ATME College of Engineering

Module-3

Elliptic Curve Arithmetic

Elliptic Curve Arithmetic

- Majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- Imposes a significant load in storing and processing keys and messages
- An alternative is to use elliptic curves
- Offers same security with smaller bit sizes

Abelian Groups

- Abelian group G, 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:

(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.

- The operator • is generic and can refer to addition, multiplication, or some other mathematical operation.

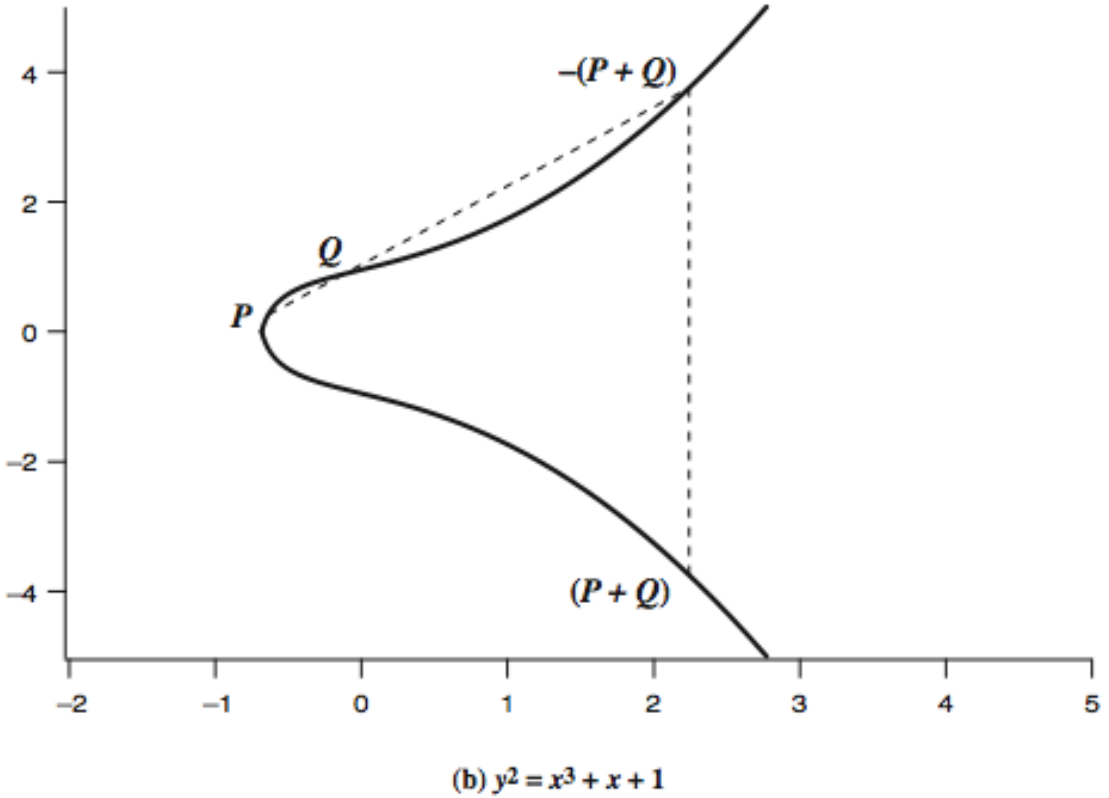
- A number of public-key ciphers are based on the use of an abelian group.
- For example, Diffie-Hellman key exchange involves multiplying pairs of nonzero integers modulo a prime number q.
- Keys are generated by exponentiation over the group, with exponentiation defined as repeated multiplication.
- For example, $a^k \bmod q = \underbrace{(a \times a \times \ldots \times a)}_{k \text{ times}} \bmod q.$
- To attack Diffie-Hellman, the attacker must determine k given a and a^k; this is the discrete log problem.
- For elliptic curve cryptography, an operation over elliptic curves, called addition, is used. Multiplication is defined by repeated addition.
- For example, $a \times k = \underbrace{(a + a + \ldots + a)}_{k \text{ times}}$ where the addition is performed over an elliptic curve.
- Cryptanalysis involves determining k given a and (a x k).

Elliptic Curves over Real Numbers

- an elliptic curve is defined by an equation in two variables x & y, with coefficients
- consider a cubic elliptic curve of form

$$y^2 = x^3 + ax + b$$

where x, y, a, b are all real numbers



(b) $y^2 = x^3 + x + 1$

Elliptic Curves over Z_p

- Elliptic curve cryptography uses curves whose variables & coefficients are finite
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - Use a cubic equation in which variables and coefficients take on values in the set of integers from 0 through p-1 and calculations are performed modulo p
    - best for software applications
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best for hardware applications

Elliptic Curve Cryptography

- ECC addition is analog of modulo multiply
- ECC repeated addition is analog of modulo exponentiation
- need “hard” problem equiv to discrete log
  - Q=kP, where Q,P belong to a prime curve
  - is “easy” to compute Q given k,P
  - but “hard” to find k given Q,P
  - known as the elliptic curve logarithm problem

