INFORMATION AND NETWORK SECURITY (2170709)

KEY MANAGEMENT AND DISTRUBITION

Key Management

- Symmetric Key Distribution Using Symmetric Encryption
- Symmetric Key Distribution Using Asymmetric Encryption
- Distribution of Public Keys
- X.509 Certificates

Symmetric Key Distribution Using Symmetric Encryption

- For symmetric encryption to work, the two parties must share the same key, and that key must be protected from access by others.

- Frequent key changes are usually desirable to limit the amount of data compromised if an attacker learns the key. Therefore, the strength of any cryptographic system rests with the *key distribution technique.*

- *key distribution technique* refers to delivering a key to two parties who wish to exchange data without allowing others to see the key.

A Key Distribution Scenario

- Let us assume that user A wishes to establish a logical connection with B and requires a one-time session key to protect the data transmitted over the connection.

- A has a master key, *Ka*, known only to itself and the KDC (Key Distribution Center); similarly, B shares the master key *Kb* with the KDC.

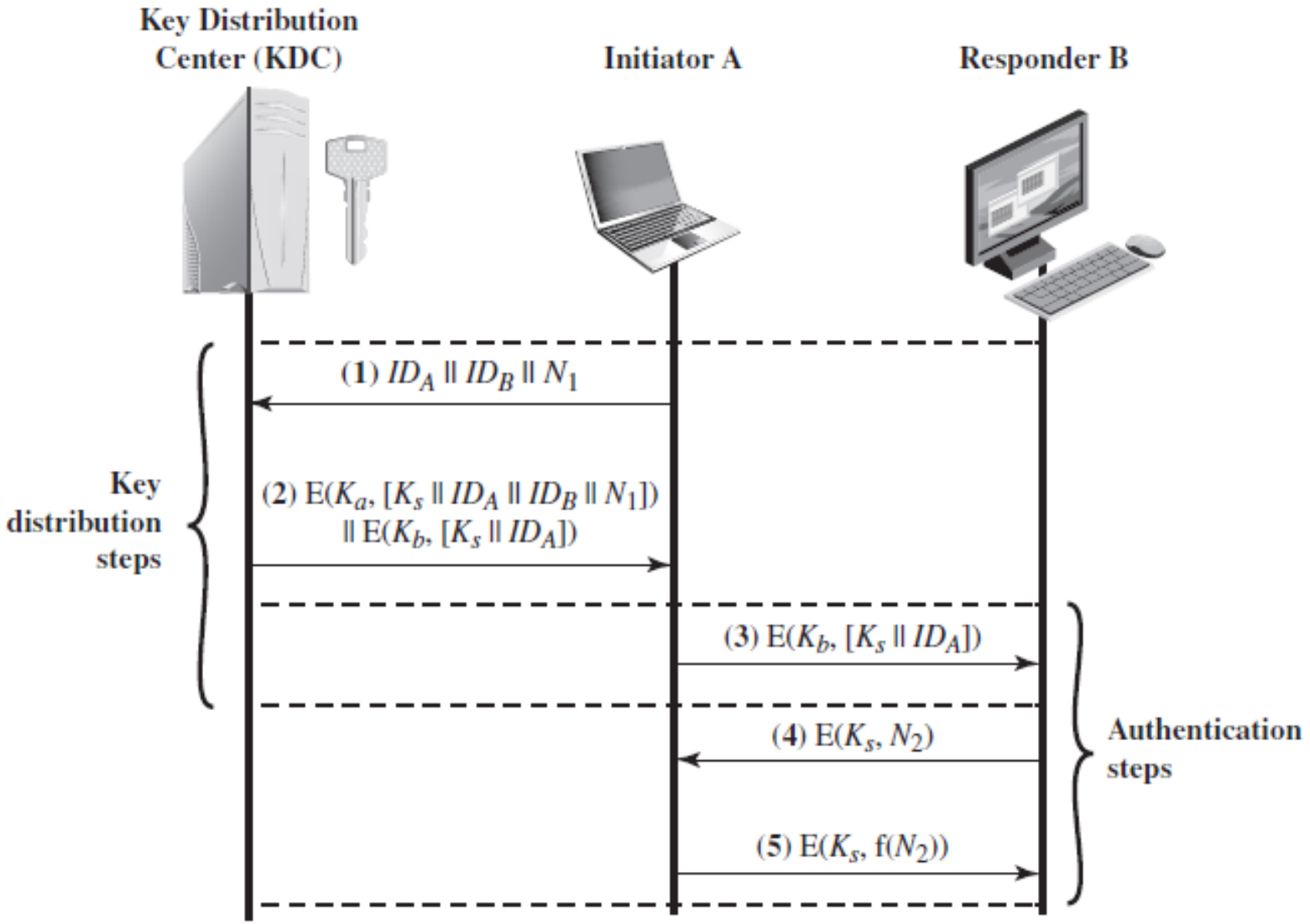


Figure 14.3 Key Distribution Scenario

A Key Distribution Scenario

- 1. *A* issues a request to the *KDC* for a session key to protect a logical connection to *B*. The message includes the identity of A and B and a unique identifier, Nonce N_1. The nonce may be a timestamp, a counter, or a random number.

- 2. The KDC responds with a message encrypted using *Ka*. Thus, *A* is the only one who can successfully read the message, and *A* knows that it originated at the *KDC*. The message includes two items:
  - The one-time session key, *Ks*, to be used for the session
  - The original request message, including the nonce, to enable A to match this response with the appropriate request

A Key Distribution Scenario

- Thus, *A* can verify that its original request was not altered before reception by the *KDC* and, because of the nonce, that this is not a replay of some previous request.

- In addition, the message includes two items intended for B:
  - The one-time session key, *Ks,* to be used for the session
  - An identifier of A (e.g., its network address), ID_A

- These last two items are encrypted with *Kb*. They are to be sent to B to establish the connection and prove A's identity.

A Key Distribution Scenario

- 3. A stores the session key and forwards to B the information namely, *E(Kb [Ks , ID_A])*.
  - B now knows the session key (*Ks)*, knows that the other party is A (from ID_A), and knows that the information originated at the KDC (because it is encrypted using *Kb*).
  - At this point, a session key has been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
- 4. Using the newly minted session key for encryption, B sends a nonce, *N_2, to A.*

A Key Distribution Scenario

- 5. Also, using *Ks*, A responds with $f(N_2)$, where f is a function that performs some transformation on N_2 (e.g., adding one).
  - These steps assure B that the original message it received (step 3) was not a replay.

- Note that the actual key distribution involves only steps 1 through 3, but that steps 4 and 5, as well as step 3, perform an authentication function.

Session Key Lifetime

- The more frequently session keys are exchanged, the more secure they are.
- On the other hand, the distribution of session keys delays the start of any exchange and places a burden on network capacity.

- For *connection-oriented protocols*, one obvious choice is to use the same session key for the length of time that the connection is open, using a new session key for each new session.

- If a logical connection has a very long lifetime, then it would be prudent to change the session key periodically.

Session Key Lifetime

- For a *connectionless protocol*, such as a transaction-oriented protocol, there is no explicit connection initiation or termination.
- The most secure approach is to use a new session key for each exchange. However, this increases overhead and delay.
- A better strategy is to use a given session key for a certain fixed period only or for a certain number of transactions.

