Key Management and Distribution

Prof. A. A. Daptardar
HIT, Nidasoshi

Symmetric Key Distribution using Symmetric Encryption

- Key Distribution Technique is a term that refers to the means of delivering a key to two parties who wish to exchange data without allowing others to see the key.

- For two parties A and B, key distribution can be achieved in a number of ways, as follows:
  - 1. A can select a key and physically deliver it to B.
  - 2. A third party can select the key and physically deliver it to A and B.
  - 3. If A and B have previously and recently used a key, one party can transmit the new key to the other, encrypted using the old key.
  - 4. If A and B each has an encrypted connection to a third party C, C can deliver a key on the encrypted links to A and B.

- The scale of the problem depends on the number of communicating pairs that must be supported.
- If end-to-end encryption is done at a network or IP level, then a key is needed for each pair of hosts on the network that wish to communicate.
- Thus, if there are *N hosts, the number of required keys is [N(N - 1)]/2.*

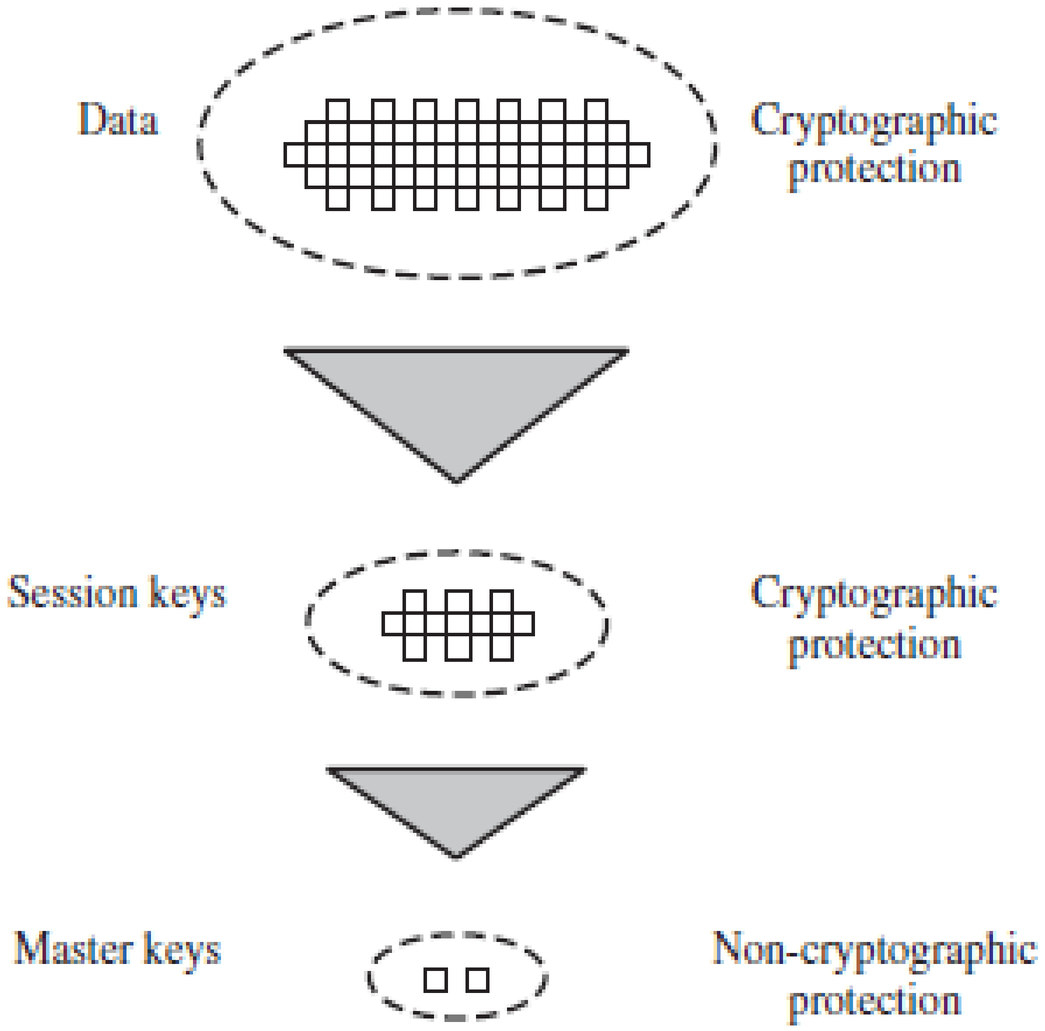
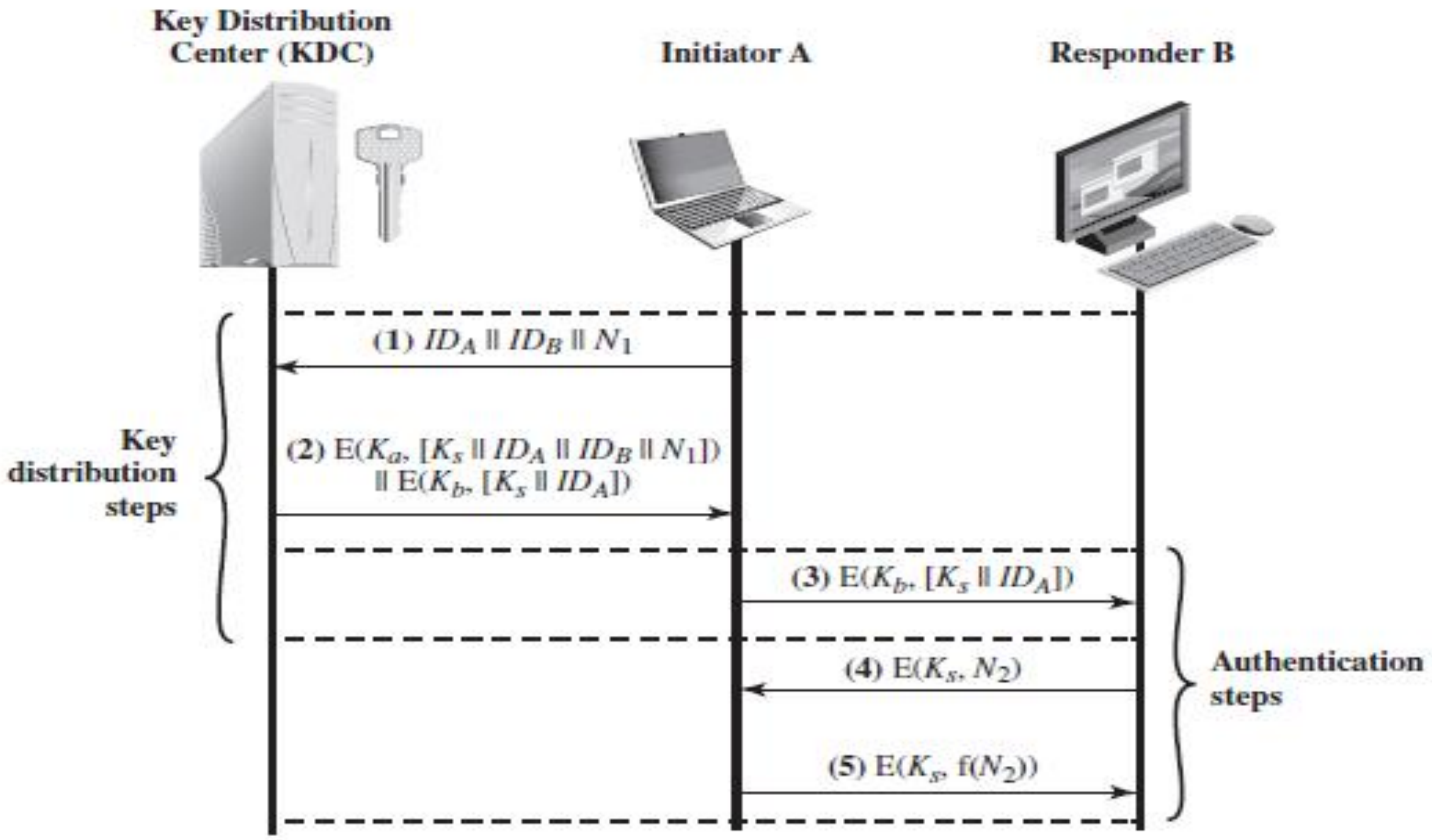