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_q(a,b)$
- select base point $G=(x_1,y_1)$
  - with large order n s.t. $nG=O$
- A & B select private keys $n_A<n$, $n_B<n$
- compute public keys: $P_A=n_AG$, $P_B=n_BG$
- compute shared key: $k=n_AP_B$, $k=n_BP_A$
  - same since $K=n_An_BG$
- attacker would need to find k, hard

Global Public Elements

$E_q(a, b)$	elliptic curve with parameters a, b, and q, where q is a prime or an integer of the form 2^m
G	point on elliptic curve whose order is large value n

User A Key Generation

Select private n_A	$n_A < n$
Calculate public P_A	$P_A = n_A \times G$

User B Key Generation

Select private n_B	$n_B < n$
Calculate public P_B	$P_B = n_B \times G$

Calculation of Secret Key by User A

$K = n_A \times P_B$

Calculation of Secret Key by User B

$K = n_B \times P_A$

ECC Encryption/Decryption

- several alternatives, will consider simplest
- must first encode any message M as a point on the elliptic curve P_m
- select suitable curve & point G as in D-H
- each user chooses private key $n_A<n$
- and computes public key $P_A=n_AG$
- to encrypt P_m : $C_m=\{kG, P_m+kP_b\}$, k random
- decrypt C_m compute:
  - $P_m+kP_b-n_B(kG) = P_m+k(n_BG)-n_B(kG) = P_m$

Security of Elliptic Curve Cryptography

- depends on how difficult it is to determine *k* given *kP* and *P*. This is referred to as the elliptic curve logarithm problem
- The fastest known technique for taking the elliptic curve logarithm is known as the Pollard rho method.
- Considerably smaller key size can be used for ECC compared to RSA.
- Thus, there is a computational advantage to using ECC with a shorter key length than a comparably secure RSA.

Symmetric Key Algorithms	Diffie-Hellman, Digital Signature Algorithm	RSA (size of n in bits)	ECC (modulus size in bits)
80	$L = 1024$ $N = 160$	1024	160–223
112	$L = 2048$ $N = 224$	2048	224–255
128	$L = 3072$ $N = 256$	3072	256–383
192	$L = 7680$ $N = 384$	7680	384–511
256	$L = 15{,}360$ $N = 512$	15,360	512+

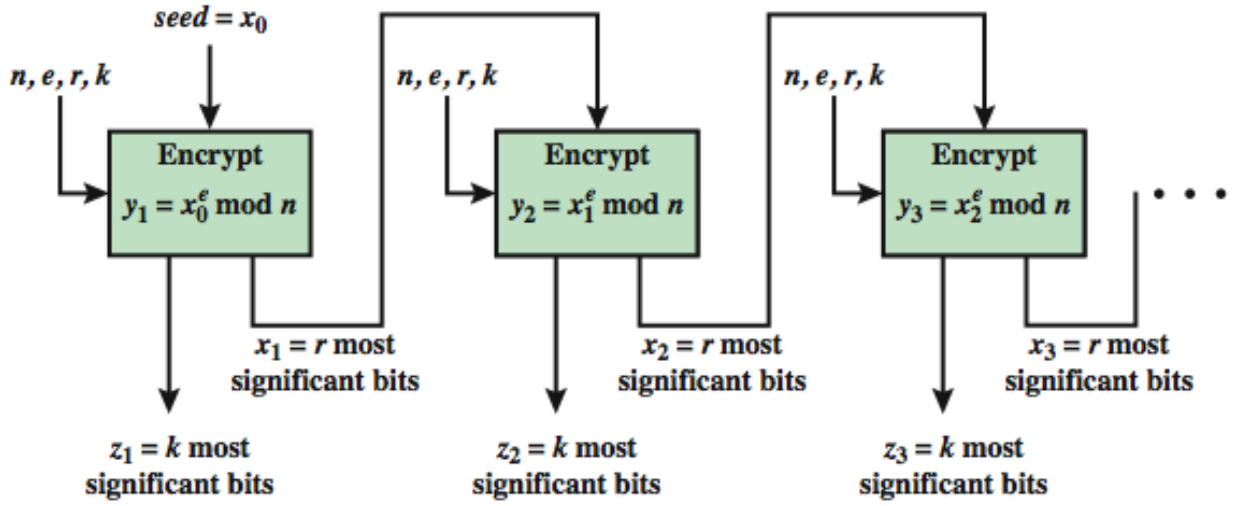
Note: L = size of public key, N = size of private key

Pseudorandom Number Generation (PRNG) based on Asymmetric Ciphers

- asymmetric encryption algorithm produce apparently random output
- hence can be used to build a pseudorandom number generator (PRNG)
- much slower than symmetric algorithms
- hence only use to generate a short pseudorandom bit sequence (eg. key)

PRNG Based on RSA

- For a sufficient key length, the RSA algorithm is considered secure and is a good
- candidate to form the basis of a PRNG. Such a PRNG, known as the Micali-Schnorr PRNG



Setup Select p, q primes; $n = pq; \phi(n) = (p - 1)(q - 1)$. Select e such that $\gcd(e, \phi(n)) = 1$. These are the standard RSA setup selections (see Figure 9.5). In addition, let $N = [\log_2 n] + 1$ (the bitlength of n). Select r, k such that $r + k = N$.

