Principles of Public key Cryptography

- Asymmetric encryption is a form of cryptosystem in which encryption and decryption are performed using the different keys- one public key and one private key
- Also known as public-key encryption
- It uses mathematical functions rather than substitution and permutation
- More secure from cryptanalysis than the symmetric encryption

Cont…

- **Asymmetric keys**
  - Two related keys, a public key and a private key, that are used to perform complementary operations, such as encryption and decryption or signature generation and signature verification
- **Public key certificate**
  - A digital document issued and digitally signed by the private key of a Certification authority that fixes the name of a subscriber to a public key. The certificate indicates that the subscriber identified in the certificate has sole control and access to the corresponding private key

Cont…

- **Public key cryptographic algorithm**
  - A cryptographic algorithm that uses two related keys, a public key and a private key
- **Public key infrastructure**
  - A set of policies, processes, server platform, software and workstations used for the purpose of controlling certificates and public-private key pairs, including the ability to issue, maintain, and cancel public certificate

Public-key cryptosystem

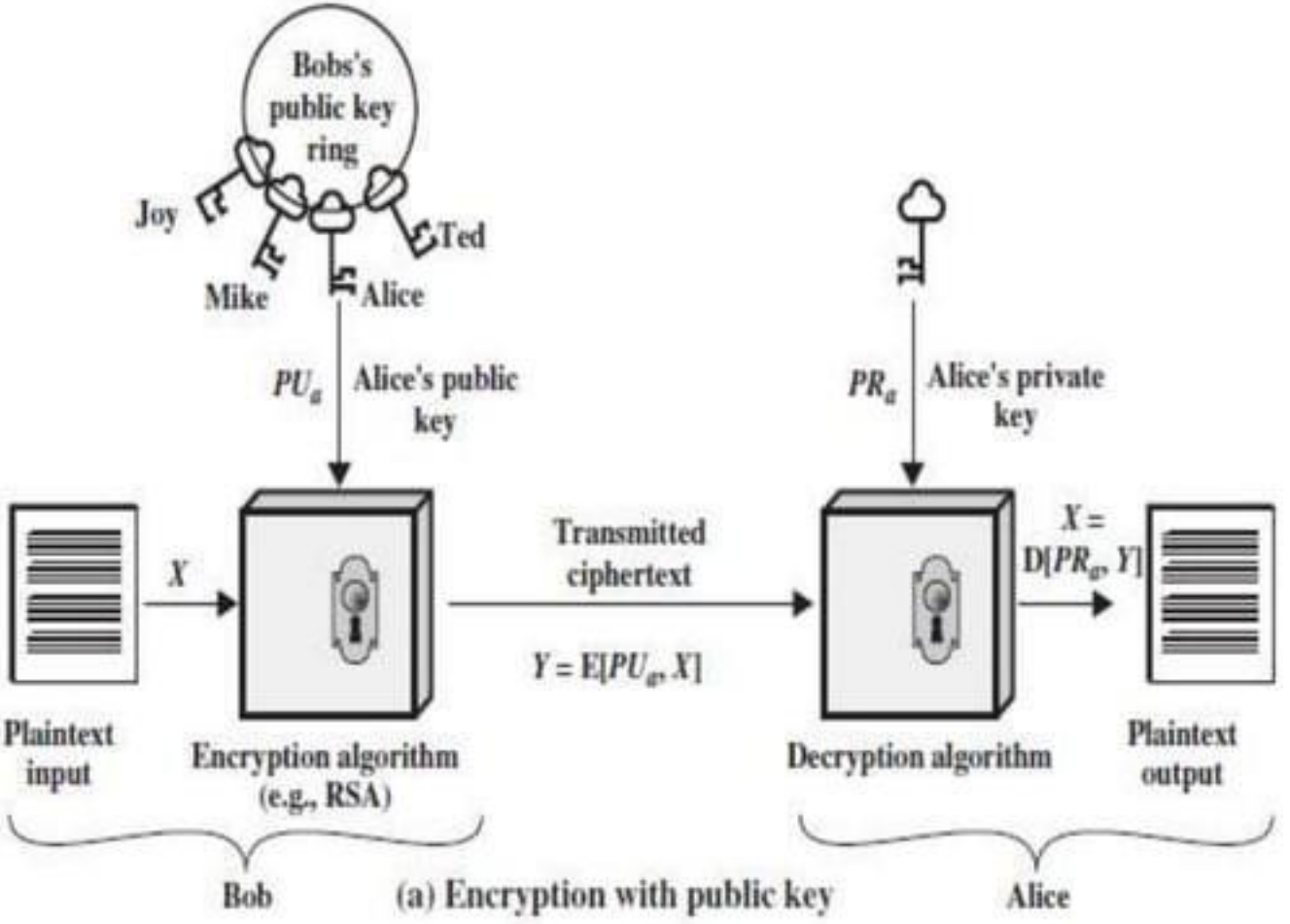
- Asymmetric algorithms rely on one key for encryption and a different but related key for decryption
- These algorithms have the following important characteristics
  - It is computationally infeasible to determine the decryption key given only knowledge of the cryptographic algorithm and the encryption key