A Transparent Key Control Scheme

- The approach assumes that communication makes use of a connection-oriented end-to-end protocol, such as TCP.
- A security service module (SSM) performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

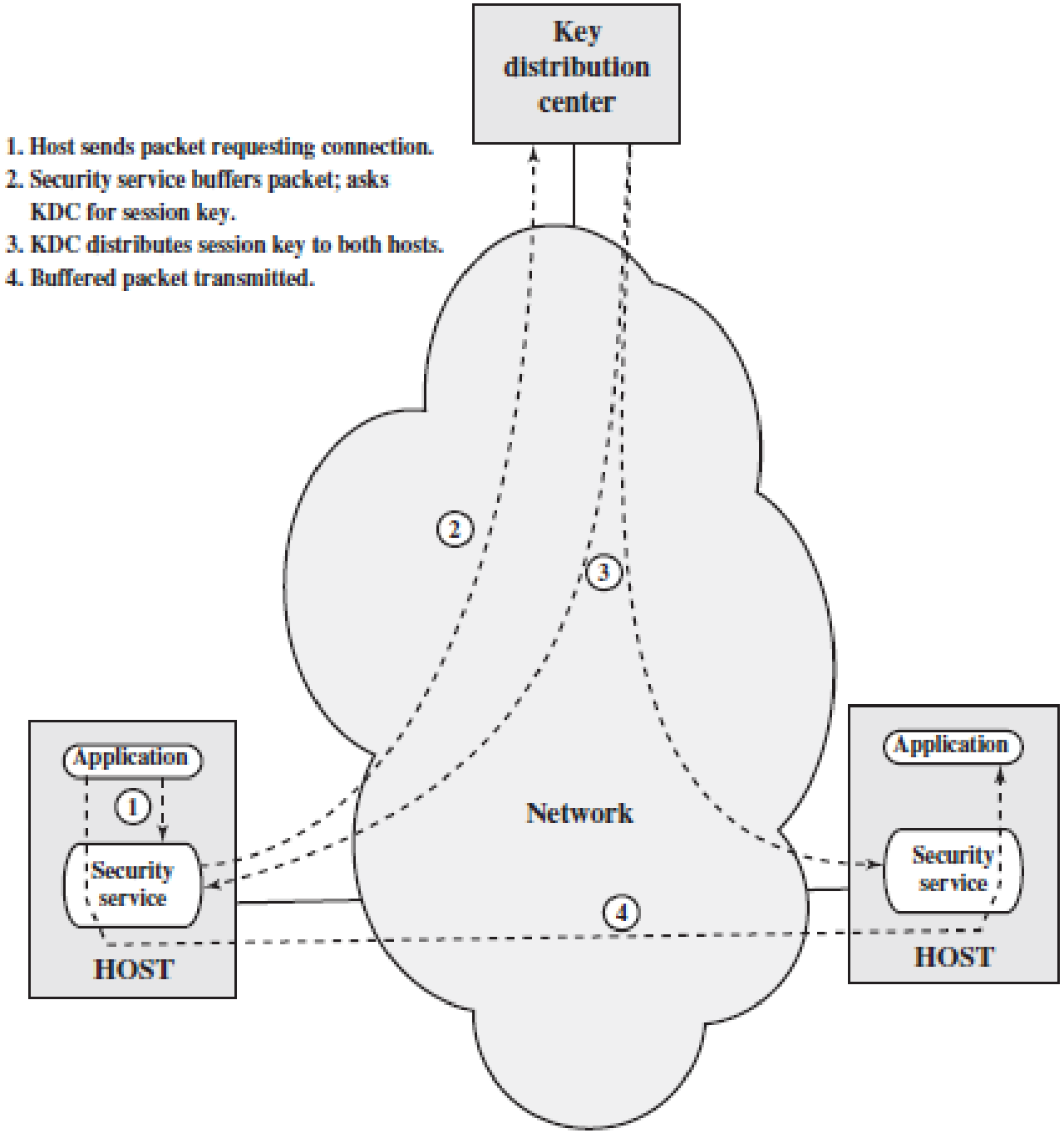


Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

A Transparent Key Control Scheme

- Step 1: When one host wishes to set up a connection to another host, it transmits a connection-request packet.
- Step 2: The SSM saves that packet and applies to the KDC for permission to establish the connection. The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.
- Step 3: If the KDC approves the connection request, it generates the session key and delivers it to the two appropriate SSMs, using a unique permanent key for each SSM.
- Step 4: The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems.
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.

Decentralized Key Control

- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential end systems for purposes of session key distribution.

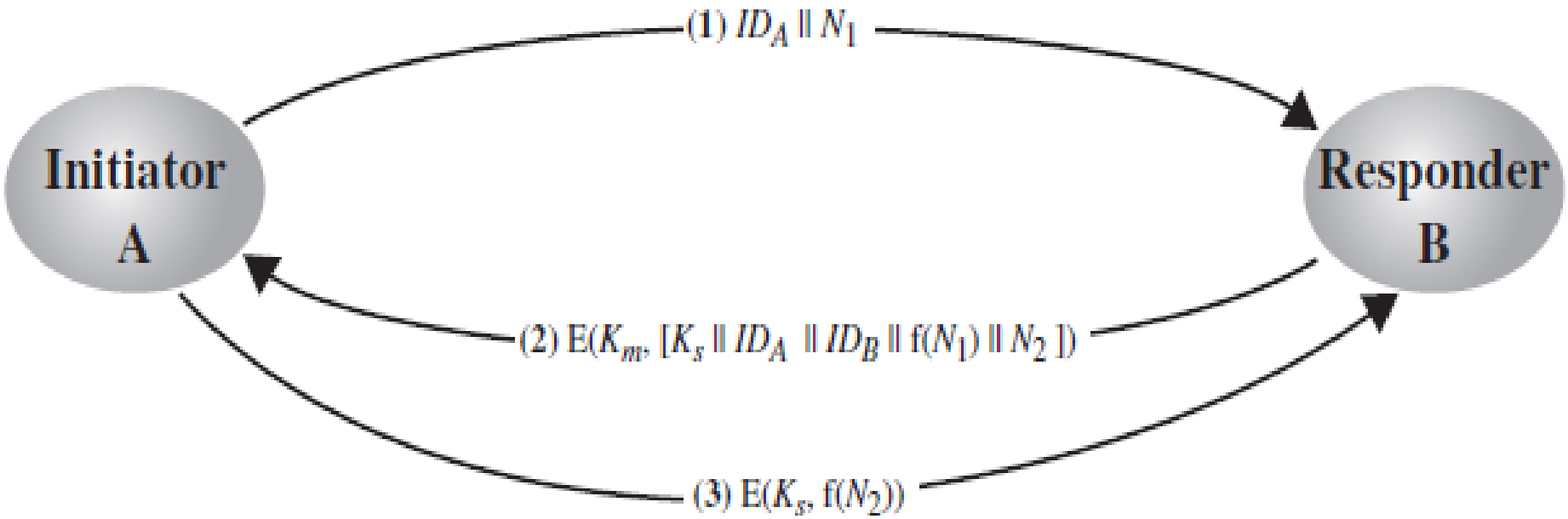


Figure 14.5 Decentralized Key Distribution

Decentralized Key Control

- 1. A issues a request to B for a session key and includes a nonce, N_1.
- 2. B responds with a message that is encrypted using the shared master key *Km*. The response includes the session key selected by B, an identifier of B, the value $f(N_1)$, and another nonce, N_2.
- 3. Using the new session key, A returns f(N_2) to B.

Controlling Key Usage

- We may wish to define different types of session keys on the basis of use, such as
  - Data-encrypting key
  - PIN-encrypting key
  - File-encrypting key

- It may be desirable to specify controls in systems that limit the ways in which keys are used.
  - Key Tags
  - Control Vector

Controlling Key Usage

- Key Tag Technique:
- Associate a tag with each key
- The proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key.

- The bits have the following interpretation:
  - One bit indicates whether the key is a session key or a master key.
  - One bit indicates whether the key can be used for encryption.
  - One bit indicates whether the key can be used for decryption.
  - The remaining bits are spared for future use.

Controlling Key Usage

- Because the tag is embedded in the key, it is encrypted along with the key when that key is distributed, thus providing protection. The drawbacks of this scheme are
  - 1. The tag length is limited to 8 bits, limiting its flexibility and functionality.
  - 2. Because the tag is not transmitted in clear form, it can be used only at the point of decryption, limiting the ways in which key use can be controlled.

Controlling Key Usage

- Control Vector Technique:
- Each session key has an associated control vector consisting of a number of fields that specify the uses and restrictions for that session key.
- The length of the control vector may vary.

- The control vector is cryptographically coupled with the key at the time of key generation at the KDC.