Seed Select a random seed x_0 of bitlength r.

Generate Generate a pseudorandom sequence of length $k \times m$ using the loop

for i **from** 1 to m **do**

$$y_i = x_{i-1}^e \bmod n$$
$$x_i = r \text{ most significant bits of } y_i$$
$$z_i = k \text{ least significant bits of } y_i$$

Output The output sequence is $z_1 \| z_2 \| \ldots \| z_m$.

The parameters n, r, e, and k are selected to satisfy the following six requirements.

1. $n = pq$	n is chosen as the product of two primes to have the cryptographic strength required of RSA.
2. $1 < e < \phi(n)$; $\gcd(e, \phi(n)) = 1$	Ensures that the mapping $s \rightarrow s^e \bmod n$ is 1 to 1.
3. $re \geq 2N$	Ensures that the exponentiation requires a full modular reduction.
4. $r \geq 2\,strength$	Protects against a cryptographic attacks.
5. k, r are multiples of 8	An implementation convenience.
6. $k \geq 8; r + k = N$	All bits are used.

PRNG based on ECC

- dual elliptic curve PRNG
  - NIST SP 800-9, ANSI X9.82 and ISO 18031
- some controversy on security /inefficiency
- algorithm

  for i = 1 to k do

  set $s_i = x(s_{i-1} P)$

  set $r_i = lsb_{240}(x(s_i Q))$

  end for

  return $r_1, \ldots, r_k$

- only use if just have ECC

Key Management and Distribution

Topics covered

- Symmetric Key Distribution Using Symmetric Encryption
- Symmetric Key Distribution Using Asymmetric Encryption
- Distribution of Public Keys

Symmetric Key Distribution Using Symmetric Encryption

- For symmetric encryption to work, the two parties to an exchange 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.
- The strength of any cryptographic system rests with the *key distribution technique*

- 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 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

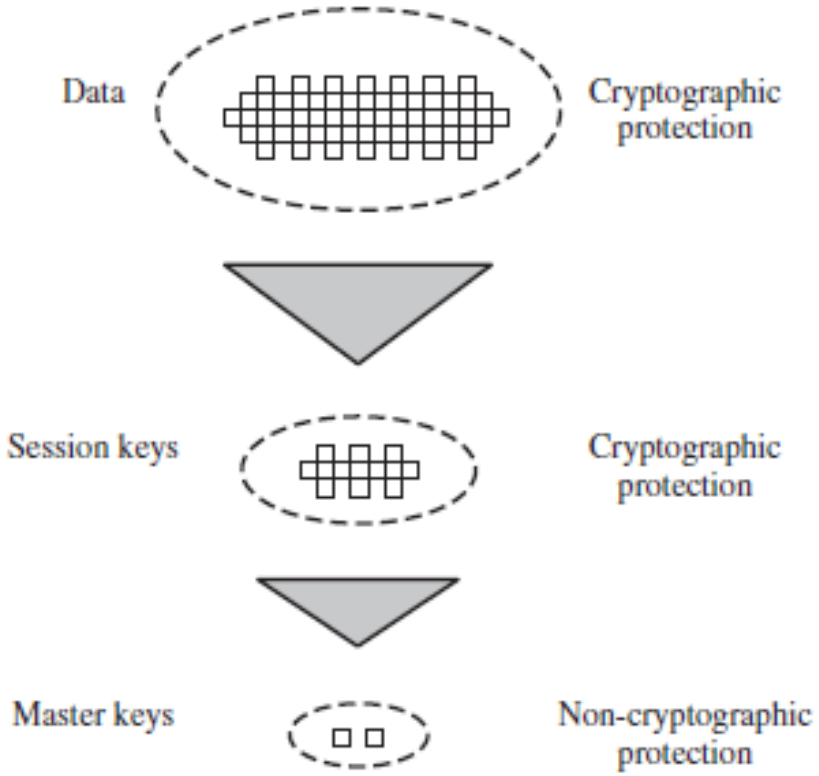


Fig: The Use of a Key Hierarchy

- Communication between end systems is encrypted using a temporary key, referred to as a **session key**.
  - session key is used for the duration of a logical connection
  - 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.
- For each end system or user, there is a unique master key that it shares with the key distribution center
- If there are *N* entities that wish to communicate in pairs, then, only *N* master keys are required, one for each entity.
- Thus, master keys can be distributed in some non-cryptographic way, such as physical delivery.