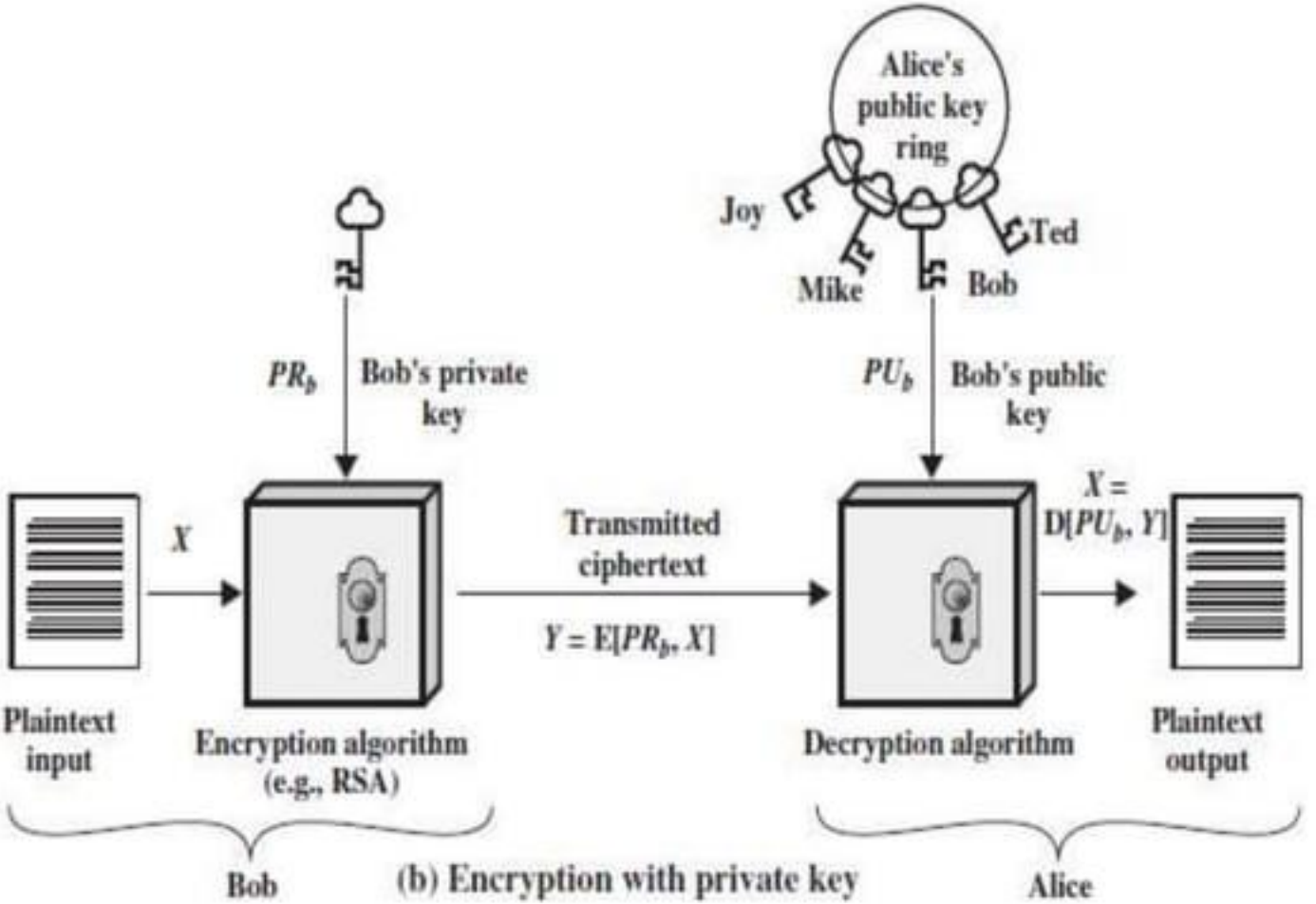
Cont…

- Six ingredients:
- **Plaintext** - This is a readable message or data that is fed(served) into the algorithm as the input
- **Encryption algorithm** - The encryption algorithm performs various transformations on the plaintext
- **Public and private keys** - this is a pair of keys that have been selected so that if one is used for encryption, the other is used for decryption. The exact transformations performed by the algorithm depend on the public key and the private key that is provided as input

Cout..

- **Cipher text** - this is the scrambled message produced as output
- **Decryption algorithm** - The algorithm that accepts the cipher text and matching key and produces the original plain text


Bobs's public key ring
Joy
Ted
Mike
Alice
PU_a
Alice's public key
PR_a
Alice's private key
X
Transmitted ciphertext
$Y = E[PU_a, X]$
$X = D[PR_a, Y]$
Plaintext input
Encryption algorithm (e.g., RSA)
Decryption algorithm
Plaintext output
Bob
(a) Encryption with public key
Alice



(b) Encryption with private key

- The essential steps are

1. Each user generates a pair of keys to be used for the encryption and decryption of messages
2. Each user places one of the two keys in public register or other accessible file. This is public key. The other key is kept private. Each user maintains a collection of public keys obtained from others
3. If Bob wishes to send a confidential message to Alice, Bob encrypts the message using Alice's public key
4. When Alice receives the message, she decrypts it using her private key
5. No other recipient can decrypt the message because only Alice knows her private key

- Here, all participants have access to public keys, and private keys are generated locally by each participant and therefore need never be distributed
- As long as a user's private key remains protected and secret, incoming communication is secure
- At any time, a system can change its private key and publish the related public key to replace its old public key

Difference between conventional encryption and Public-key encryption

Conventional Encryption	Public-Key Encryption
Needed to Work: 1. The same algorithm with the same key is used for encryption and decryption. 2. The sender and receiver must share the algorithm and the key. *Needed for Security:* 1. The key must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus samples of ciphertext must be insufficient to determine the key.	*Needed to Work:* 1. One algorithm is used for encryption and decryption with a pair of keys, one for encryption and one for decryption. 2. The sender and receiver must each have one of the matched pair of keys (not the same one). *Needed for Security:* 1. One of the two keys must be kept secret. 2. It must be impossible or at least impractical to decipher a message if no other information is available. 3. Knowledge of the algorithm plus one of the keys plus samples of ciphertext must be insufficient to determine the other key.

Applications of Public-key Cryptosystem

- A applications are divided in two broad categories:

- Encryption/decryption – The sender encrypts the message with the receiver's public key
- Digital Signature – The sender "signs" a message with its private key

RSA Algorithm use in Public key principle

- Asymmetric key cryptographic algorithm
- Rivest-Shamir-Adleman (RSA) name is given by taking the firstname of its inventors
- It uses prime numbers
- This algorithm is based on the fact that it is easy to find and multiply large prime numbers together, but it is extremely difficult to factor their product
- The private and public keys in RSA are based on very large prime numbers
- The real challenge in RSA is the selection and generation of the public key and private key
- Lets know how private key and public key are generated and, using them, how can we perform encryption and decryption

RSA Algorithm use in Public key principle

1. Choose two prime numbers P and Q
2. Calculate $N = P * Q$
3. Select the public key E (i.e. Encryption key) such that it is not a factor of (P–1) and (Q–1)
4. Select the private key D (i.e. Decryption key) such that the following equation is true