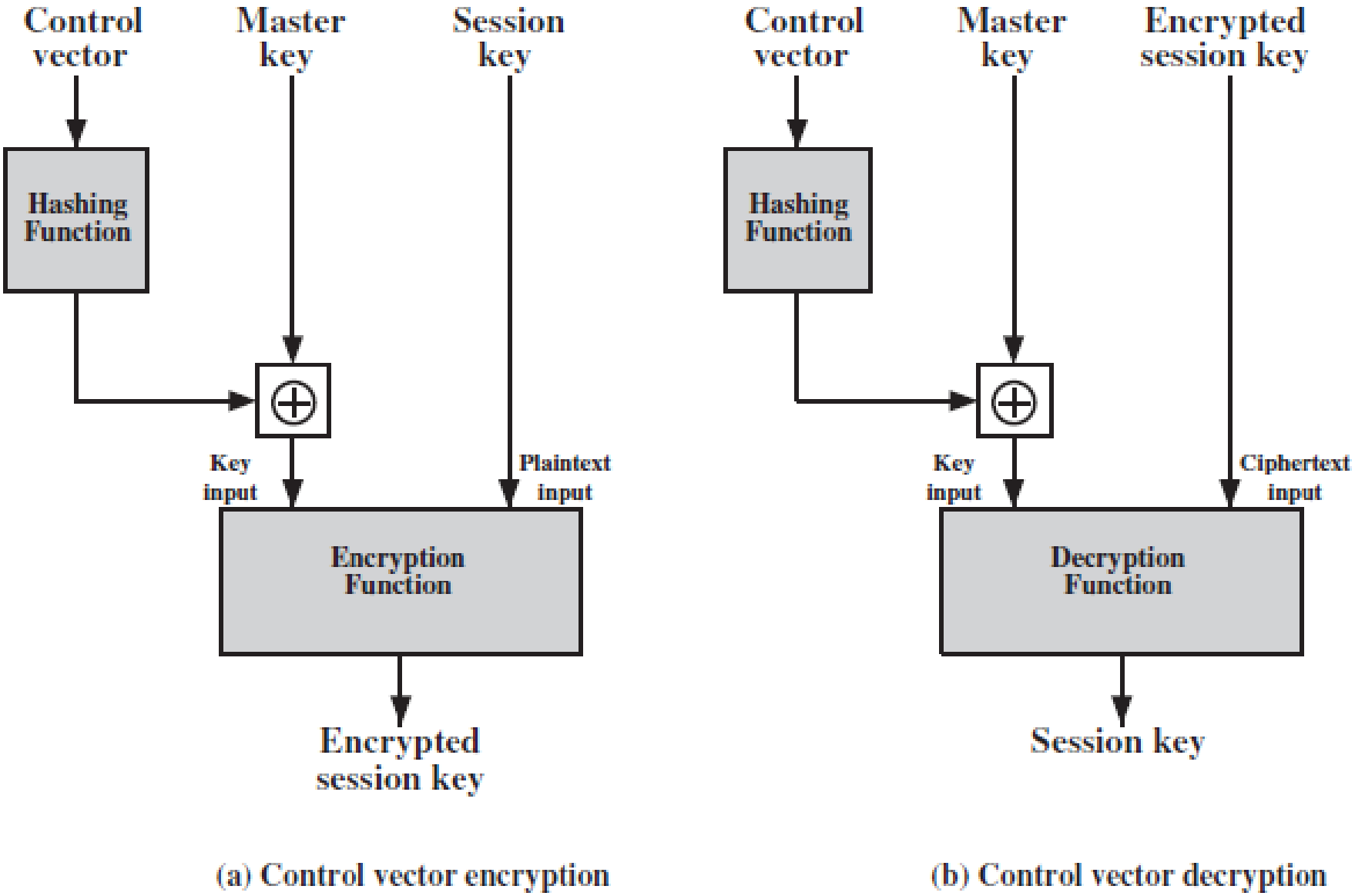


Figure 14.6 Control Vector Encryption and Decryption

Controlling Key Usage

- Step 1: the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length.

- Step 2: The hash value is then XORed with the master key to produce an output that is used as the key input for encrypting the session key.
- Thus,
  - Hash value = $H = h(CV)$
  - Key input = $Km \; XOR \; H$
  - Ciphertext = $E([Km \; XOR \; H], Ks)$
  - where *Km is the master key and Ks is the session key.*

Controlling Key Usage

- Step 3: The session key is recovered in plaintext by the reverse operation:
  - D([*Km XOR H*], *E*([*Km XOR H*], *Ks*))

- Use of the control vector has two advantages over use of an 8-bit tag.
  - There is no restriction on length of the control vector.
  - The control vector is available in clear form at all stages of operation. Thus, control of key use can be exercised in multiple locations.

Symmetric Key Distribution Using Asymmetric Encryption

- Schemes used:
  - Simple secret key distribution
  - Secret key distribution with Confidentiality and Authentication

Simple Secret Key Distribution

- *A* generates a public/private key pair $\{PU_A, PR_A\}$ and transmits a message to B consisting of PU_A and ID_A
- *B* generates a secret key *Ks* and transmits it to *A*, encrypted with *A*'s public key

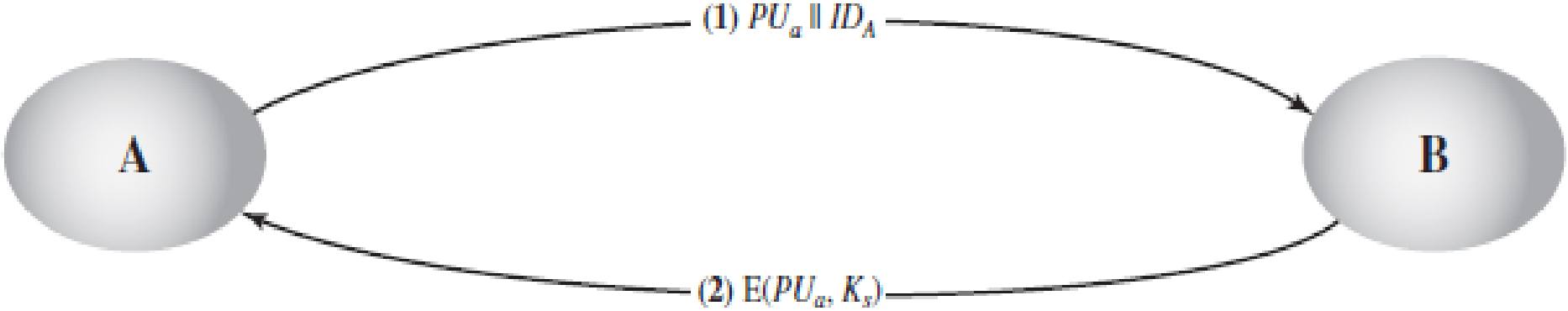
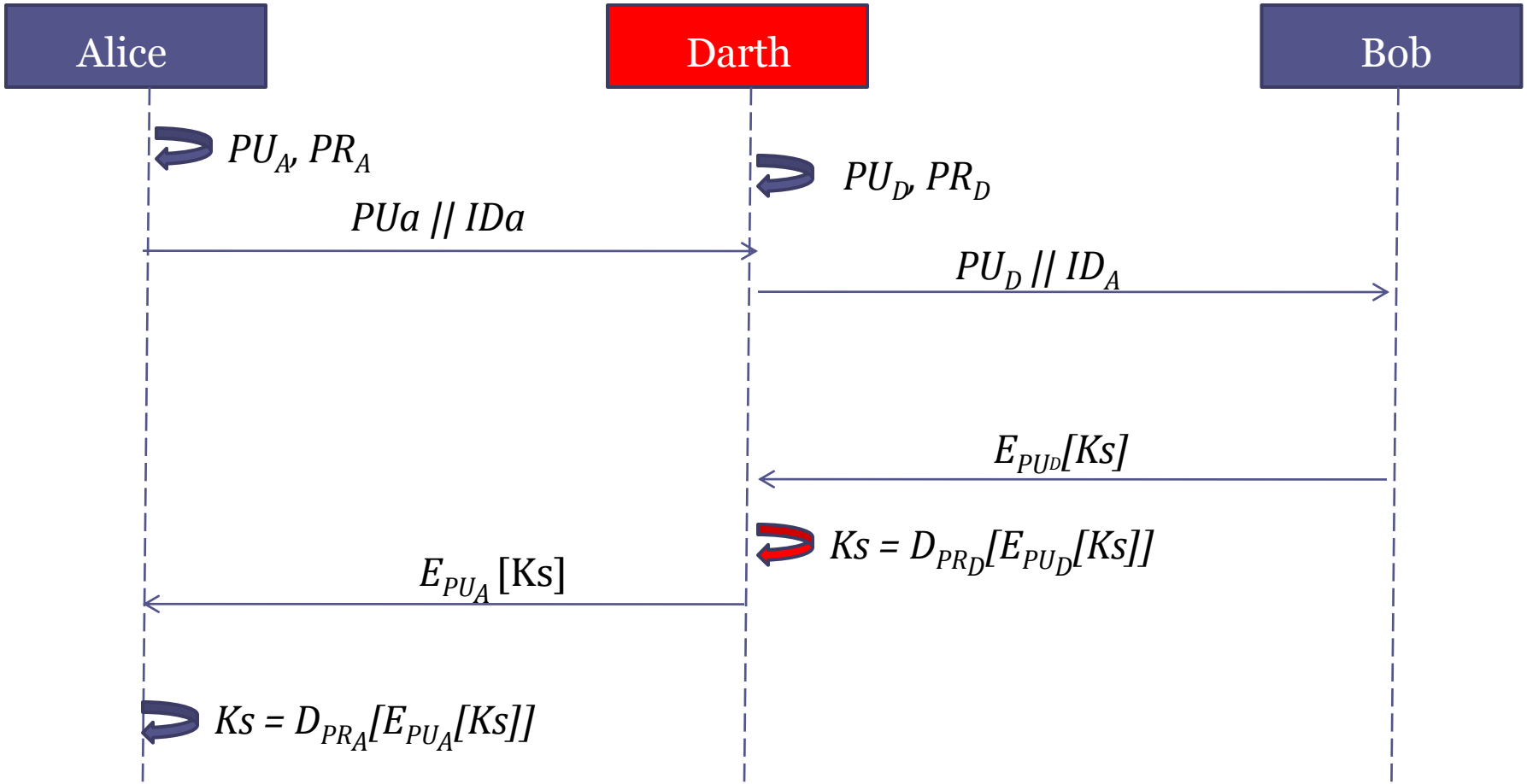