A Key Distribution Scenario

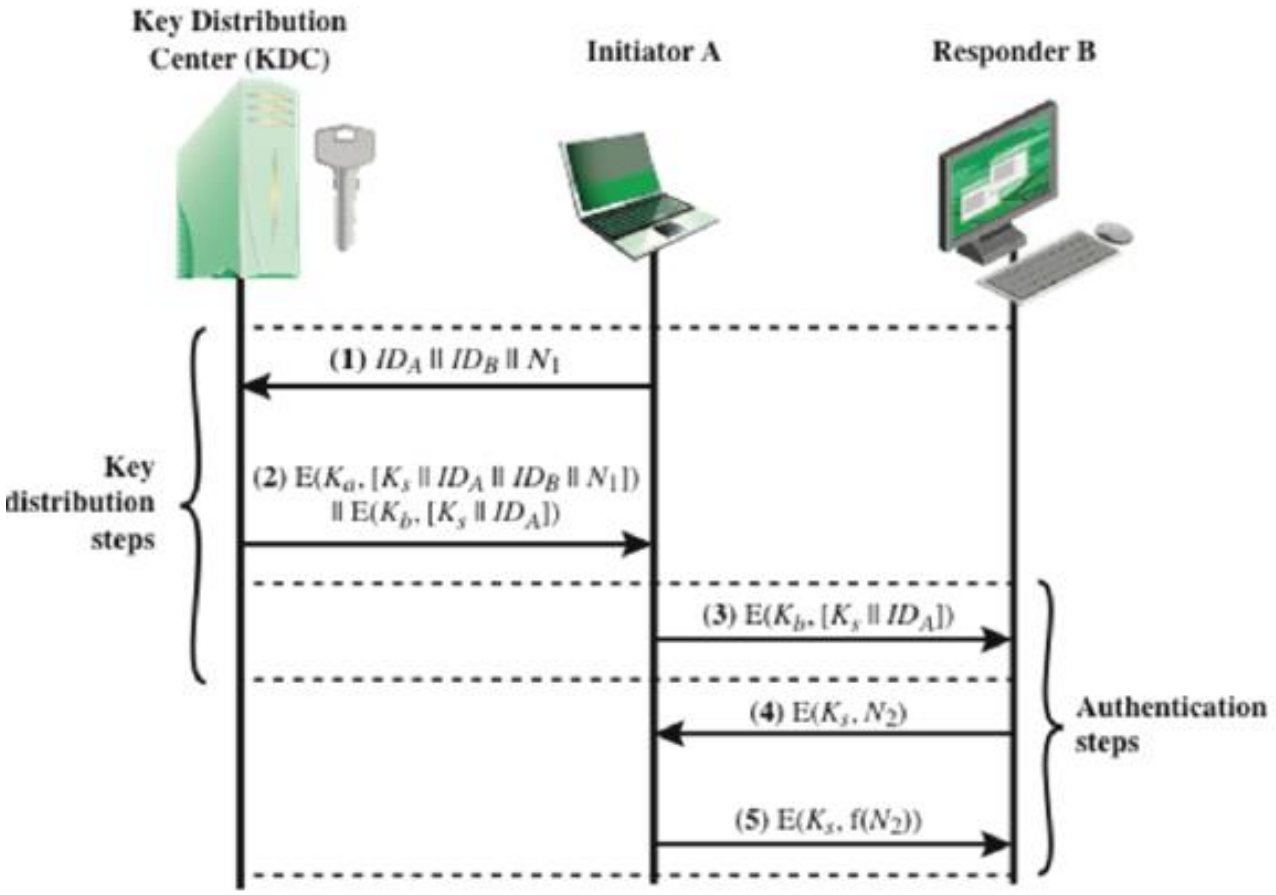


Figure 14.3 Key Distribution Scenario

- 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.
- K_a : A's master key. K_b : B's master key

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, *N*1, for this transaction, which we refer to as a **nonce**.