   $(D * E) \bmod (P-1) * (Q-1) = 1$
5. For encryption, calculate the cipher text CT from the plain text PT as follows

   $CT = PT^E \bmod N$
6. Send CT as the cipher text to the receiver
7. For decryption, calculate the plain text PT from the cipher text CT as follows

   $PT = CT^D \bmod N$

Example of RSA

1. **Choose two large prime numbers P and Q**
   - Let P= 7, Q=17
2. **Calculate N = P * Q**
   - N= 7 * 17= 119
3. **Select the public key E such that it is not a factor of (P-1) * (Q-1)**
   - Lets find (7-1) * (17-1)= 6*16= 96
   - The factors of 96 are 2,2,2,2,2 and 3 (because 96 = 2*2*2*2*2*3)
   - Thus we have to choose E such that none of the factors of E is 2 and 3
   - Lets choose E as 5

4. **Select the private key D such that the following equation is true**

   (D * E) mod (P–1) * (Q–1) = 1

   - Lets substitute the values of E, P and Q in the equation
   - We have (D * 5) mod (7-1) * (17-1) = 1
   - i.e. (D * 5) mod (6) * (16) = 1
   - i.e. (D * 5) mod (96) = 1
   - After some calculations, let us take D=77
   - So that (77 * 5) mod (96) = 385 mod 96 = 1

5. **For encryption, calculate the cipher text CT from the plain text PT as follows**

   $CT = PT^E \bmod N$

   - Lets assume that plaintext PT = 10
   - Then, $CT = 10^5 \bmod 119 = 100000 \bmod 119 = 40$

6. **Send CT as the cipher text to the receiver**
   - Send 40 as the cipher text to the receiver

7. **For decryption, calculate the plain text PT from the cipher text CT as follows**

   $PT = CT^D \bmod N$

   - $PT = 40^{77} \bmod 119 = 10$

Key management

Key management

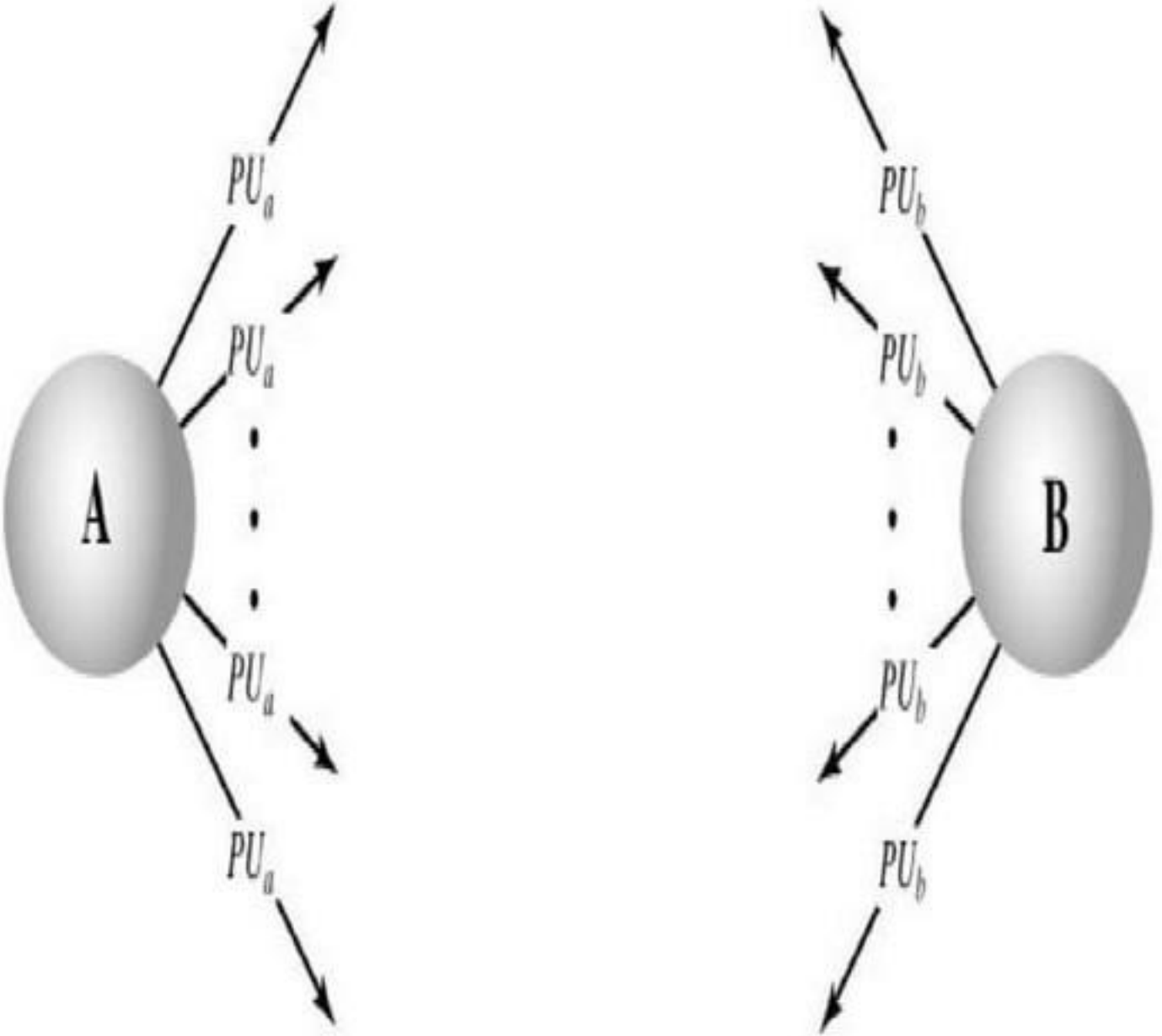
- One of the major roles of public-key encryption has been to address the problem of key distribution
- There are actually two distinct aspects to the use of public-key cryptography in this regard:
  - The distribution of public keys
  - The use of public-key encryption to distribute secret keys

Distribution of Public Keys

- Schemes for Key distribution:
  - Public announcement
  - Publicly available directory
  - Public-key authority
  - Public-key certificates