Figure 14.2 The Use of a Key Hierarchy

- For end-to-end encryption, a key distribution center is responsible for distributing keys to pairs of users (hosts, processes, applications) as needed.
- Each user must share a unique key with the key distribution center for purposes of key distribution.
- The use of a key distribution center is based on the use of a hierarchy of keys.
- At a minimum, two levels of keys are used.
  - Communication between end systems is encrypted using a temporary key, often referred to as a **session key.**
  - Typically, the session key is used for the duration of a logical connection, such as a frame relay connection or transport connection, and then discarded.
  - Each session key is obtained from the key distribution center over the same networking facilities used for end-user communication.
  - Accordingly, session keys are transmitted in encrypted form, using a **master key** that is shared by the key distribution center and an end system or user.

A Key Distribution Scenario



Key Distribution Center (KDC)
Initiator A
Responder B
(1) $ID_A \| ID_B \| N_1$
Key distribution steps
(2) $E(K_a, [K_s \| ID_A \| ID_B \| N_1]) \| E(K_b, [K_s \| ID_A])$
(3) $E(K_b, [K_s \| ID_A])$
(4) $E(K_s, N_2)$
Authentication steps
(5) $E(K_s, f(N_2))$

Figure 14.3 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; similarly, B shares the master key Kb with the KDC. The following steps occur.
  - 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, N1, for this transaction, which we refer to as a nonce.
  - 2. The KDC responds with a message encrypted using Ka. The message includes two items intended for A:
    - 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
  - 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), IDA
  - 3. A stores the session key for use in the upcoming session and forwards to B the information that originated at the KDC for B, namely, E(Kb,[Ks || IDA]).
  - 4. Using the newly minted session key for encryption, B sends a nonce, N2, to A.
  - 5. Also, using Ks, A responds with f(N2), where f is a function that performs some transformation on N2 (e.g., adding one).

Hierarchical Key Control

- It is not necessary to limit the key distribution function to a single KDC.
- A hierarchy of KDCs can be established.
- For example, there can be local KDCs, each responsible for a small domain of the overall internetwork, such as a single LAN or a single building.
- For communication among entities within the same local domain, the local KDC is responsible for key distribution.

- If two entities in different domains desire a shared key, then the corresponding local KDCs can communicate through a global KDC.
- In this case, any one of the three KDCs involved can actually select the key.
- A hierarchical scheme minimizes the effort involved in master key distribution, because most master keys are those shared by a local KDC with its local entities.
- Furthermore, such a scheme limits the damage of a faulty or subverted KDC to its local area only.

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.
- A security manager must try to balance these competing considerations in determining the lifetime of a particular session key.

- 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, perhaps every time the PDU (protocol data unit) sequence number cycles.
- For a connectionless protocol, such as a transaction-oriented protocol, there is no explicit connection initiation or termination.
- Thus, it is not obvious how often one needs to change the session key.
- The most secure approach is to use a new session key for each exchange.
- 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 scheme (Figure 14.4) is useful for providing end-to- end encryption at a network or transport level in a way that is transparent to the end users.
- The approach assumes that communication makes use of a connection-oriented end-to-end protocol, such as TCP.
- The noteworthy element of this approach is a session security module (SSM), which may consist of functionality at one protocol layer, that performs end-to-end encryption and obtains session keys on behalf of its host or terminal.

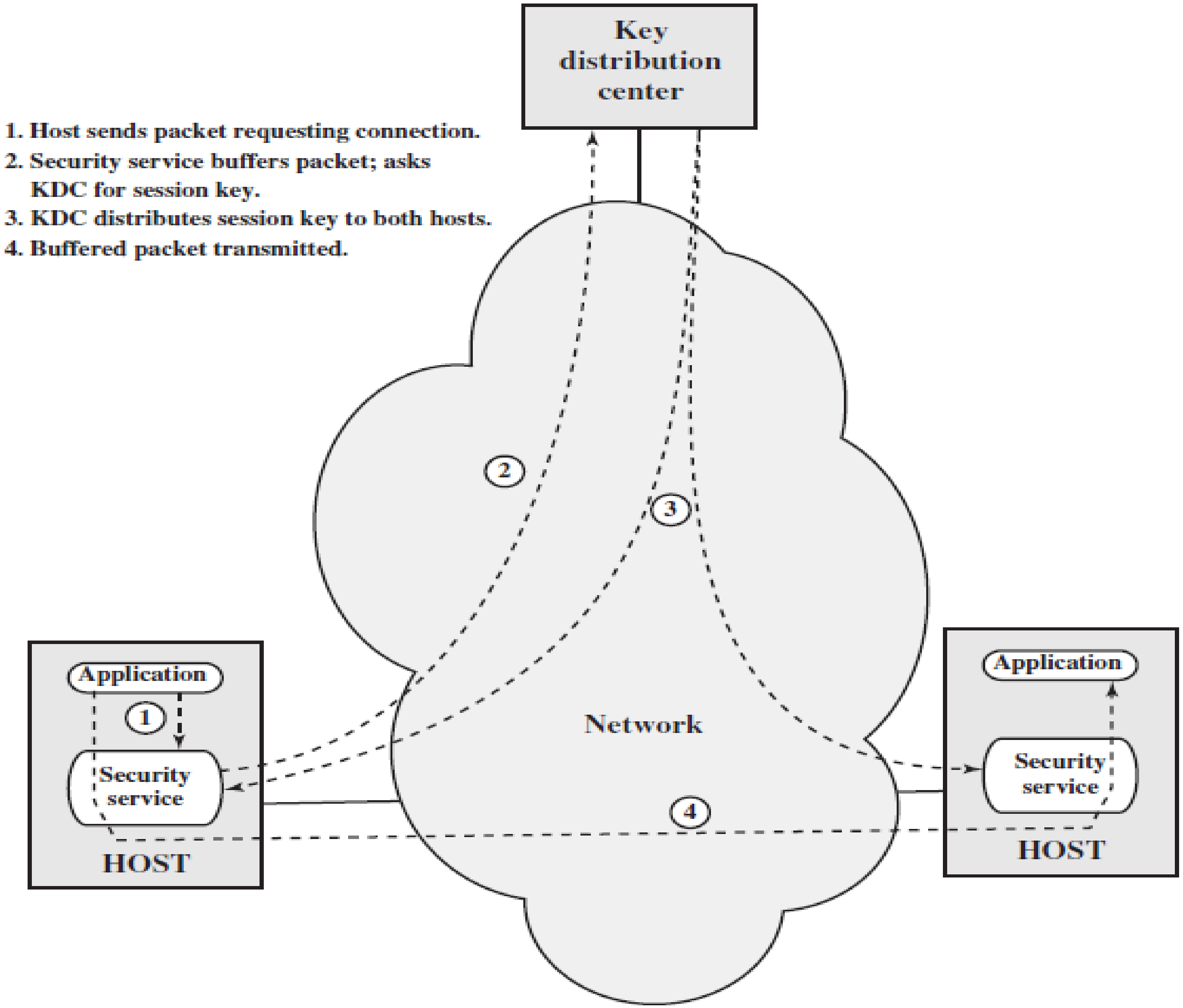

Key distribution center
1. Host sends packet requesting connection.
2. Security service buffers packet; asks KDC for session key.
3. KDC distributes session key to both hosts.
4. Buffered packet transmitted.
2
3
Application
1
Security service
HOST
Network
4
Application
Security service
HOST