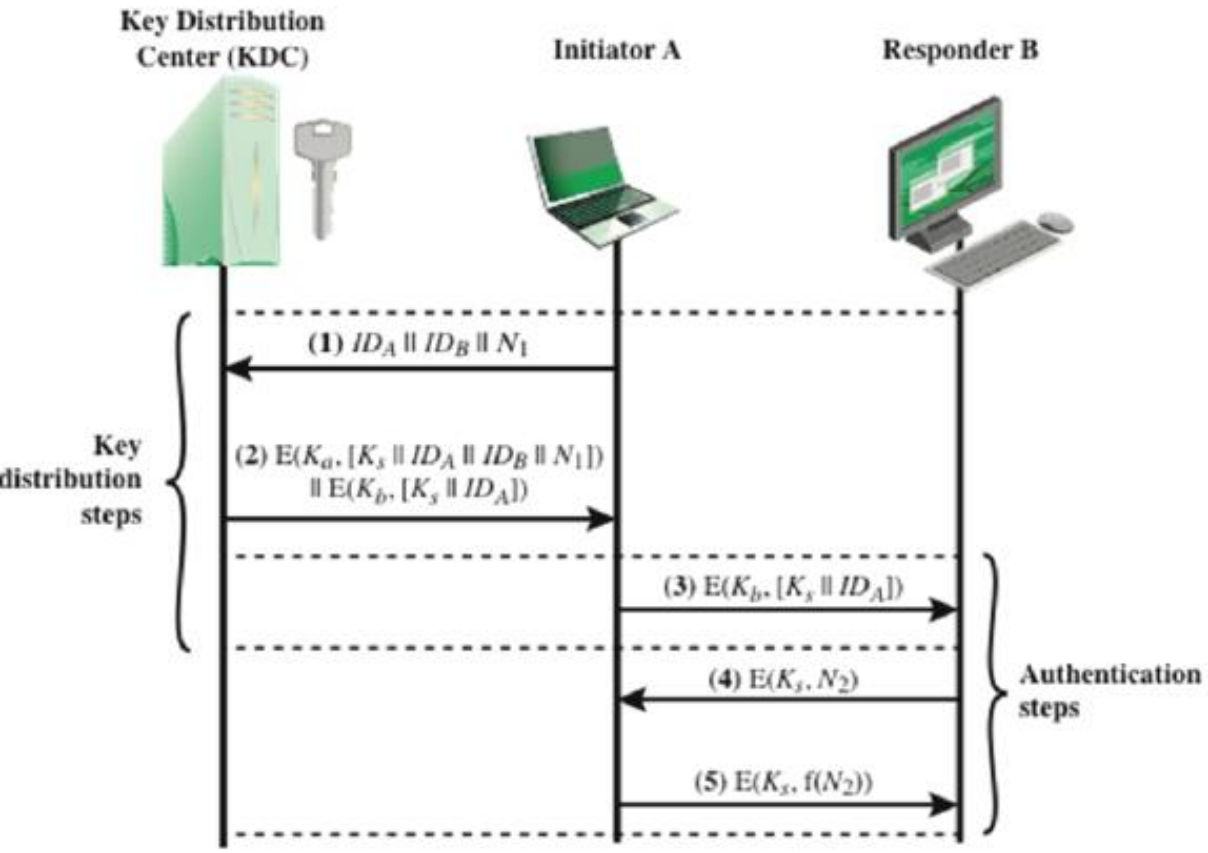


Figure 14.3 Key Distribution Scenario

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 intended for A:

   - The one-time session key, K_s, 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, K_s, to be used for the session
   - An identifier of A (e.g., its network address), ID_A