Public announcement of Public Keys

- 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
- Although this approach is convenient, it has a major weakness
- 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

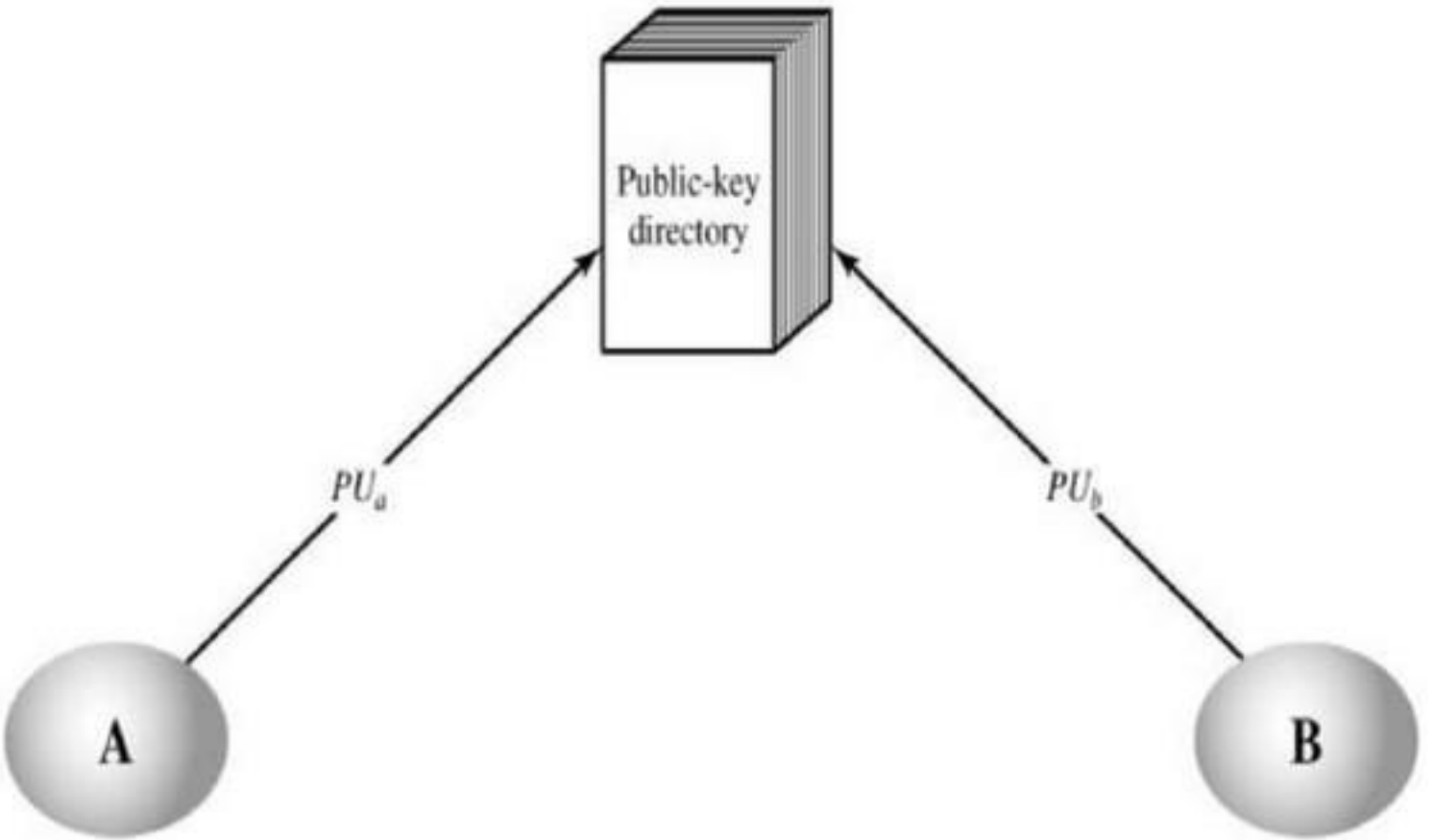

A
PU_a
PU_a
PU_a
PU_a
B
PU_b
PU_b
PU_b
PU_b

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
- Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Each participant registers a public key with the directory authority. Registration would have to be in person or by some form of secure authenticated communication.
3. A participant may replace the existing key with a new one at any time, either because of the desire to replace a public key that has already been used for a large amount of data, or because the corresponding private key has been compromised in some way.
4. Participants could also access the directory electronically. For this purpose, secure, authenticated communication from the authority to the participant is mandatory.

- This scheme is clearly more secure than individual public announcements but still has vulnerabilities
- If an adversary succeeds in obtaining or computing the private key of the directory authority, the adversary could authoritatively pass out counterfeit public keys and subsequently impersonate any participant and eavesdrop on messages sent to any participant


Public-key
directory
PU_a
PU_b
A
B

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, the scenario assumes that a central authority maintains a dynamic directory of public keys of all participants
- Each participant reliably knows a public key for the authority, with only the authority knowing the corresponding private key

- The following steps occur:

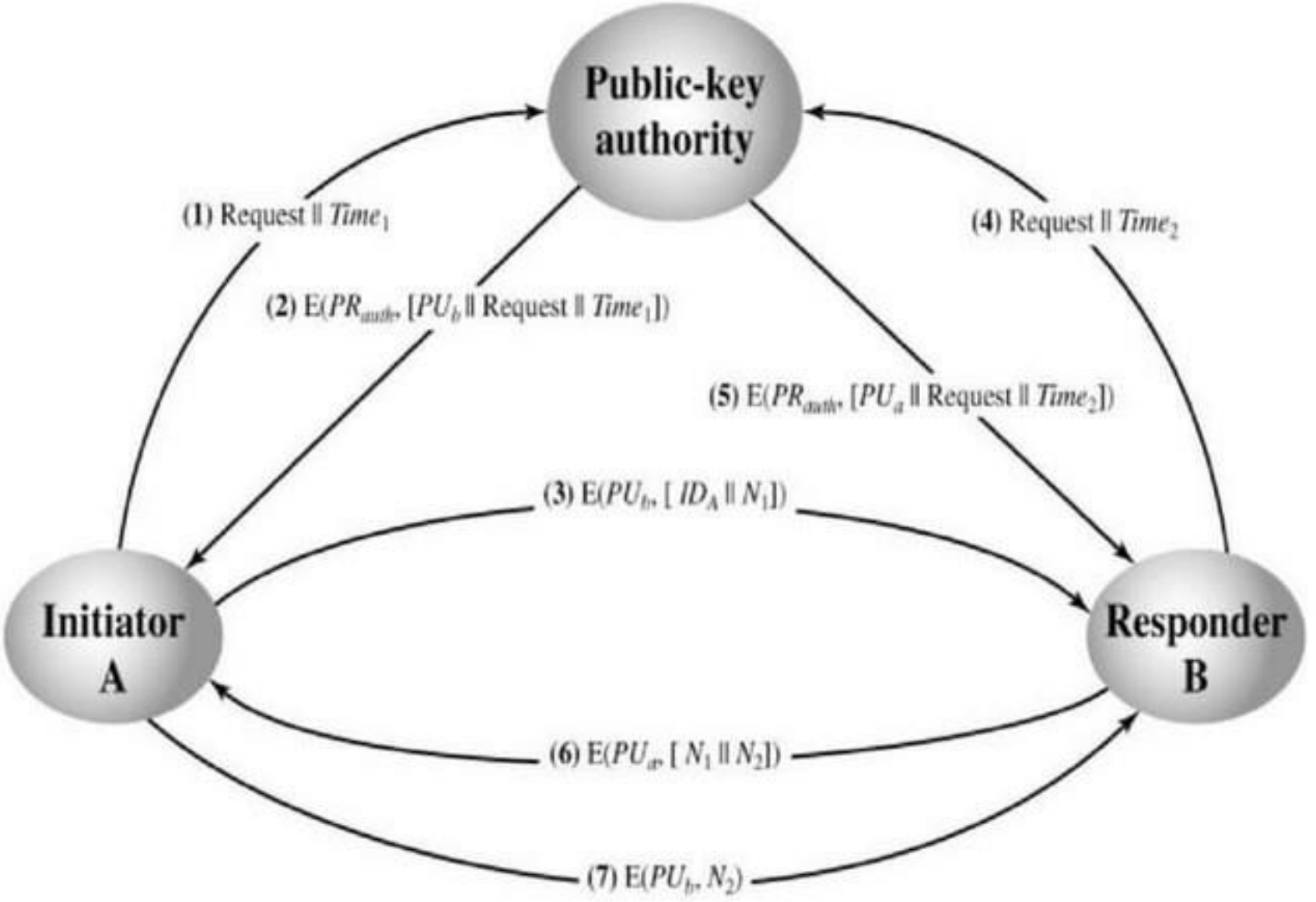
1. A sends a timestamped message to the public-key authority containing a request for the current public key of B.
2. The authority responds with a message that is encrypted using the authority's private key, *PRauth Thus, A is able to decrypt the* message using the authority's public key. Therefore, A is assured that the message originated with the authority. The message includes the following:
    1. B's public key, PUb which A can use to encrypt messages destined for B
    2. The original request, to enable A to match this response with the corresponding earlier request and to verify that the original request was not altered before reception by the authority
    3. The original timestamp, so A can determine that this is not an old message from the authority containing a key other than B's current public key

3. A stores B's public key and also uses it to encrypt a message to B containing an identifier of A (*IDA) and a nonce (N1), which is* used to identify this transaction uniquely.

4, 5. B retrieves A's public key from the authority in the same manner as A retrieved B's public key

6. B sends a message to A encrypted with *PUa and containing A's nonce (N1) as well as a new nonce generated by B (N2) b*ecause only B could have decrypted message (3), the presence of *N1 in message (6) assures A that the correspondent is B.*

7. A returns *N2, encrypted using B's public key, to assure B that its correspondent is A.*


Public-key authority
(1) Request || $Time_1$
(2) E(PR_{auth}, [PU_b || Request || $Time_1$])
(4) Request || $Time_2$
(5) E(PR_{auth}, [PU_a || Request || $Time_2$])
(3) E(PU_b, [ID_A || N_1])
Initiator A
Responder B
(6) E(PU_a, [N_1 || N_2])
(7) E(PU_b, N_2)