Figure 14.4 Automatic Key Distribution for Connection-Oriented Protocol

- 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.
- 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.

Decentralized Key Control

- The use of a key distribution center imposes the requirement that the KDC be trusted and be protected from subversion.
- This requirement can be avoided if key distribution is fully decentralized.
- A decentralized approach requires that each end system be able to communicate in a secure manner with all potential partner end systems for purposes of session key distribution.
- Thus, there may need to be as many as [n(n - 1)]/2 master keys for a configuration with n end systems.

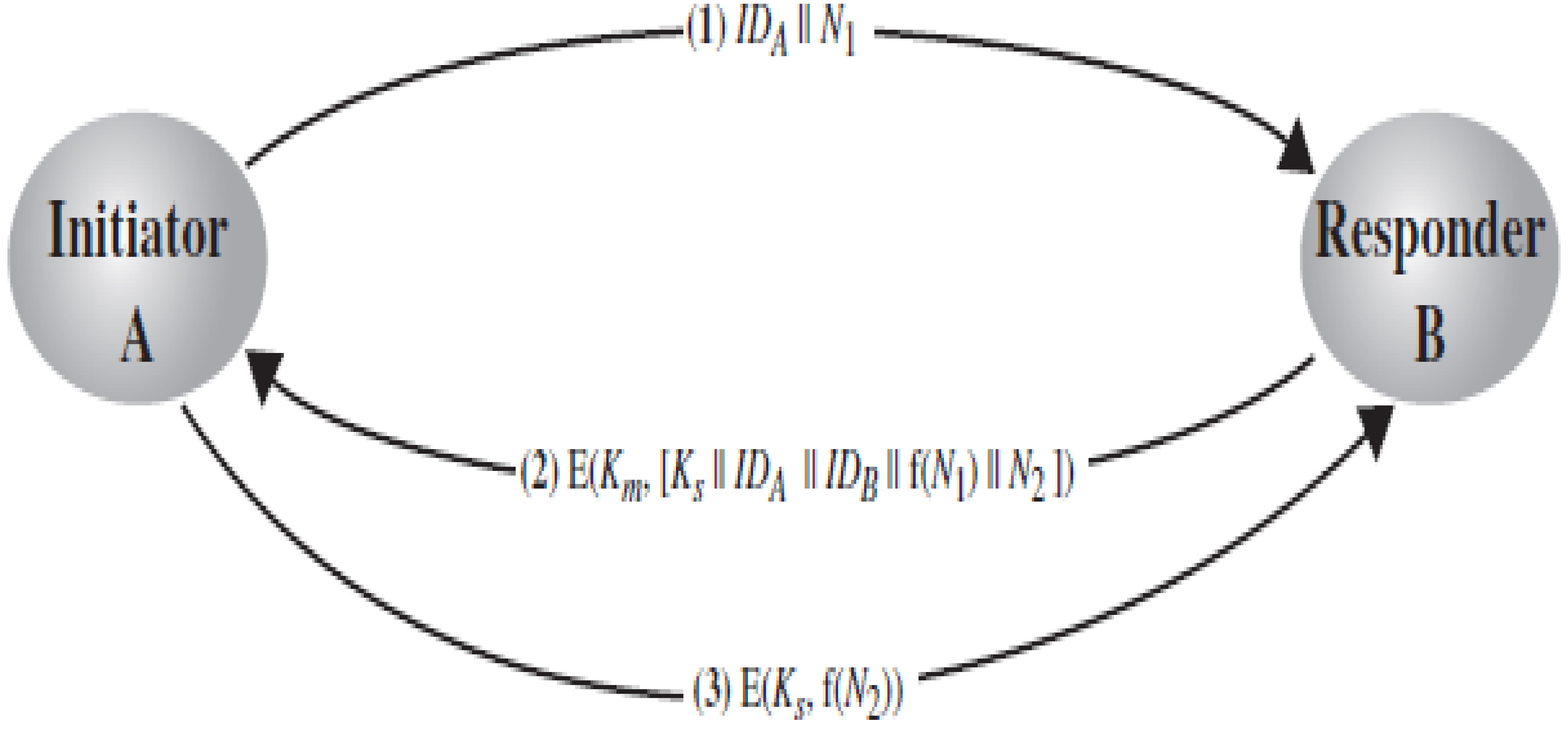


Figure 14.5 Decentralized Key Distribution

- A session key may be established with the following sequence of steps (Figure 14.5).
  - 1. A issues a request to B for a session key and includes a nonce, N1.
  - 2. B responds with a message that is encrypted using the shared master key. The response includes the session key selected by B, an identifier of B, the value f(N1), and another nonce, N2.
  - 3. Using the new session key, A returns f(N2) to B.

Controlling Key Usage

- The concept of a key hierarchy and the use of automated key distribution techniques greatly reduce the number of keys that must be manually managed and distributed.
- It also may be desirable to impose some control on the way in which automatically distributed keys are used.
- For example, in addition to separating master keys from session keys, we may wish to define different types of session keys on the basis of use, such as
  – Data-encrypting key, for general communication across a network
  – PIN-encrypting key, for personal identification numbers (PINs) used in electronic funds transfer and point-of-sale applications
  – File-encrypting key, for encrypting files stored in publicly accessible locations

- The Proposed technique is for use with DES and makes use of the extra 8 bits in each 64-bit DES key. That is, the eight non-key bits ordinarily reserved for parity checking form the key tag.
- 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 spares for future use.
- 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.

- A more flexible scheme, referred to as the control vector.
- In this scheme, 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.