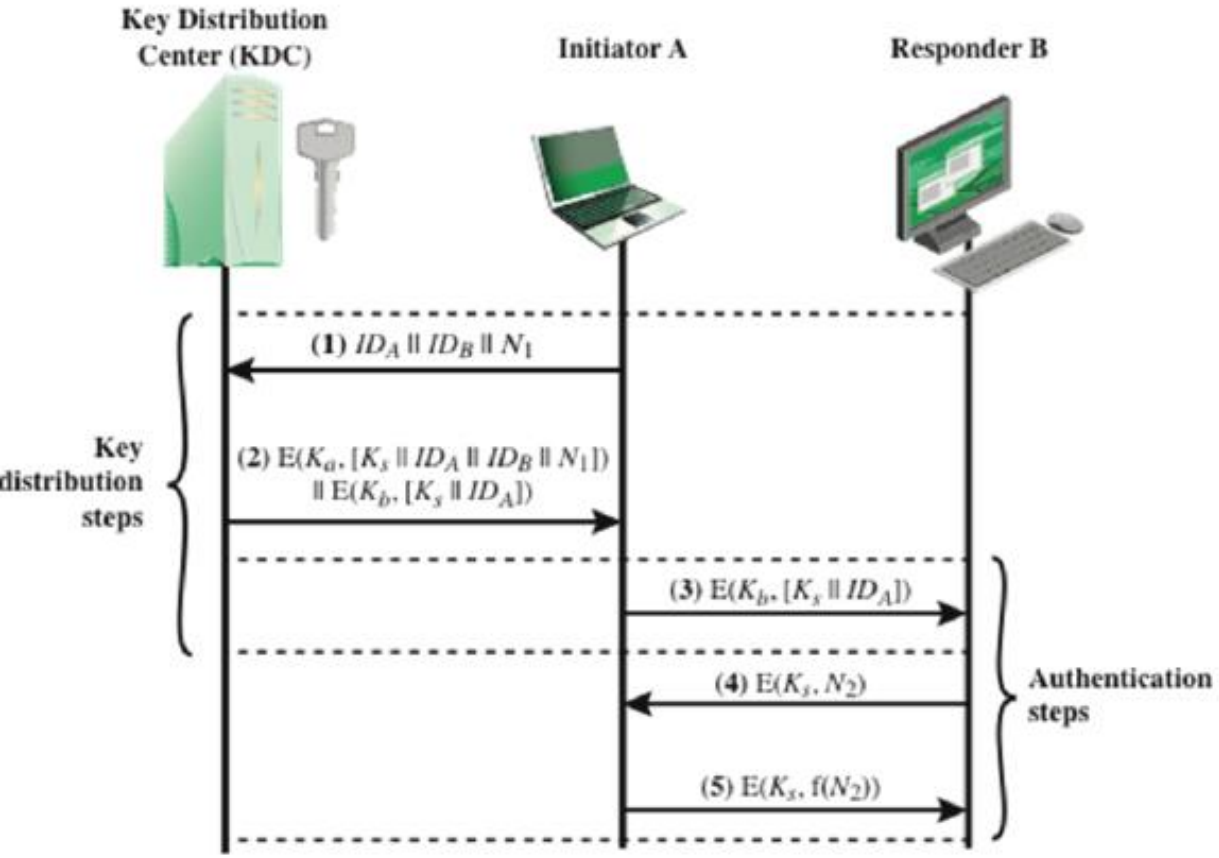


Figure 14.3 Key Distribution Scenario

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 new session key for encryption, B sends a nonce, *N*2, to A.
5. Also, using *Ks*, A responds with f(*N*2), where f is a function that performs some transformation on *N*2

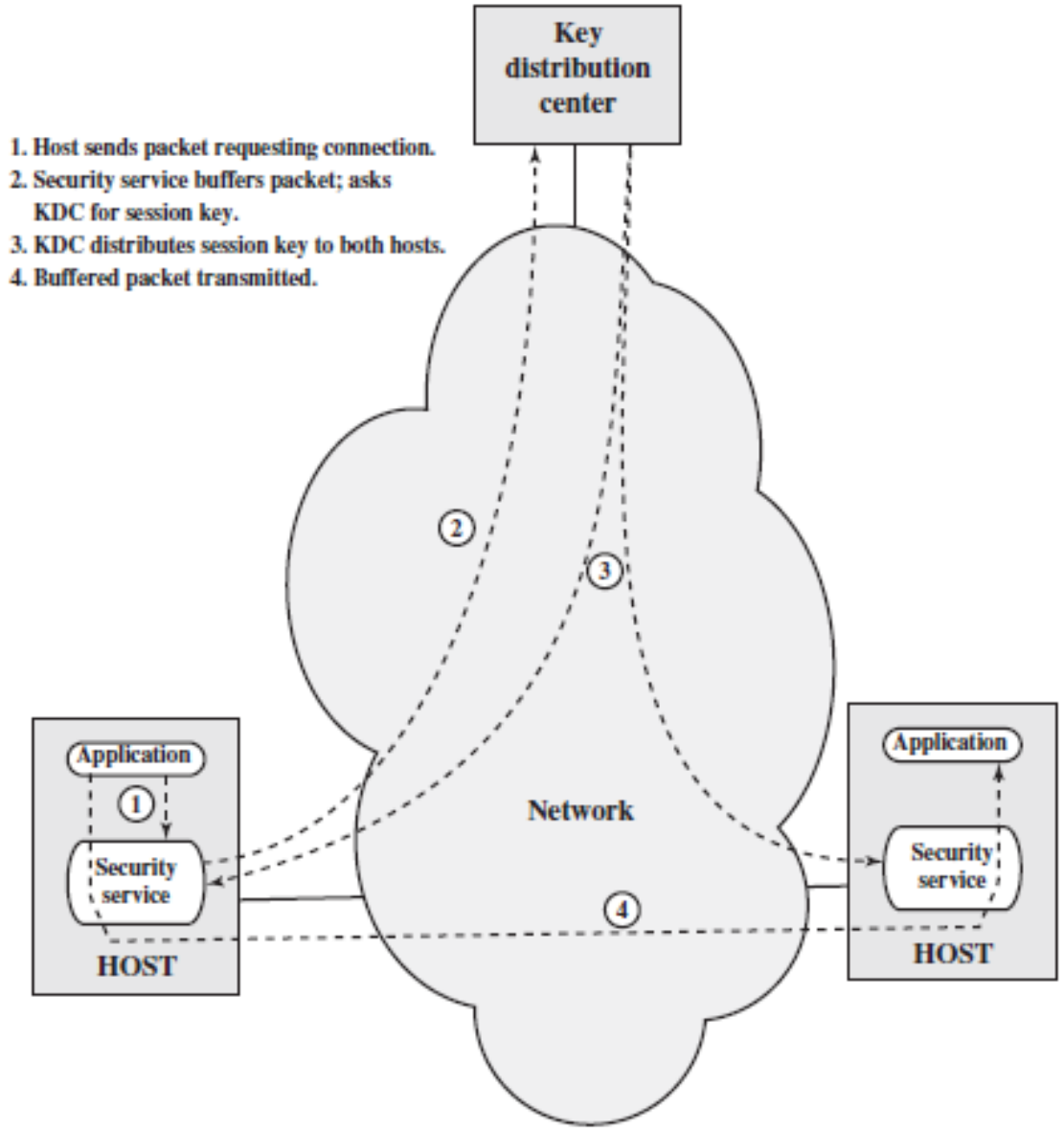
Hierarchical Key Control

- Hierarchy of KDCs can be established. 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.
- Any one of the three KDCs involved can actually select the key.
- The hierarchical concept can be extended to three or even more layers, depending on the size of the user population and the geographic scope of the internetwork.
- Advantages:
    - Minimizes the effort involved in master key distribution
    - Limits the damage of a faulty KDC to its local area only.