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

Simple Secret Key Distribution

- *A* computes $D_{PR_A}[E_{PU_A}[Ks]]$ to recover the secret key
- *A* discards PU_A and PR_A and *B* discards PU_A
- *A* and *B* securely communicates using session key *Ks*
- At the completion of communication, both *A* and *B* discard *Ks*
- No key exists before or after the communication. Hence, the risk of compromise of key is minimum.

Simple Secret Key Distribution:
Weakness



Simple Secret Key Distribution

- Weakness
  - If an opponent E has the control of the intervening communication channel, then E can do the following:
    - *A* generates a public/private key pair $\{PU_A, PR_A\}$ and transmits a message intended to *B* consisting of PU_A and ID_A
    - *E* intercepts the message, create its own public/private key pair $\{PU_E, PR_E\}$ and transmits $PU_E \,||\, ID_A$ to *B*

Simple Secret Key Distribution

- B generates a secret key Ks and transmits it to A, encrypted with public key PU_E
- E intercepts the message and learns Ks by computing $D_{PR_E}[E_{PU_E}[Ks]]$
- E transmits E_{PU_A} [Ks] to A
- Both A and B knows Ks but are unaware that Ks has been known to E also
- Knowing Ks, E can decrypt all messages

Secret Key Distribution with Confidentiality and Authentication

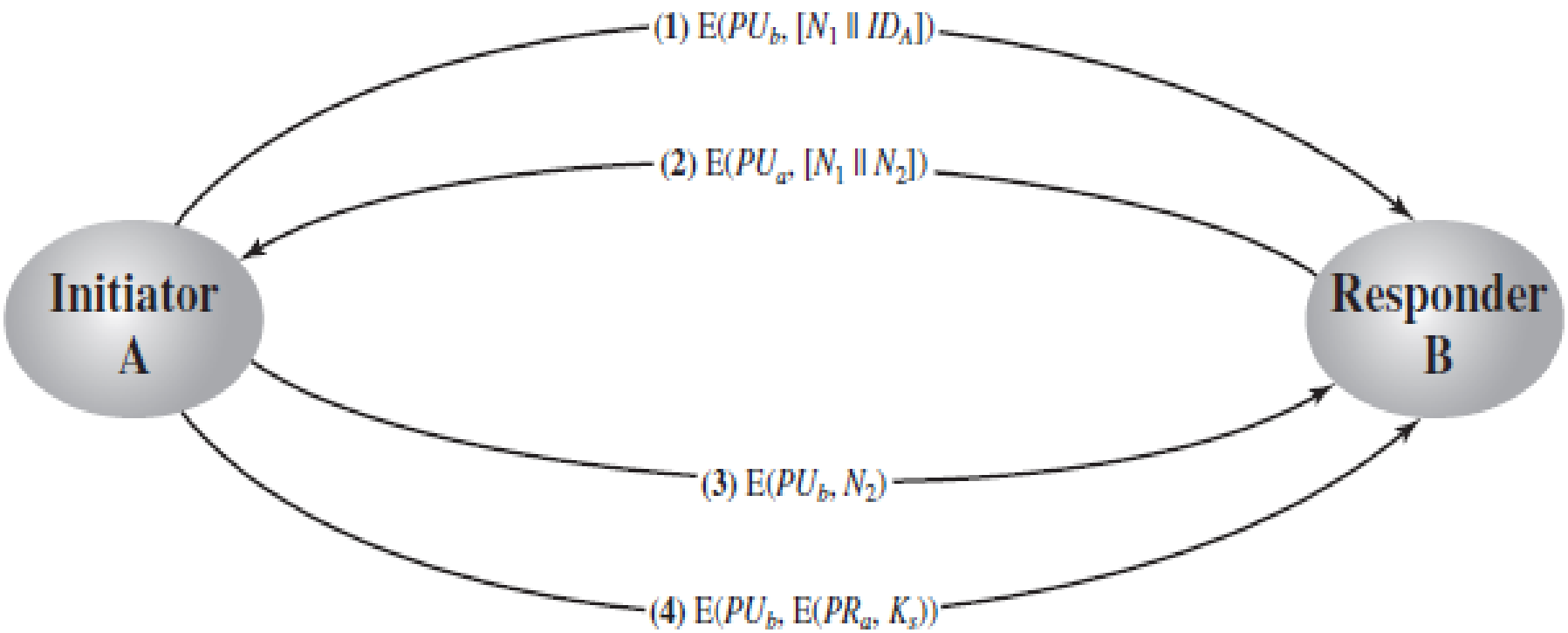


Figure 14.9 Public-Key Distribution of Secret Keys

Secret Key Distribution with Confidentiality and Authentication

- A uses B's public key to encrypt a message containing ID_A and a nonce N_1
- B sends message to A encrypted with KU_A and containing A's nonce N_1 as well as a new nonce N_2
- The presence of N_1 in message 2 assures A that the correspondent is B only
- A returns N_2, encrypted with B's public key to assure B that its correspondent is A only

Secret Key Distribution with Confidentiality and Authentication

- *A* selects a secret key *Ks* and sends to *B*

$$M = E_{KU_B}[E_{KR_A}[Ks]]$$

- Encryption of this message with *B*'s public key ensures that only *B* can read it, thus providing confidentiality

- Encryption with *A*'s private key ensures that only *A* could have sent it, thus providing authentication

Distribution of Public Keys

- Public keys are known to everyone.

- Schemes used for Public Key distribution:
  - Public Announcement
  - Publicly Available Directory
  - Public Key Authority
  - Public Key Certificates

Public Announcement

- Any participant can send his public key to any other participant or broadcast the key

- Weakness:
  - Anyone can make use of this public announcement
  - Some unauthorized user can pretend to be user *A* and broadcast a public key, thus able to read all encrypted messages intended to user *A*.

Public Announcement

Figure 14.10 Uncontrolled Public-Key Distribution

Fig. Public key distribution using Public Announcement

Publicly Available Directory

- A publicly available dynamic directory is maintained for all public keys.
- Maintenance and distribution of directory is responsibility of some trusted authority.