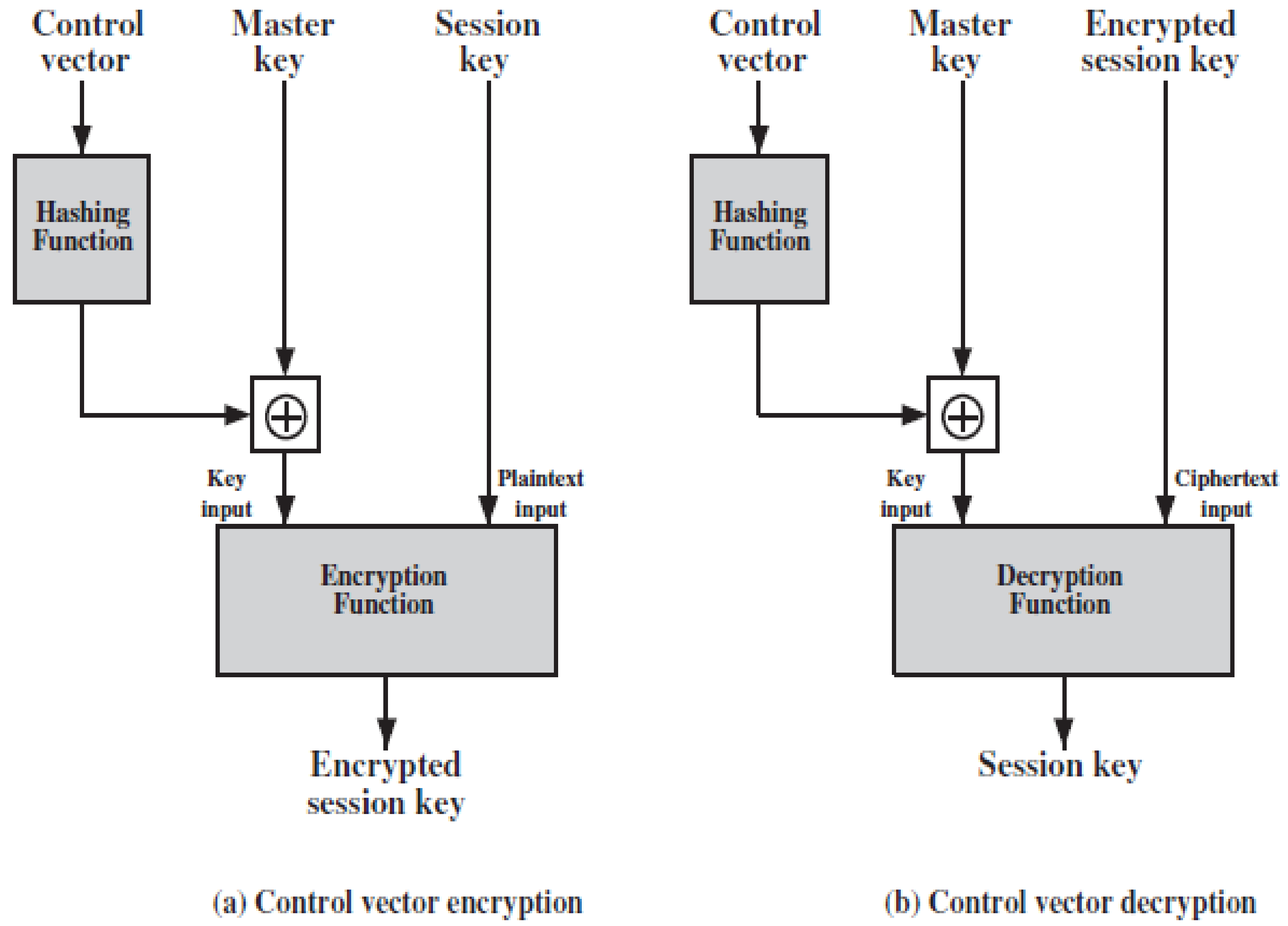


Figure 14.6 Control Vector Encryption and Decryption

- As a first step, the control vector is passed through a hash function that produces a value whose length is equal to the encryption key length.
- 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 = $K_m \oplus H$
  - Ciphertext = $E([K_m \oplus H], K_s)$
- where K_m is the master key and K_s is the session key.
- The session key is recovered in plaintext by the reverse operation:
- $D([K_m \oplus H], E([K_m \oplus H], K_s))$

- Use of the control vector has two advantages over use of an 8-bit tag.
  - First, there is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use.
  - Second, 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

- Because of the inefficiency of public-key cryptosystems, they are almost never used for the direct encryption of sizable block of data, but are limited to relatively small blocks.
- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution.

Simple Secret Key Distribution

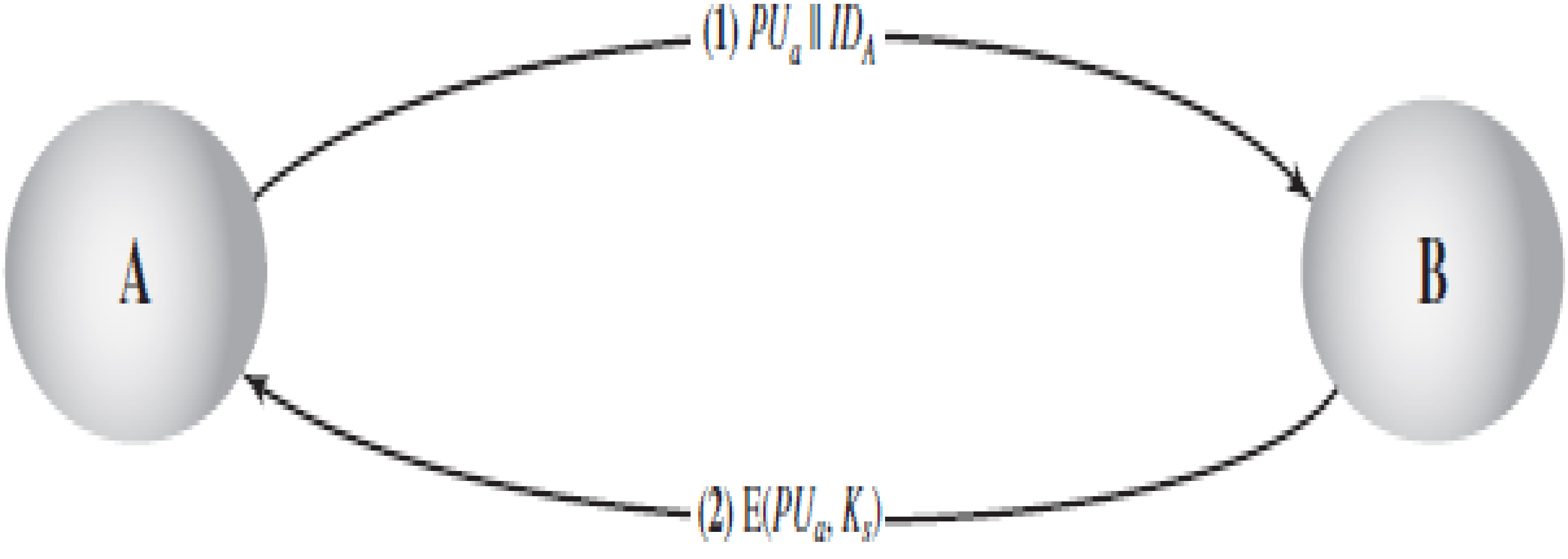


Figure 14.7 Simple Use of Public-Key Encryption to Establish a Session Key

- If A wishes to communicate with B, the following procedure is employed:
- 1. A generates a public/private key pair $\{PU_a, PR_a\}$ and transmits a message to B consisting of PU_a and an identifier of A, ID_A.
- 2. B generates a secret key, K_s, and transmits it to A, which is encrypted with A's public key.
- 3. A computes $D(PR_a, E(PU_a, K_s))$ to recover the secret key. Because only A can decrypt the message, only A and B will know the identity of K_s.
- 4. A discards PU_a and PR_a and B discards PU_a.