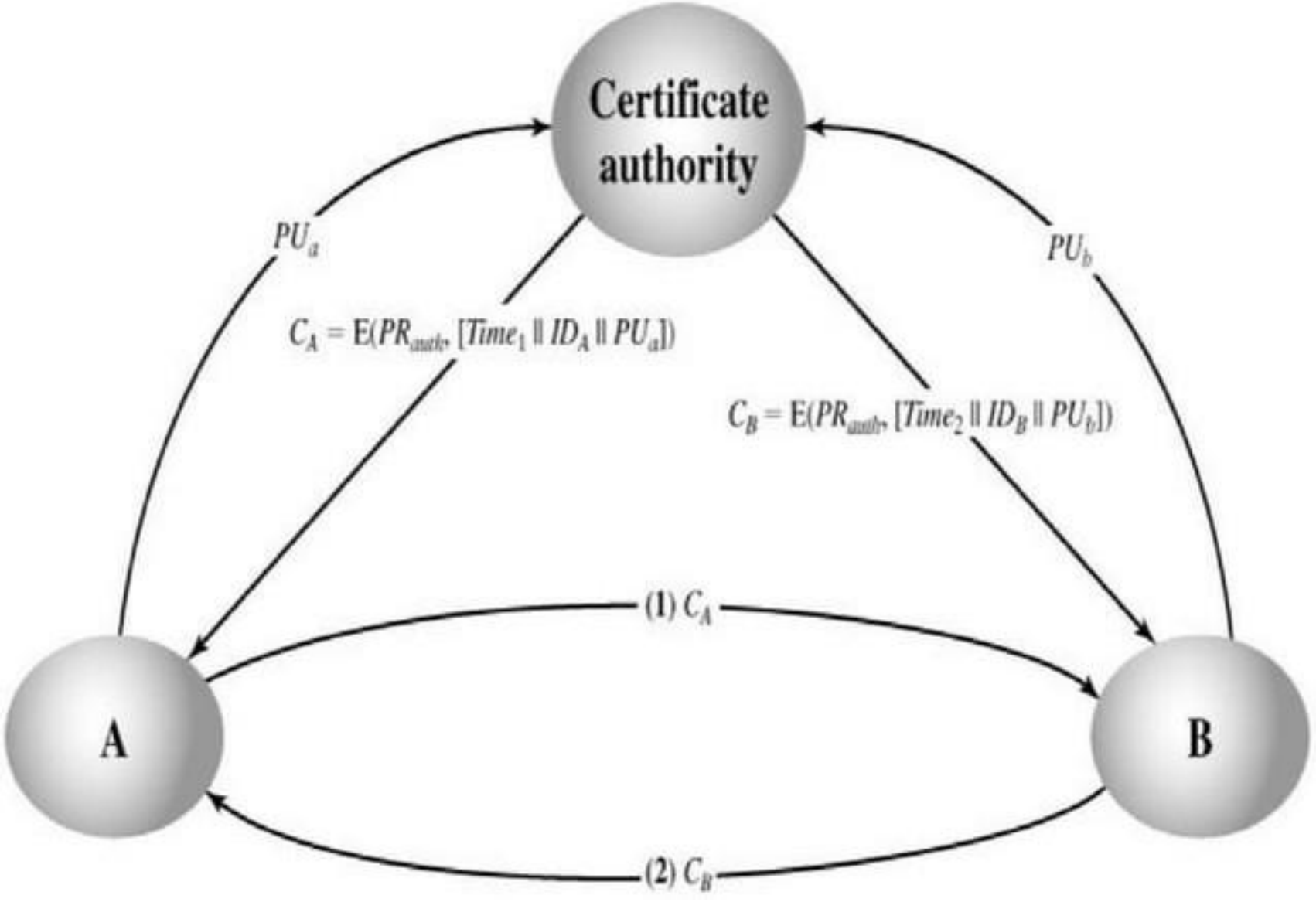
Public-Key Certificates

- 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
- The directory of names and public keys maintained by the authority is vulnerable to tampering.
- An alternative approach is to use **certificates** that can be used by participants to exchange keys without contacting a public-key authority
- A certificate consists of a public key plus an identifier of the key owner, with the whole block signed by a trusted third party
- The third party is a certificate authority, such as a government agency or a financial institution, that is trusted by the user community

- A user can present his or her public key to the authority in a secure manner, and obtain a certificate And then can publish the certificate
- Anyone needed this user's public key can obtain the certificate and verify that it is valid by way of the attached trusted signature
- A participant can also convey its key information to another by transmitting its certificate. Other participants can verify that the certificate was created by the authority

- We can place the following requirements on this scheme:

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


Certificate authority
PU_a
PU_b
$C_A = E(PR_{auth}, [Time_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [Time_2 \| ID_B \| PU_b])$
(1) C_A
A
B
(2) C_B

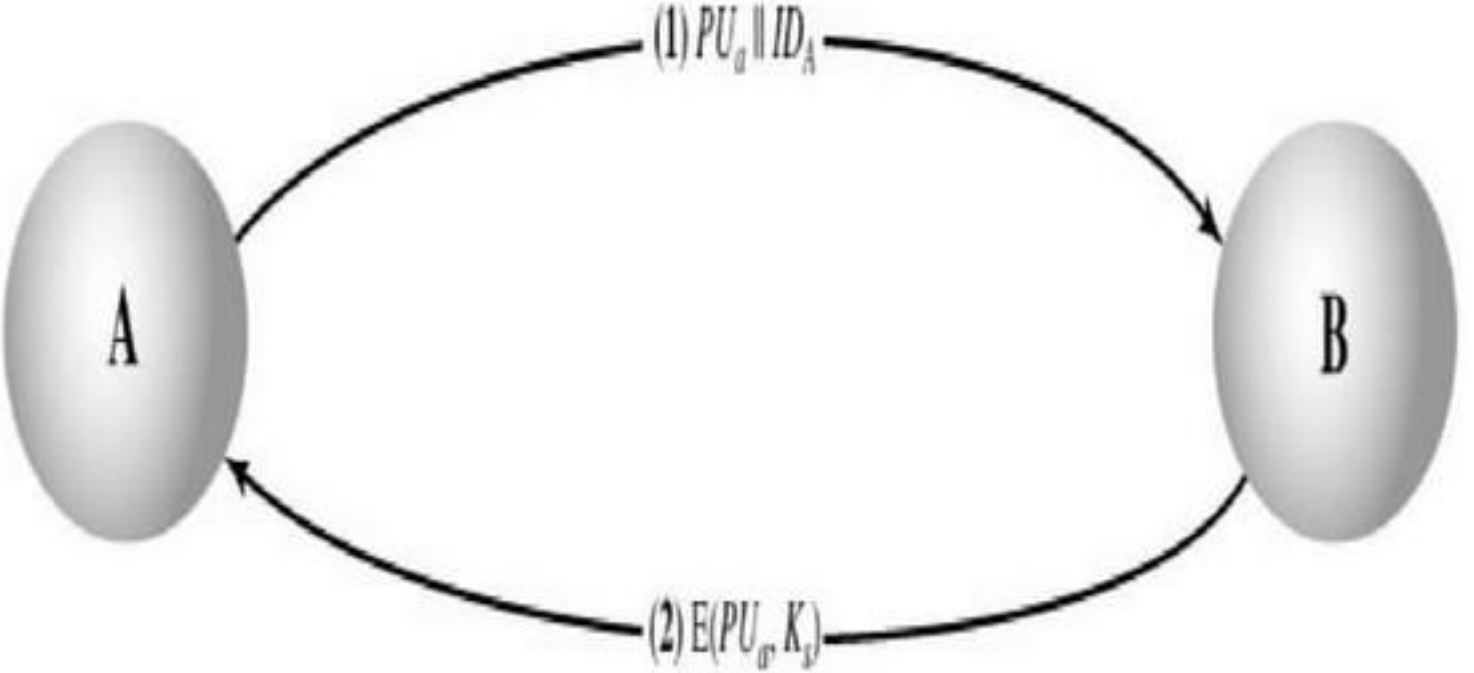
Distribution of Secret Keys Using Public-Key Cryptography

- Simple Secret Key Distribution
- Secret Key Distribution with Confidentiality and Authentication
- A Hybrid Scheme

Simple Secret Key Distribution

- If A wishes to communicate with B, the following procedure is employed:

1. A generates a public/private key pair {PUa, PRa} and transmits a message to B consisting of PUa and an identifier of A, IDA.
2. B generates a secret key, Ks, and transmits it to A, encrypted with A's public key.
3. A computes D(PRa, E(PUa, Ks)) to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of Ks.
4. A discards PUa and PRa and B discards PUa.