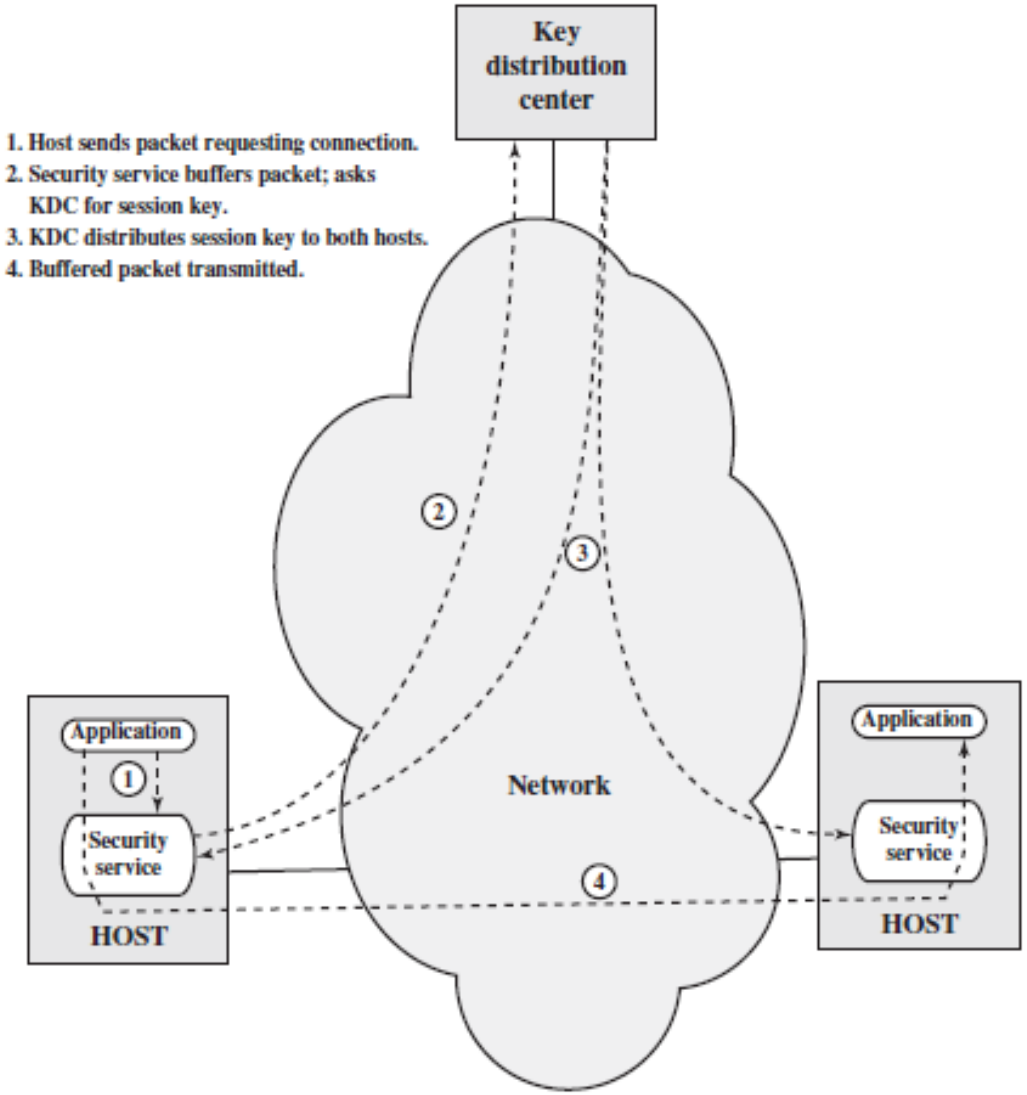
Session Key Lifetime

- For connection-oriented protocols, 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
- For a connectionless 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.
- However, this negates one of the principal benefits of connectionless protocols, which is minimum overhead and delay for each transaction.
- 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