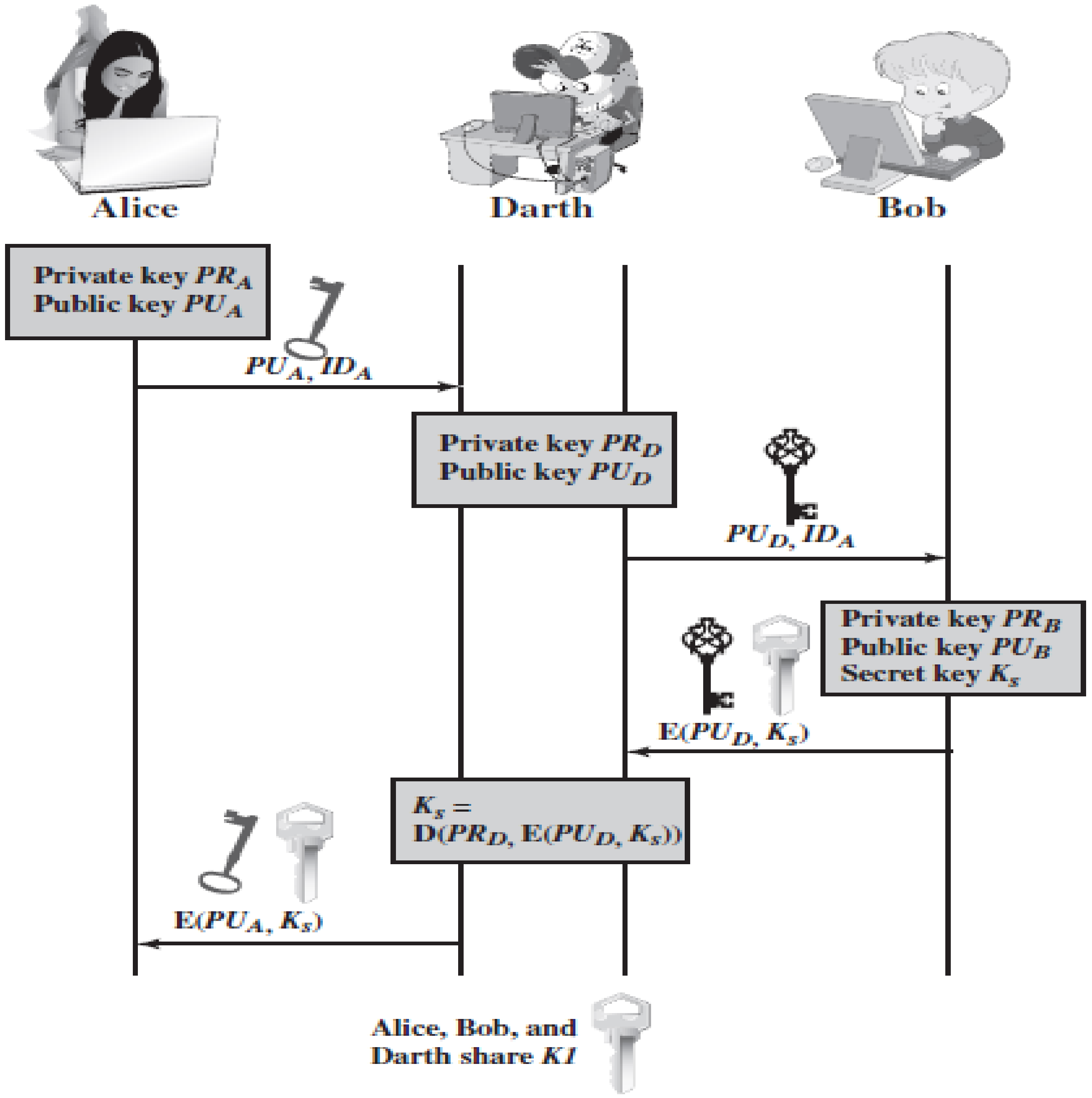


Figure 14.8 Another Man-in-the-Middle Attack

- In the present case, if an adversary, D, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected (Figure 14.8).
- 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. D intercepts the message, creates its own public/private key pair $\{PU_d, PR_d\}$ and transmits $PU_s||$ ID_A to B.
- 3. B generates a secret key, K_s, and transmits $E(PU_s, K_s)$.
- 4. D intercepts the message and learns K_s by computing $D(PR_d, E(PU_d, K_s))$.
- 5. D transmits $E(PU_a, K_s)$ to A.