(1) $PU_a \| ID_A$
A
B
(2) $E(PU_b, K_s)$

- A and B can now securely communicate using conventional encryption and the session key *Ks*
- At the completion of the exchange, both A and B discard *Ks*.
- Despite its simplicity, this is an attractive protocol. No keys exist before the start of the communication and none exist after the completion of communication.
- The risk of compromise of the keys is minimal

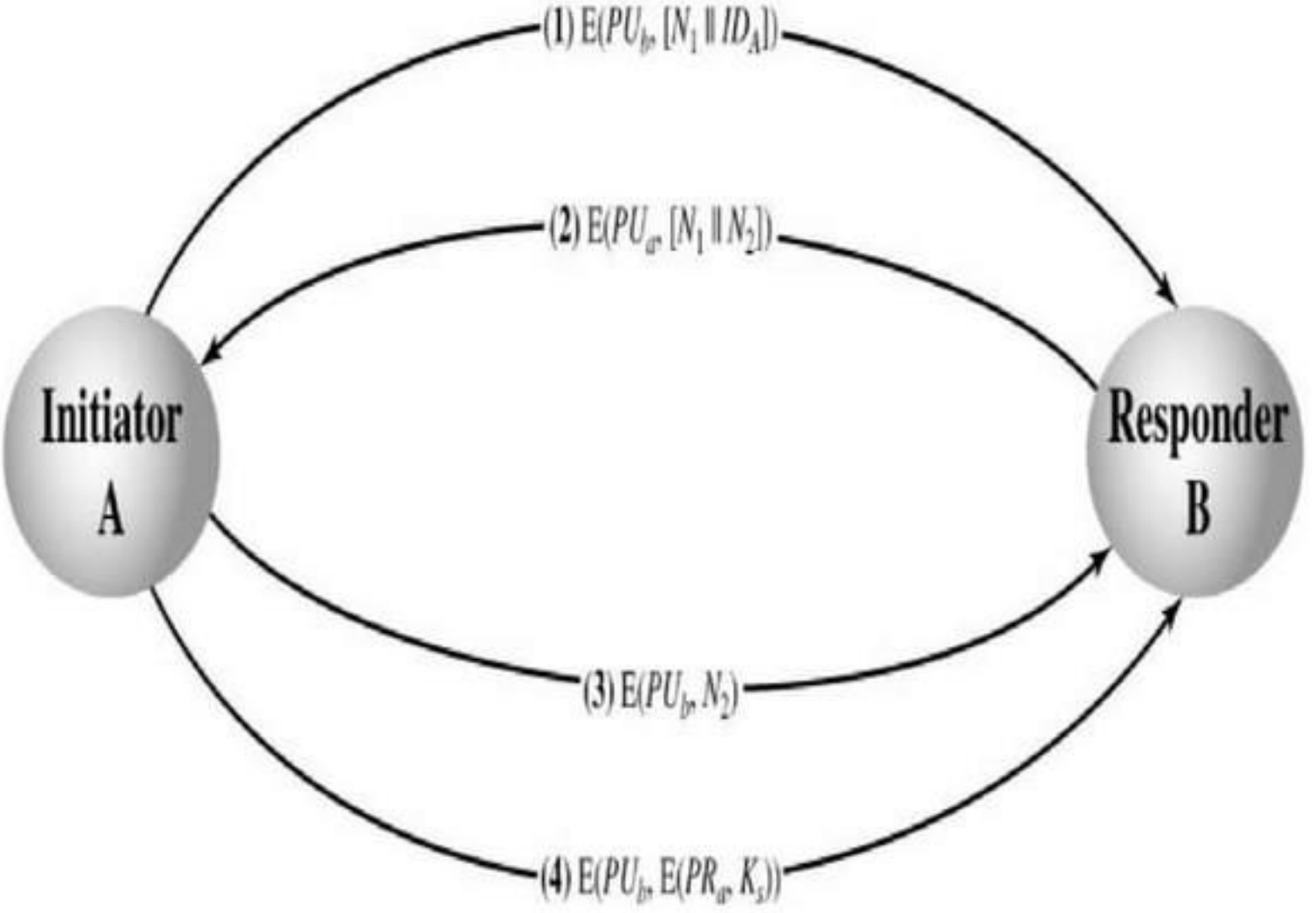
- The protocol depicted in Figure is insecure against an adversary who can intercept messages and then either relay the intercepted message or substitute another message
- Such an attack is known as a **man-in-the-middle attack**
- In this case, If an adversary, E, has control of the intervening communication channel, then E can compromise the communication in the following fashion without being detected:

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

Secret Key Distribution with Confidentiality and Authentication

- It provides protection against both active and passive attacks
- It is assumed that A and B have exchanged public keys by one of the schemes

1. A uses B's public key to encrypt a message to B containing an identifier of A (IDA) 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) as well as 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.
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.
5. B computes D(PUa, D(PRb, M)) to recover the secret key.


(1) E(PU_b, [N_1 || ID_A])
(2) E(PU_a, [N_1 || N_2])
Initiator
A
Responder
B
(3) E(PU_b, N_2)
(4) E(PU_b, E(PR_a, K_s))

A Hybrid Scheme

- This scheme retains the use of a key distribution center (KDC) that shares a secret master key with each user and distributes secret session keys encrypted with the master key
- A public key scheme is used to distribute the master keys
- The following rationale is provided for using this three-level approach:

- **Performance:**
  - There are many applications, especially transaction-oriented applications, in which the session keys change frequently.
  - Distribution of session keys by public-key encryption could degrade overall system performance because of the relatively high computational load of public-key encryption and decryption. With a three-level hierarchy, public-key encryption is used only occasionally to update the master key between a user and the KDC.
- **Backward compatibility:**
  - The hybrid scheme is easily overlaid on an existing KDC scheme, with minimal disruption or software changes.