Publicly Available Directory

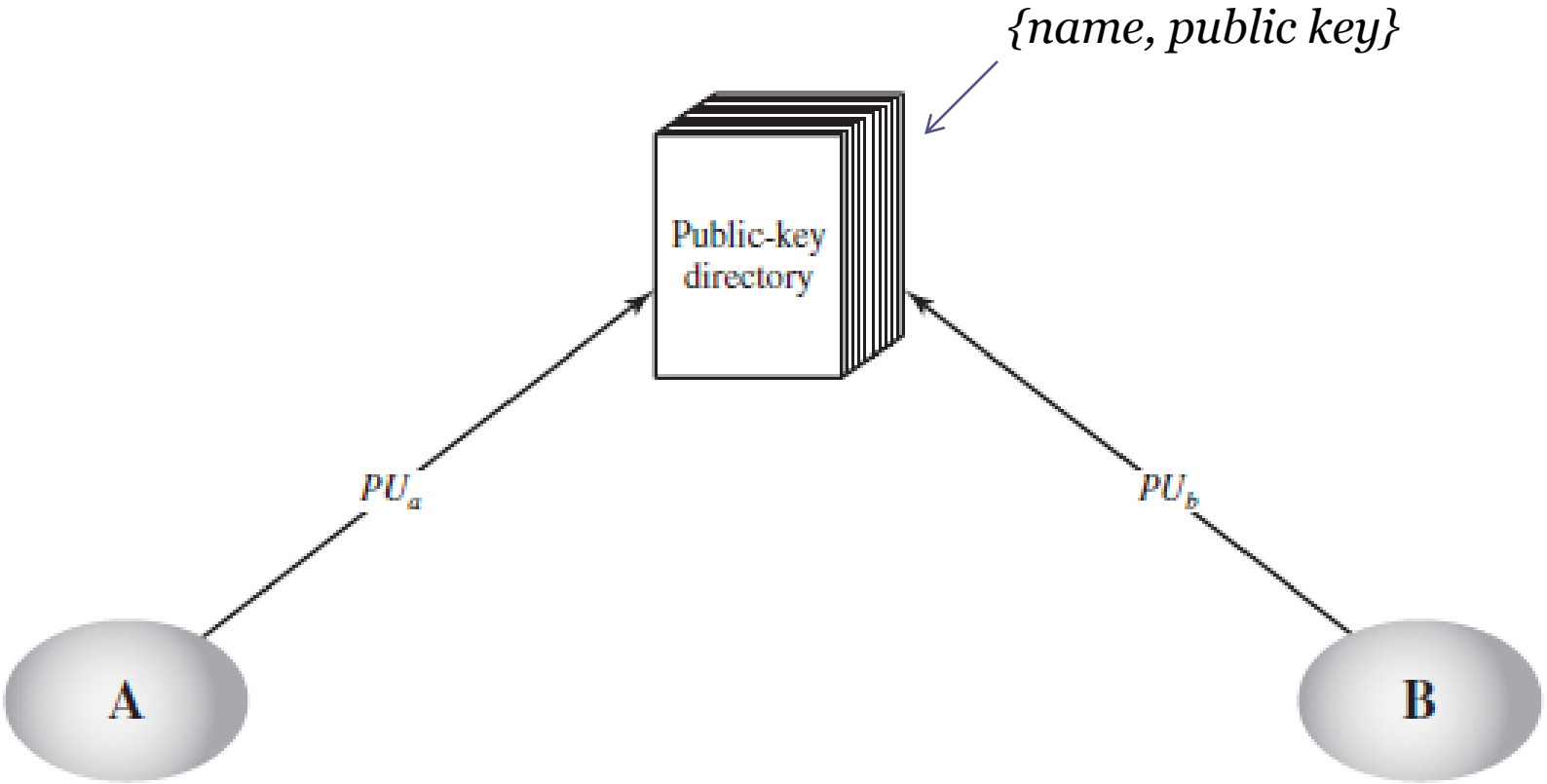


Figure 14.11 Public-Key Publication

Fig. Public key distribution using Publicly Available Directory

Publicly Available Directory

- The authority maintains a directory with a *{name, public key}* entry for each participant
- Each participant registers a public key with the directory authority
- A participant may replace the existing key with a new one, at any time
- The authority publishes the entire directory or updates it
- Participants can access the directory electronically

Publicly Available Directory

- Secure than Public Announcement scheme
- Weakness:
  - If an opponent succeeds in obtaining the private key of the authority, then he could pass fake public keys to participants.

Public Key Authority

- The central authority maintains a dynamic directory of public keys of all participants
- This directory is not publicly available
- Each participant knows the public key of the authority

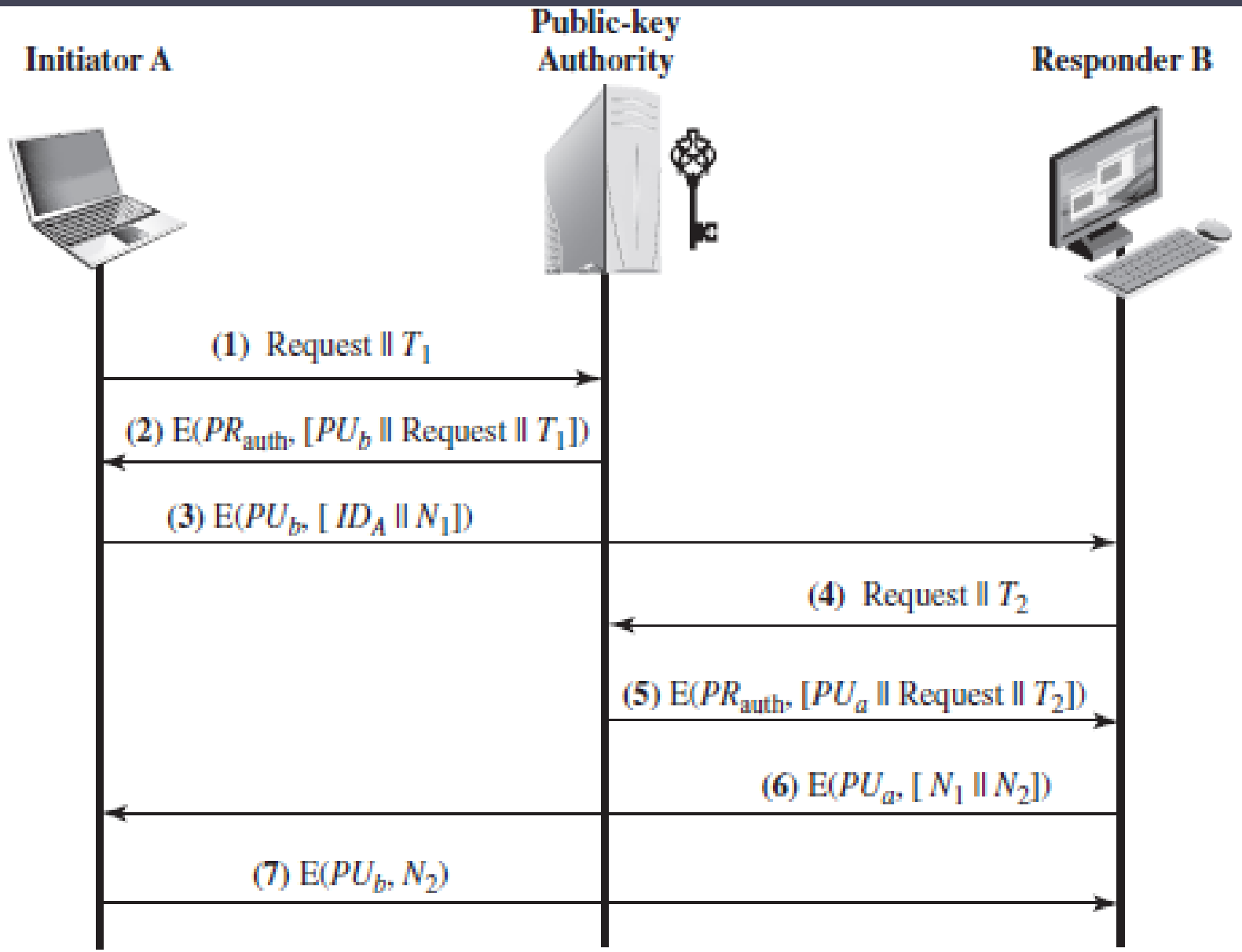


Figure 14.12 Public-Key Distribution Scenario

Public Key Authority

- Participant *A* sends a time-stamped message to the public key authority and requests for the current public key of participant *B*
- Authority responds with a message encrypted with its private key KR_{Auth}
- *A* is able to decrypt the message using authority's public key. So, *A* is assured that message is originated from trusted authority
- Message includes
  - *B*'s public key
  - Original request
  - Original time-stamp

Public Key Authority

- *A* stores *B*'s public key and uses it to encrypt a message.
- Message contains
  - Identity of *A* (ID_A)
  - A nonce (N_1)
- *B* receives *A*'s public key from authority in the same manner
- *B* sends a message to *A* encrypted with KU_A and containing *A*'s Nonce N_1 and a new nonce N_2
- The presence of N_1 in *B*'s message assures *A* that the correspondent is *B*
- A returns N_2, encrypted using *B*'s public key to assure *B* that *A* has sent a message

Public Key Authority

- Weakness
  - A user must appeal to the authority for a public key for every other user

Public Key Certificates

- Participants can exchange their public keys without contacting the authority
- Each participant applies to a certificate authority by sending its public key and requesting a certificate

