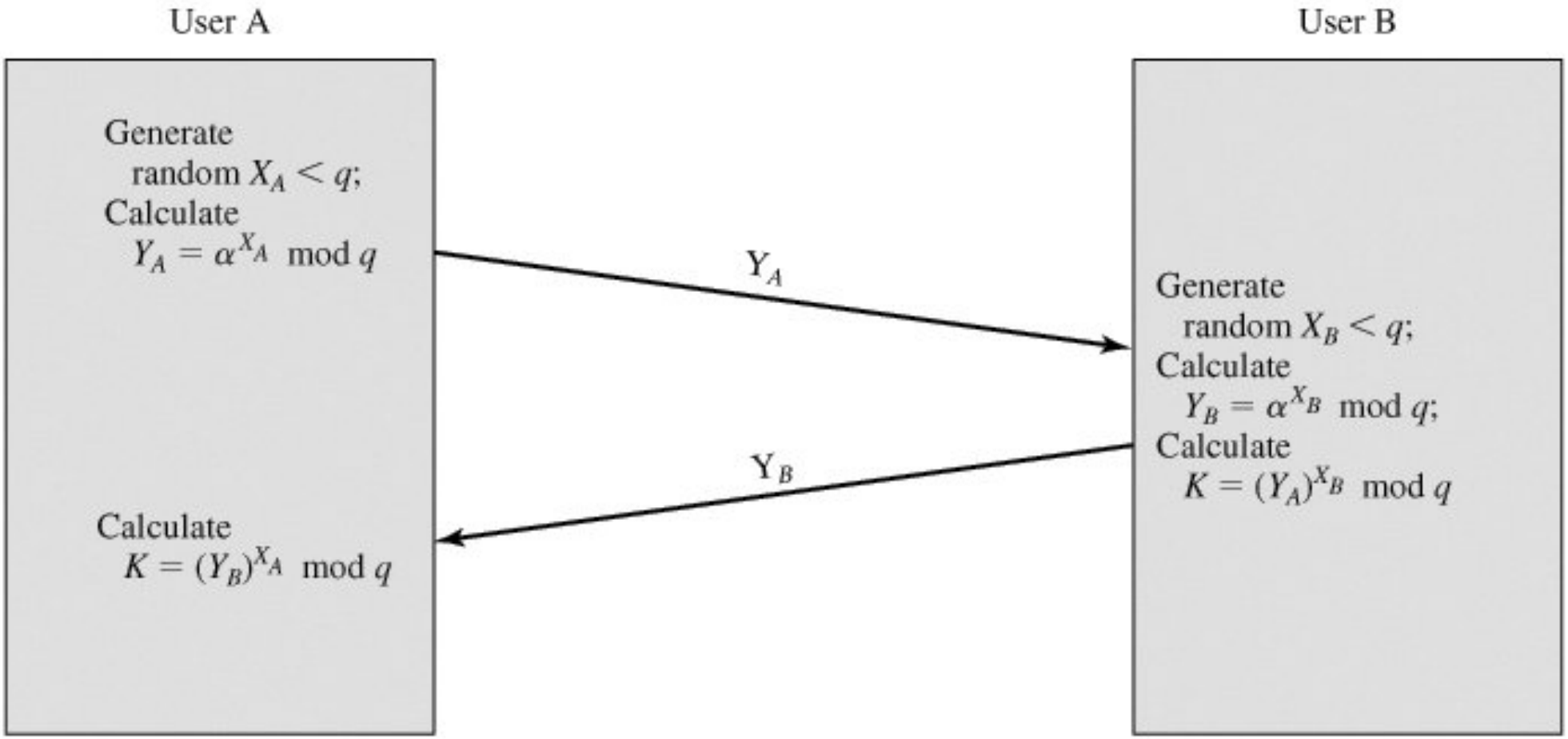
$K = (Y_A)^{X_B} \bmod q$

Diffie-Hellman Key Exchange Algorithm

Example Practice of D-H Key Exchange Algorithm

Key Exchange Protocol

Based on the use of the Diffie-Hellman Calculation



- This technique does not protect against replay attack.
- This technique is insecure against a man-in-the-middle attack.

Man-in-the-Middle Attack on DIFF based Key Exchange Protocol

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