- The addition of a public-key layer provides a secure, efficient means of distributing master keys
- This is an advantage in a configuration in which a single KDC serves a widely distributed set of users

Diffie-Hellman key exchange Algorithm

- The two parties, who want to communicate securely, can agree on a symmetric key using this technique
- The can then can be used for encryption and decryption
- This algorithm can be used only for key agreement, but not for encryption and decryption
- Once both parties agree on the key to be used, they need to use other symmetric encryption algorithms

Description of algorithm

- Lets assume that Alice and Bob want to agree upon a key to be used for encrypting/decrypting messages that would be exchanged between them
- Then the Diffie-Hellman algorithm works as follows:

1. Firstly, Alice and Bob agree on two large prime numbers, n and g. these two integers need not be kept secret. Alice and Bob can use insecure channel to agree on them
2. Alice chooses another large random number x, and calculates A such that

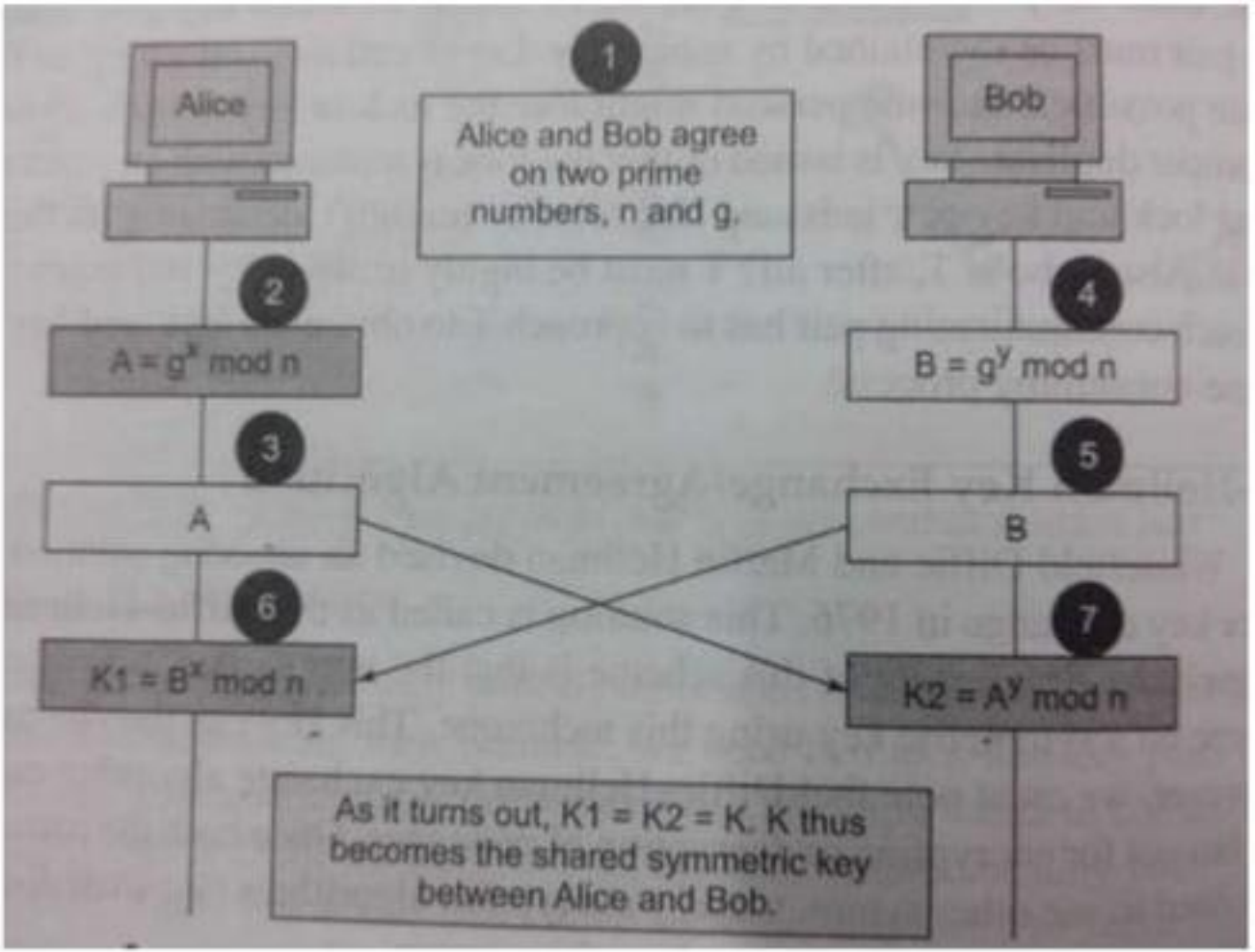
   $A = g^x \bmod n$

3. Alice sends the number A to Bob
4. Bob independently chooses another large random integer y and the n calculates B such that

   $B = g^y \bmod n$
5. Bob sends the number B to Alice
6. A now computes the secret key K1

   $K1 = B^x \bmod n$
7. B now computes the secret key K2

   $K2 = A^y \bmod n$


Alice
Bob
1
Alice and Bob agree on two prime numbers, n and g.
2
$A = g^x \bmod n$
3
A
4
$B = g^y \bmod n$
5
B
6
$K1 = B^x \bmod n$
7
$K2 = A^y \bmod n$
As it turns out, K1 = K2 = K. K thus becomes the shared symmetric key between Alice and Bob.

Example

1. Let n=11, g=7
2. Let x=3. then, we have $A=7^3 \bmod 11=343 \bmod 11 =2$
3. Alice sends 2 to Bob
4. Let y=6. then we have, $B=7^6 \bmod 11=117649 \bmod 11 =4$
5. Bob sends the 4 to Alice
6. We have, $K1=4^3 \bmod 11= 64 \bmod 11 = 9$
7. We have $K2 = 2^6 \bmod 11=64 \bmod 11 = 9$