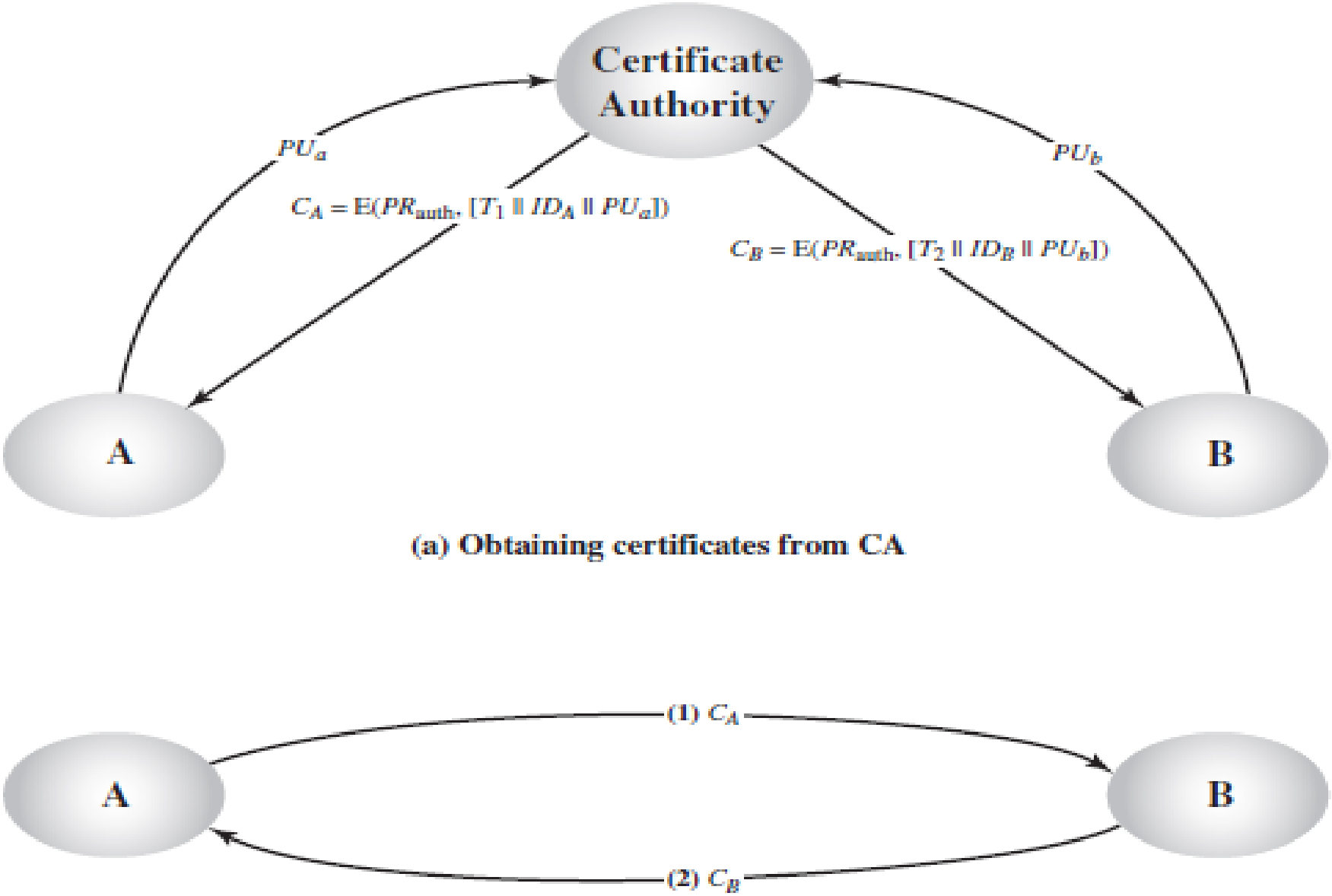


Figure 14.13 Exchange of Public-Key Certificates

Public Key Certificates

- For participant *A*, authority provides a certificate

$$C_A = E_{KR_{Auth}}[T, ID_A, KU_A]$$

- *A* passes this certificate to *B*

- *B* decrypts *A*'s certificate using authority's public key

$$D_{KU_{Auth}}[C_A] = D_{KU_{Auth}}[E_{KR_{Auth}}[T, ID_A, KU_A]]$$
$$= (T, ID_A, KU_A)$$

- *B* is assured that certificate was originated from the authority only.

X.509 Authentication Service

- X.509 defines a framework for the provision of authentication services by the X.500 directory to its users.
- The directory may serve as a repository of public-key certificates.
- Each certificate contains the public key of a user and is signed with the private key of a trusted certification authority.

X.509 Certificates

- The heart of the X.509 scheme
  - Associated with each user
- Created by some trusted certification authority(CA) and placed in the directory by CA or by the user
- The directory is merely storing certificate

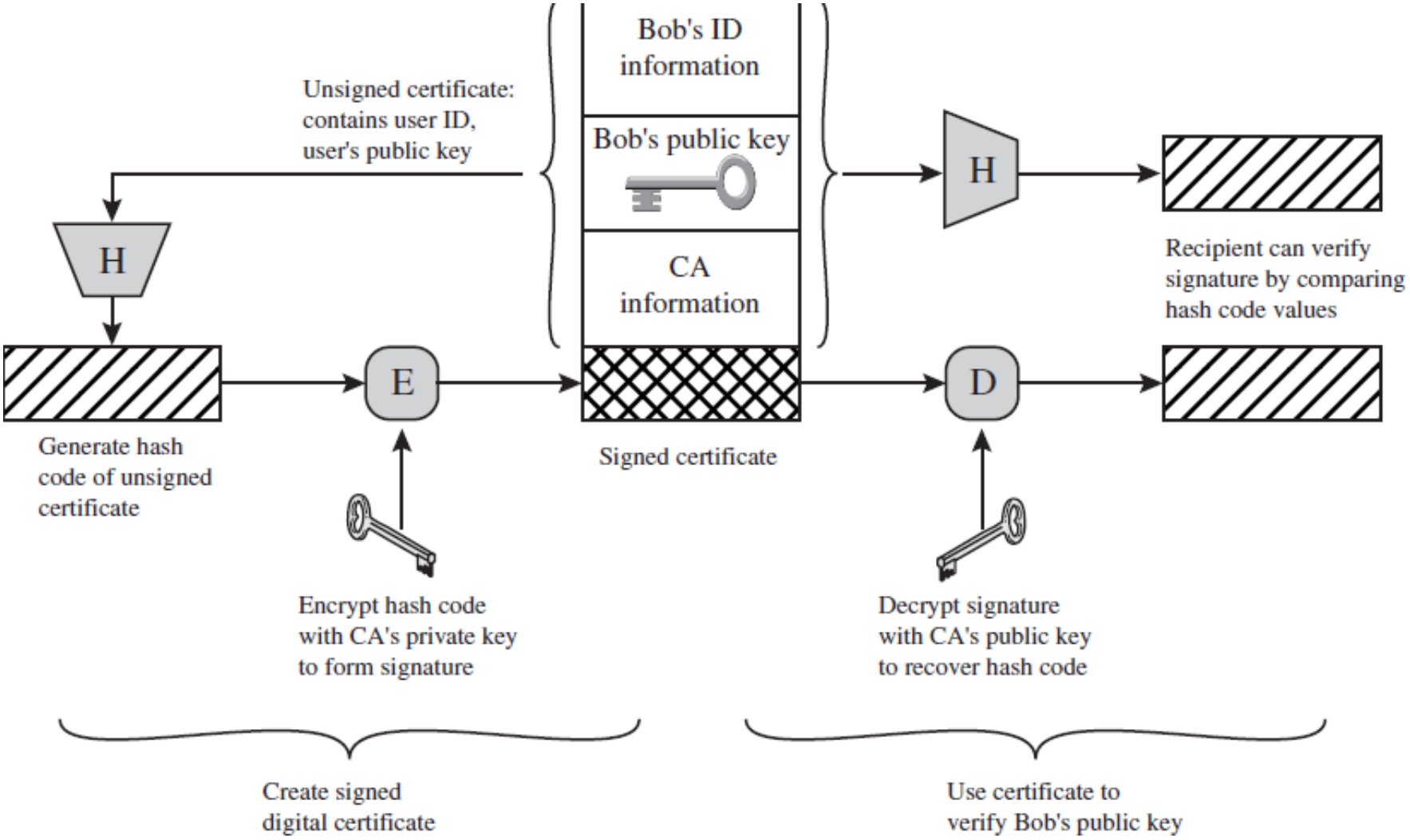
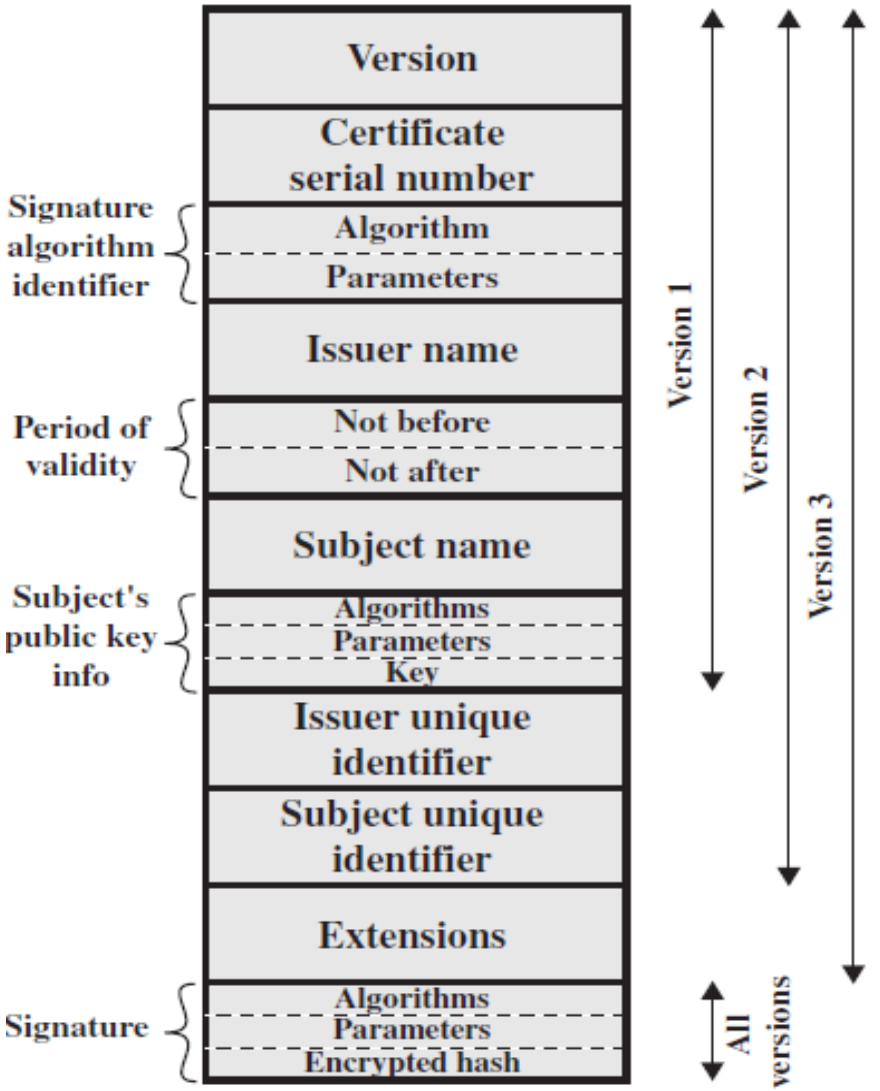