Secret Key Distribution with Confidentiality and Authentication

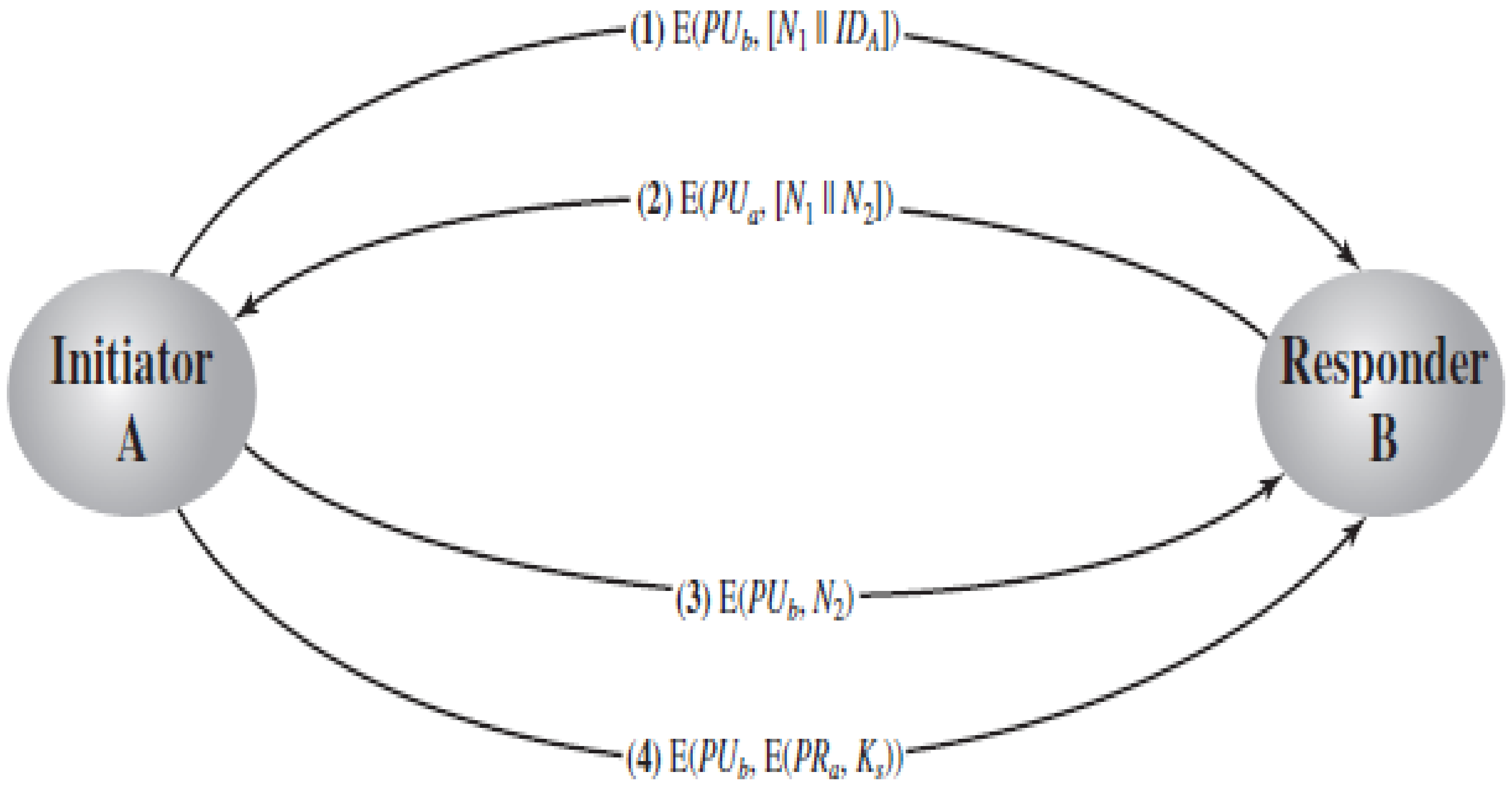

(1) $E(PU_b, [N_1 \| ID_A])$
(2) $E(PU_a, [N_1 \| N_2])$
Initiator A
Responder B
(3) $E(PU_b, N_2)$
(4) $E(PU_b, E(PR_a, K_s))$

Figure 14.9 Public-Key Distribution of Secret Keys

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

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. 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.

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 Keys

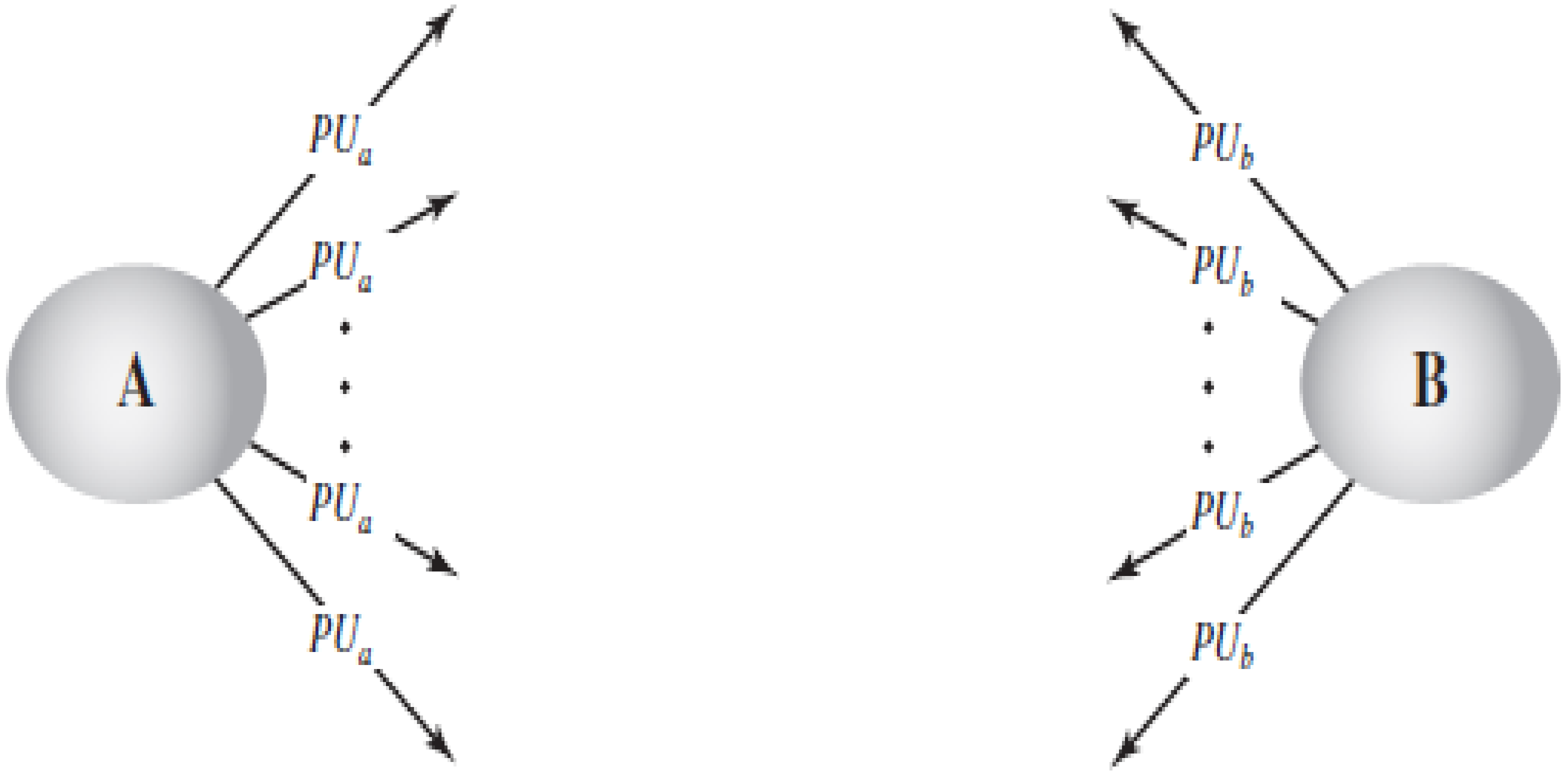


Figure 14.10 Uncontrolled Public-Key Distribution

- 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 (Figure 14.10).
- For example, because of the growing popularity of PGP (pretty good privacy) which makes use of RSA, many PGP users have adopted the practice of appending their public key to messages that they send to public forums, such as USENET newsgroups and Internet mailing lists.

- 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

Publicly Available Directory

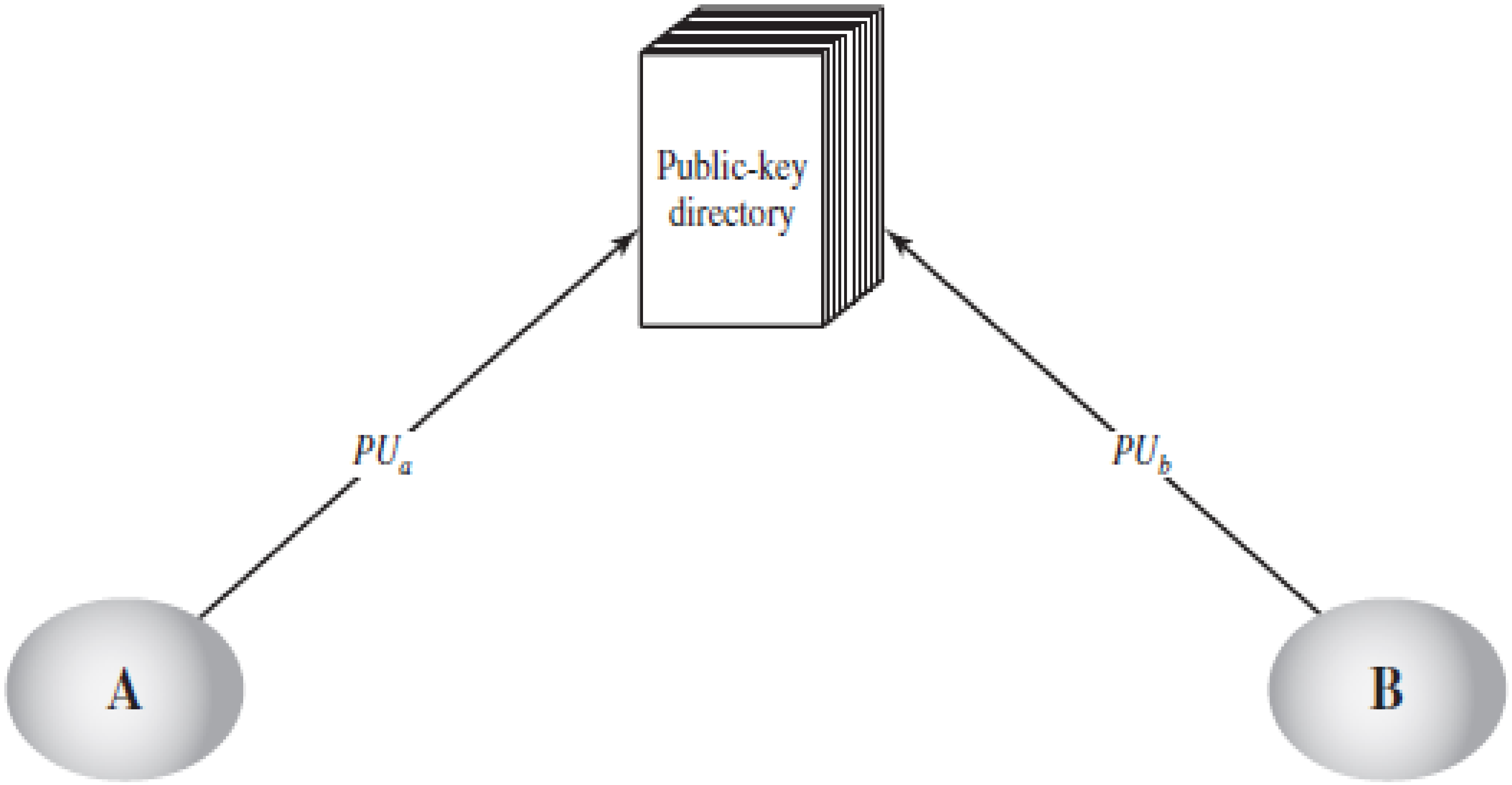


Figure 14.11 Public-Key Publication

- 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 (Figure 14.11). 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.

- 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.
- Another way to achieve the same end is for the adversary to tamper with the records kept by the authority.

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.
- A typical scenario is illustrated in Figure 14.12

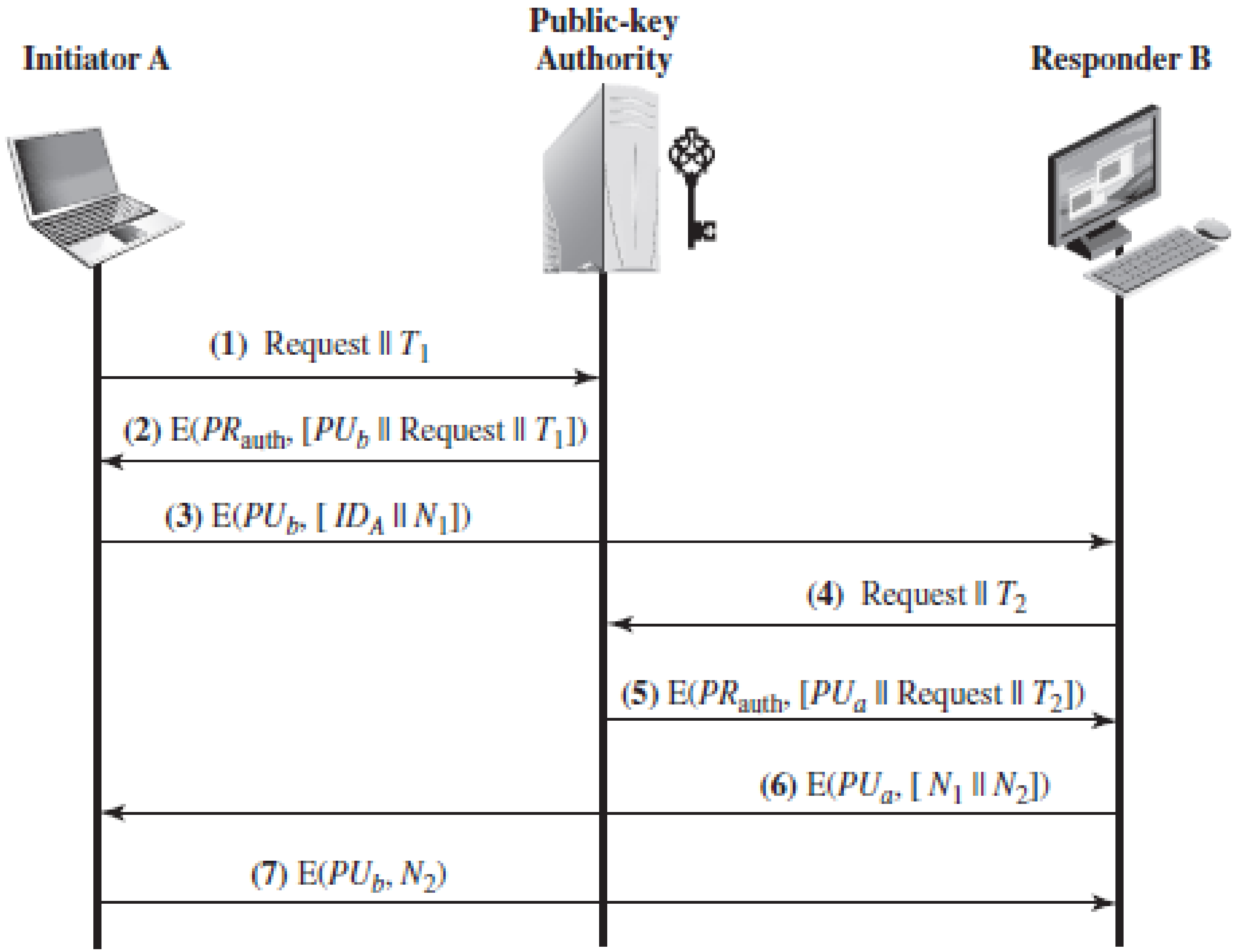

Initiator A
Public-key Authority
Responder B
(1) Request ‖ T_1
(2) E(PR_{auth}, [PU_b ‖ Request ‖ T_1])
(3) E(PU_b, [ID_A ‖ N_1])
(4) Request ‖ T_2
(5) E(PR_{auth}, [PU_a ‖ Request ‖ T_2])
(6) E(PU_a, [N_1 ‖ N_2])
(7) E(PU_b, N_2)

Figure 14.12 Public-Key Distribution Scenario

- The scenario assumes that a central authority maintains a dynamic directory of public keys of all participants. In addition, 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:
- B’s public key, *PUb, which A can use to encrypt messages destined for B*
- The original request used 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
- The original timestamp given 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.

- At this point, public keys have been securely delivered to A and B, and they may begin their protected exchange. However, two additional steps are desirable:
- 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). Because 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, which is encrypted using B’s public key, to assure B that its correspondent is A.