Figure 14.14 Public-Key Certificate Use

Certificate

- Format of certificates
  - Version (v1,v2,v3)
  - Serial number
  - Signature algorithm Identifier
  - Issuer name
  - Period of validity
  - Subject's public-key information
  - Issuer unique identifier
  - Subject unique identifier
  - Extension
  - Signature



(a) X.509 certificate

Figure 14.15 X.509 Formats

Certificates

- Using notation to define a certificate
  - CA<<A>>=CA{V, SN, AI, CA, T_A, A, A_p}
    - Y<<X>> : certificate of user X issued by Y(CA)

- Characteristics
  - Any user with access to the public key of the CA can recover all of the user certificates certified by the CA
  - No one except CA can modify any user's certificate without detected

Certificates

- If all users subscribe to the same CA, they can directly get any certificate using directory of CA.

- But single CA for large number of users is impractical
  - CA must sign all user certificates
  - CA's certificates must be distributed to all users
  - CA must distribute his public-key securely to all users

Obtaining a User's Certificate

- Case of More than one CA
  - Exist some numbers of CAs
  - Distribute its public-key securely to some fraction of users
  - Now we need to get certificate of users that are certified by another CA.

Obtaining a User's Certificate

- How to get other user's certificate
  - Assume user A has the certificate of X1(CA) and B has that of X2(CA)
  - Procedure
    - A gets X2's certificate signed by X1 in directory
      - A can verify X2's certificate by X1's public key
      - Then A can get X2's public key by X2's certificate
    - A gets B's certificate signed by X2 in directory
      - A can verify B's certificate by X2's public key
    - B does the Same operations as A
    - In notation,
      - A : X1<<X2>>X2<<B>>
      - B : X2<<X1>>X1<<A>>

Obtaining a User's Certificate

- For each CA, directory includes the entries for two types of certificates
  - Forward Certificates (By Other CA)
  - Reverse Certificates (By CA)

- Procedure of exchanging certificate between A and B
  - Using certificate chain

A : X<<W>>W<<V>>V<<Y>>Y<<Z>>Z<<B>>

B : Z<<Y>>Y<<V>>V<<W>>W<<X>>X<<A>>

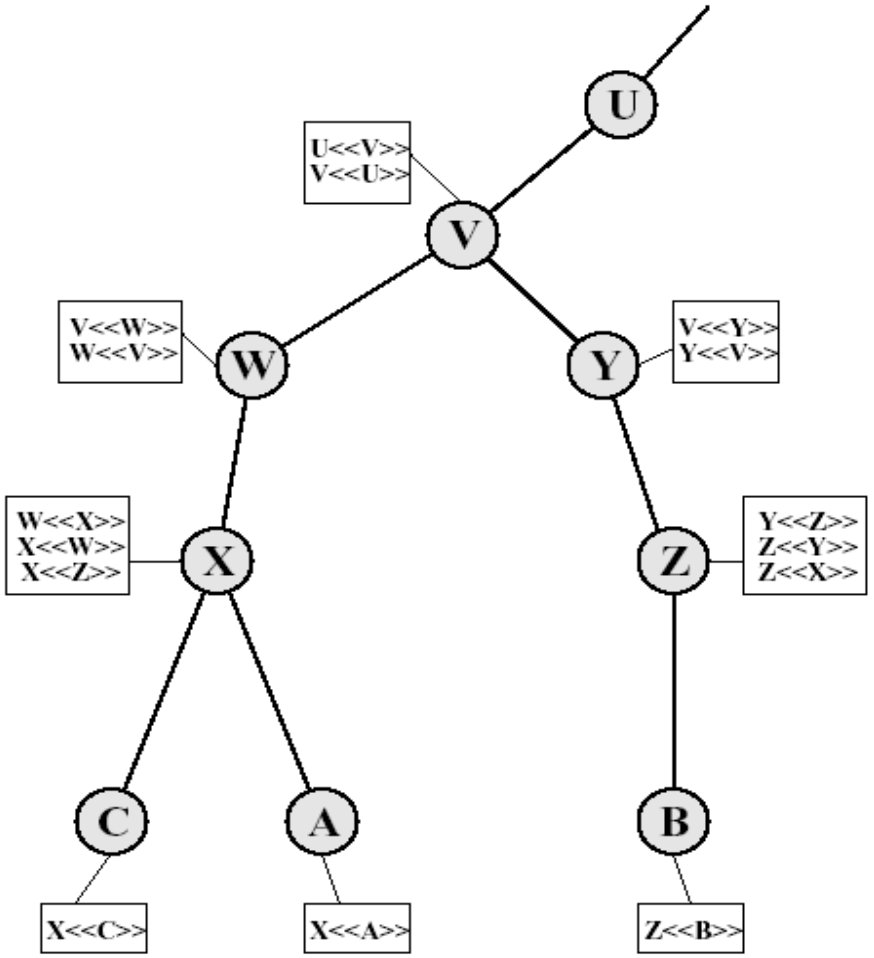
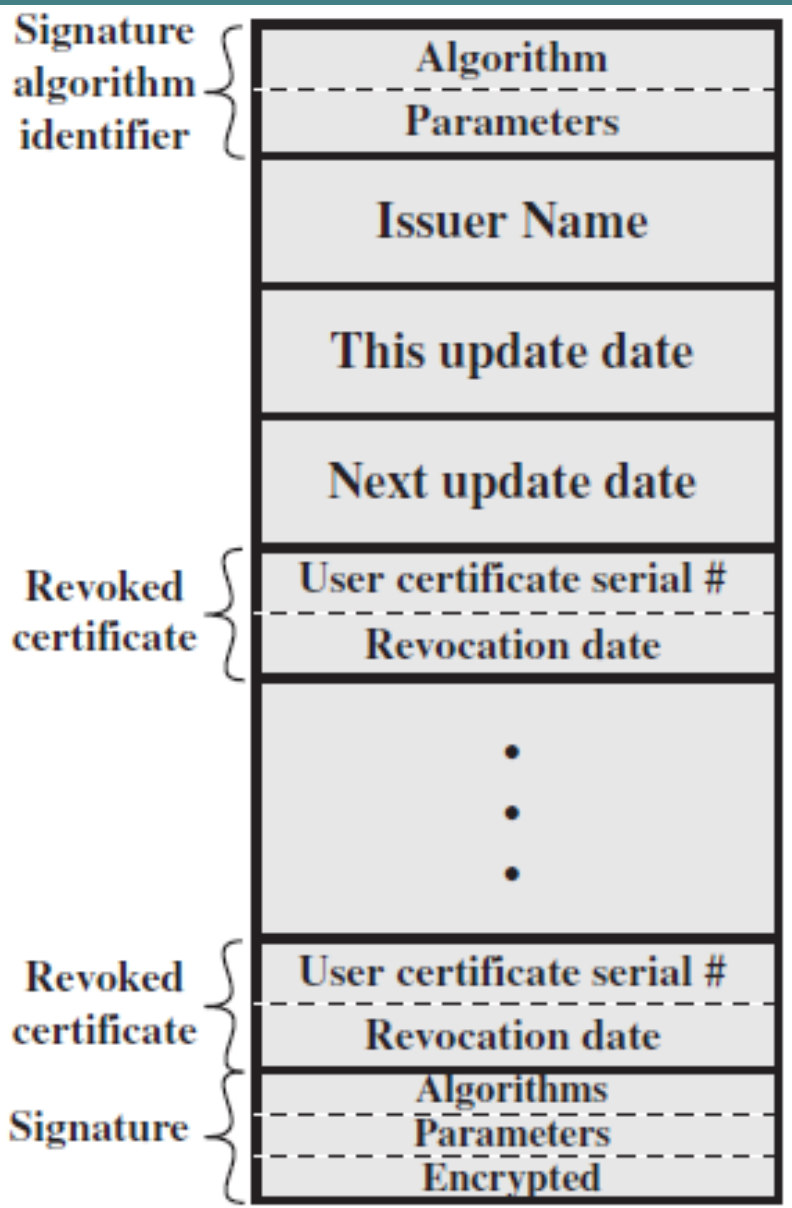


Figure 11.4 X.509 CA Hierarchy: a Hypothetical Example

Revocation of Certificate

- Reason of revoking certificate
  - User's (private) key assumed to be compromised
  - User is no longer certified by the CA
  - CA's certificates assumed to be compromised
- CA must maintain the list of revoked certificates that are not expired
  - Serial number is unique in each certificate
- If user receives a certificate, user must check whether it is revoked



(b) Certificate revocation list

Public-Key Infrastructure

- Public-key infrastructure (PKI) is the set of hardware, software, people, policies, and procedures needed to create, manage, store, distribute, and revoke digital certificates based on asymmetric cryptography.

- Public Key Infrastructure X.509 (PKIX) working group is setting up a formal model based on X.509 that is suitable for deploying a certificate-based architecture on the Internet.

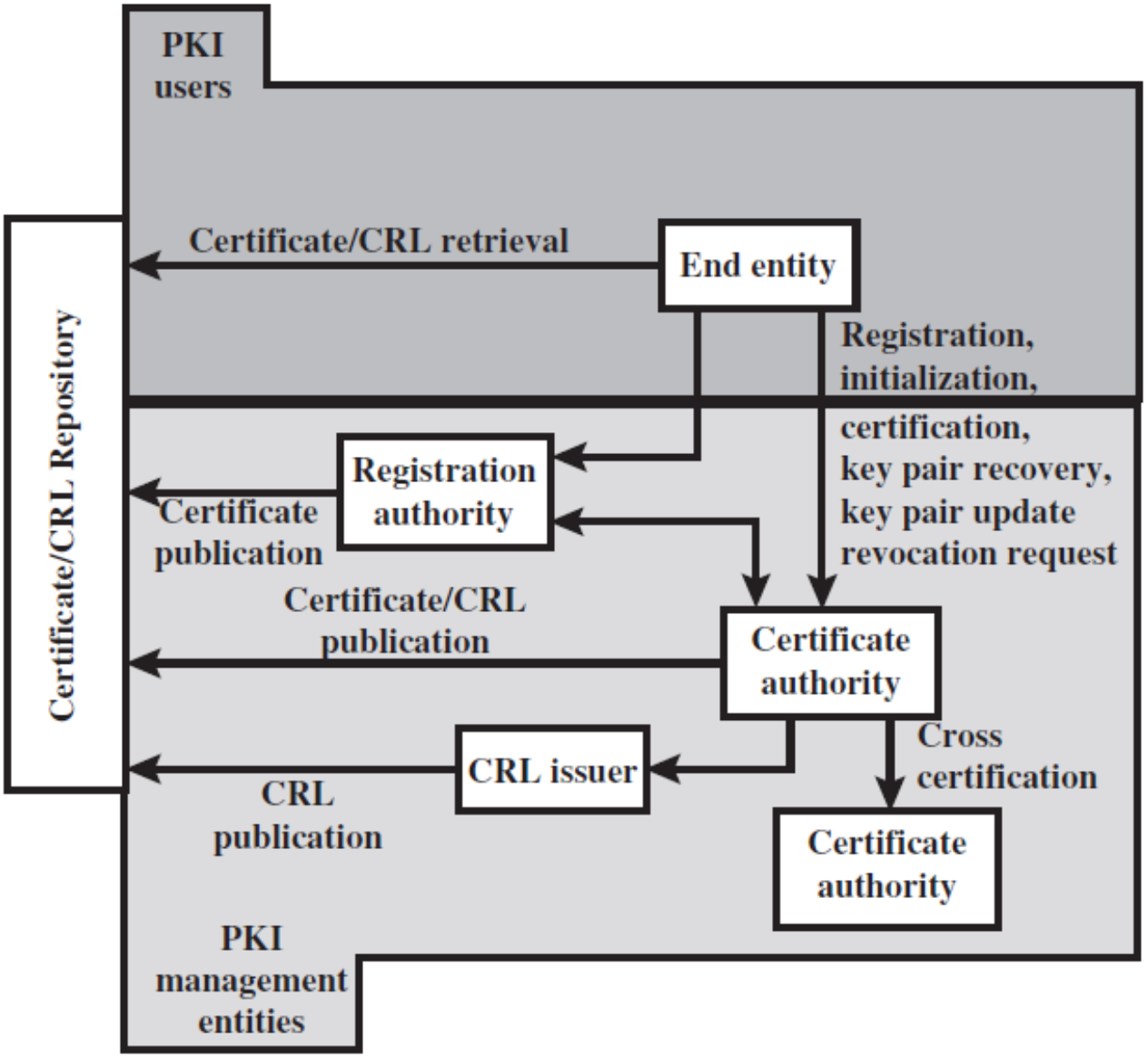


Figure 14.17 PKIX Architectural Model

Public-Key Infrastructure

- **End entity:** denotes end users, devices or any other entity that can be identified in a public-key certificate.
- **Certification authority (CA):** The issuer of certificates and certificate revocation lists (CRLs).
- **Registration authority (RA):** The RA is often associated with the end entity registration process.
- **CRL issuer:** An optional component that a CA can delegate to publish CRLs.
- **Repository:** Any method for storing certificates and CRLs so that they can be retrieved by end entities.

Public-Key Infrastructure

- **PKIX Management Functions**

- **Registration:** This is the process whereby a user first makes itself known to a CA (directly or through an RA), prior to that CA issuing a certificate or certificates for that user. Registration begins the process of enrolling in a PKI.

- **Initialization:** Before a client system can operate securely, it is necessary to install key materials that have the appropriate relationship with keys stored elsewhere in the infrastructure.

- **Certification:** This is the process in which a CA issues a certificate for a user's public key, returns that certificate to the user's client system, and/or posts that certificate in a repository.

Public-Key Infrastructure

- **Key pair recovery:** It is important to provide a mechanism to recover the necessary decryption keys. Key pair recovery allows end entities to restore their encryption/decryption key pair from an authorized key backup facility.

- **Key pair update:** All key pairs need to be updated regularly and new certificates issued. Update is required when the certificate lifetime expires and as a result of certificate revocation.

- **Revocation request:** An authorized person advises a CA of an abnormal situation requiring certificate revocation.

- **Cross certification:** A cross-certificate is a certificate issued by one CA to another CA that contains a CA signature key used for issuing certificates.

END