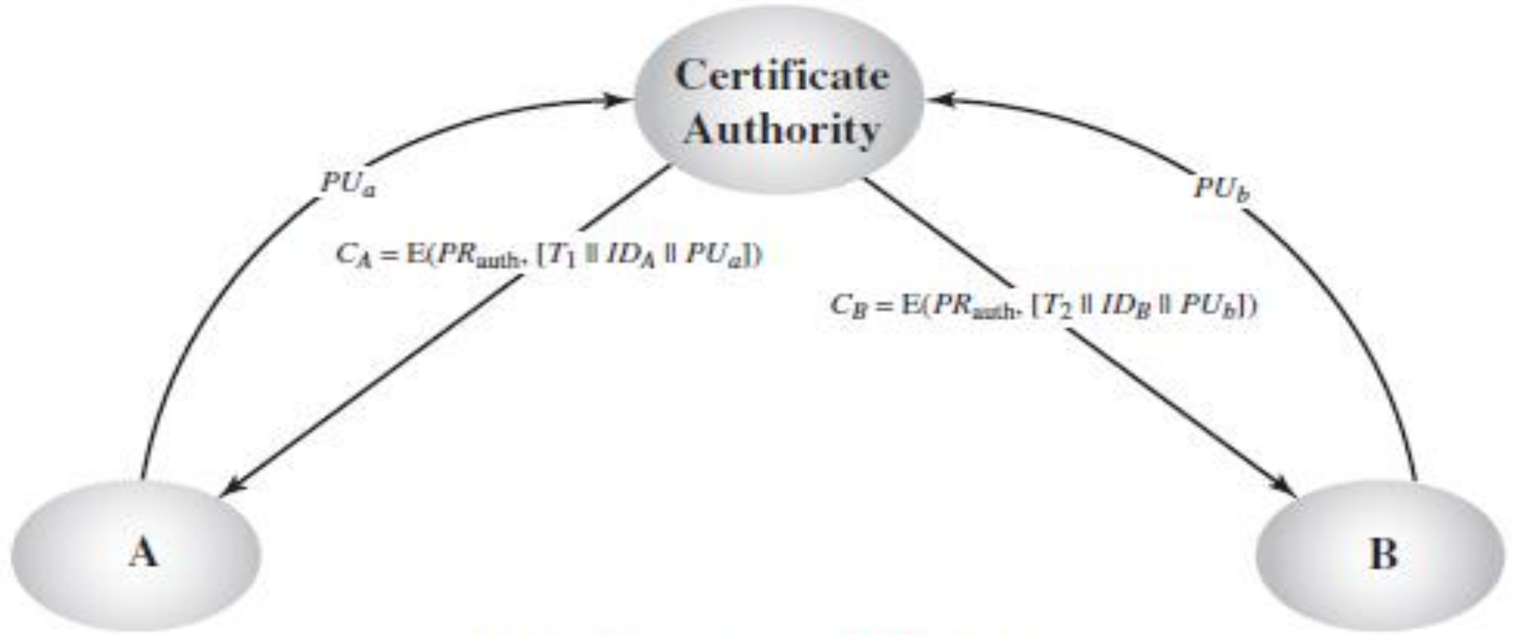
Drawbacks

- 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.

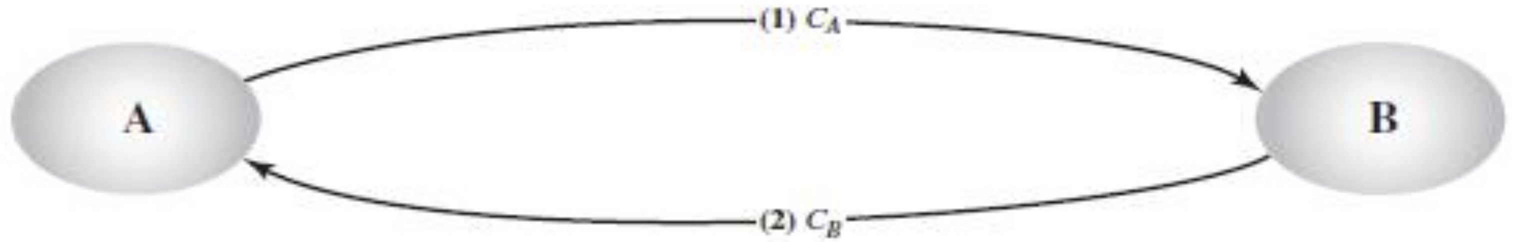
Public-Key Certificates

- An alternative approach, first suggested by Kohnfelder is to use certificates that can be used by participants to exchange keys without contacting a public-key authority, in a way that is as reliable as if the keys were obtained directly from a public-key authority.
- In essence, a certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.
- Typically, 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.

- 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.
- 4. Any participant can verify the currency of the certificate.


Certificate Authority
PU_a
PU_b
$C_A = E(PR_{auth}, [T_1 \| ID_A \| PU_a])$
$C_B = E(PR_{auth}, [T_2 \| ID_B \| PU_b])$
A
B

(a) Obtaining certificates from CA



(b) Exchanging certificates

Figure 14.13 Exchange of Public-Key Certificates

- Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
- Application must be in person or by some form of secure authenticated communication.
- For participant A, the authority provides a certificate of the form

  CA = E(PRauth, [T||IDA ||PUa])
- where *PRauth is the private key used by the authority and T is a timestamp.*
- *A may* then pass this certificate on to any other participant, who reads and verifies the certificate as follows:

  D(*PUauth, CA) = D(PUauth, E(PRauth, [T||IDA ||PUa])) = (T||IDA }PUa)*

- The recipient uses the authority’s public key, PUauth, to decrypt the certificate.
- Because the certificate is readable only using the authority’s public key, this verifies that the certificate came from the certificate authority.
- The elements IDA and PUa provide the recipient with the name and public key of the certificate’s holder.
- The timestamp T validates the currency of the certificate.

Elliptic Curve Arithmetic

Elliptic Curve Arithmetic

- Most of the products and standards that use public-key cryptography for encryption and digital signatures use RSA.
- The key length for secure RSA use has increased over recent years, and this has put a heavier processing load on applications using RSA.
- This burden has ramifications, especially for electronic commerce sites that conduct large numbers of secure transactions.
- A competing system challenges RSA: elliptic curve cryptography (ECC).

- ECC is showing up in standardization efforts, including the IEEE P1363 Standard for Public-Key Cryptography.
- The principal attraction of ECC, compared to RSA, is that it appears to offer equal security for a far smaller key size, thereby reducing processing overhead.

Abelian Groups

- An abelian group *G, sometimes denoted by {G, * }, is* a set of elements with a binary operation, denoted by * , that associates to each ordered pair (*a, b) of elements in G an element (a * b) in G, such that the following* axioms are obeyed:3
  - (A1) Closure: If *a and b belong to G, then a * b is also in G.*
  - (A2) Associative: *a * (b * c) = (a * b) * c for all a, b, c in G.*
  - (A3) Identity element: There is an element e in *G such that a * e = e * a = a* for all *a in G.*
  - (A4) Inverse element: For each *a in G there is an element a′ in G such that a * a′ = a′ * a = e.*
  - (A5) Commutative: *a * b = b * a for all a, b in G.*

- For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition. For example,
- *a * k = (a + a + ...+ a)*

  k times
- where the addition is performed over an elliptic curve.
- Cryptanalysis involves determining *k given a and (a * k).*
- An elliptic curve is defined by an equation in two variables with coefficients.