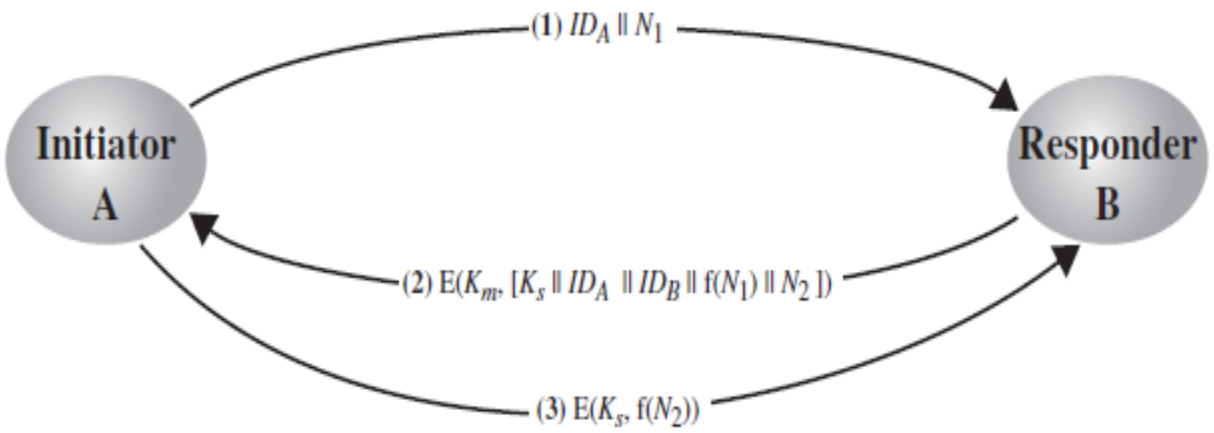
- When one host wishes to set up a connection to another host, it transmits a connection-request packet (step 1).
- The SSM saves that packet and applies to the KDC for permission to establish the connection (step 2).
- The communication between the SSM and the KDC is encrypted using a master key shared only by this SSM and the KDC.
- 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 3).



- The requesting SSM can now release the connection request packet, and a connection is set up between the two end systems (step 4).
- All user data exchanged between the two end systems are encrypted by their respective SSMs using the one-time session key.
- The automated key distribution approach provides the flexibility and dynamic characteristics needed to allow a number of terminal users to access a number of hosts and for the hosts to exchange data with each other

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.



- A session key may be established with the following sequence of steps

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. 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

- It is 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

- It may be desirable to institute controls in systems that limit the ways in which keys are used, based on characteristics associated with those keys.
- One simple plan is to associate a tag with each key
  - use extra 8 bits in each key to from 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.

- Drawbacks :

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.

Control vector

- 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

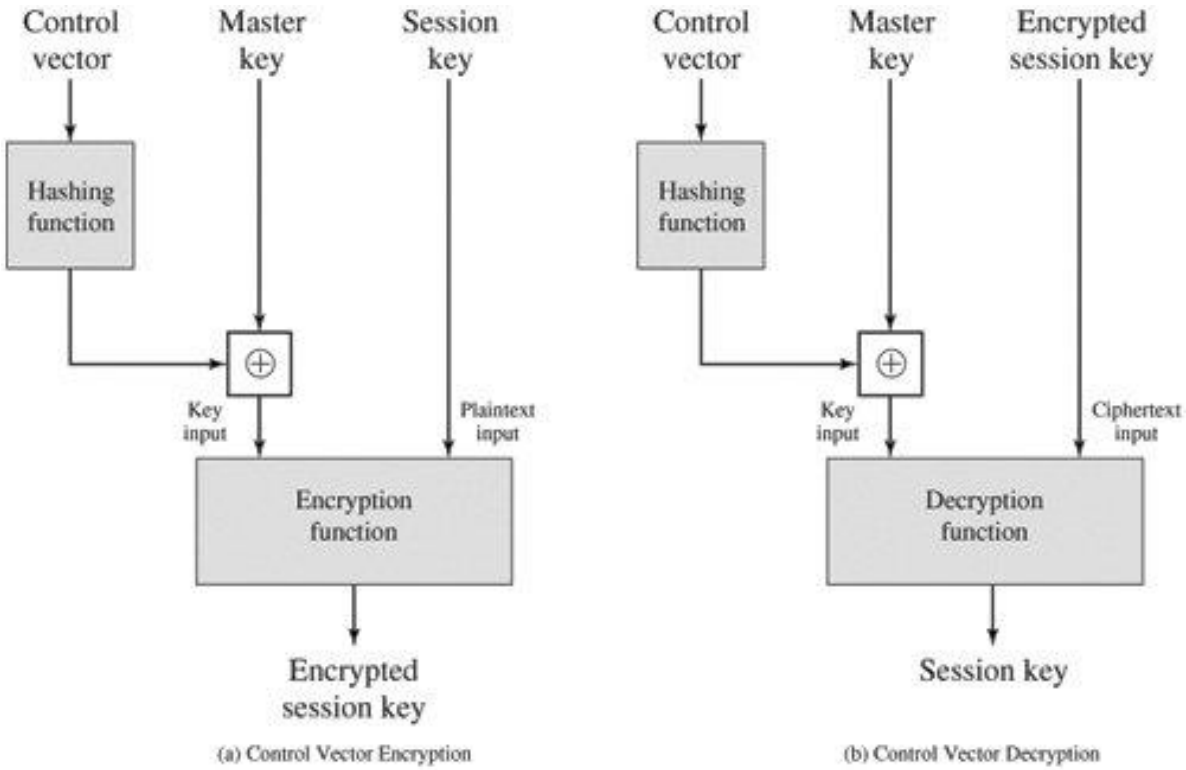


Fig: Control vector Encryption and Decryption

- 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.

$$\text{Hash value} = H = \text{h(CV)}$$
$$\text{Key input} = K_m \oplus H$$
$$\text{Ciphertext} = \text{E}([K_m \oplus H], K_s)$$

- K_m : master key

- K_s : session key

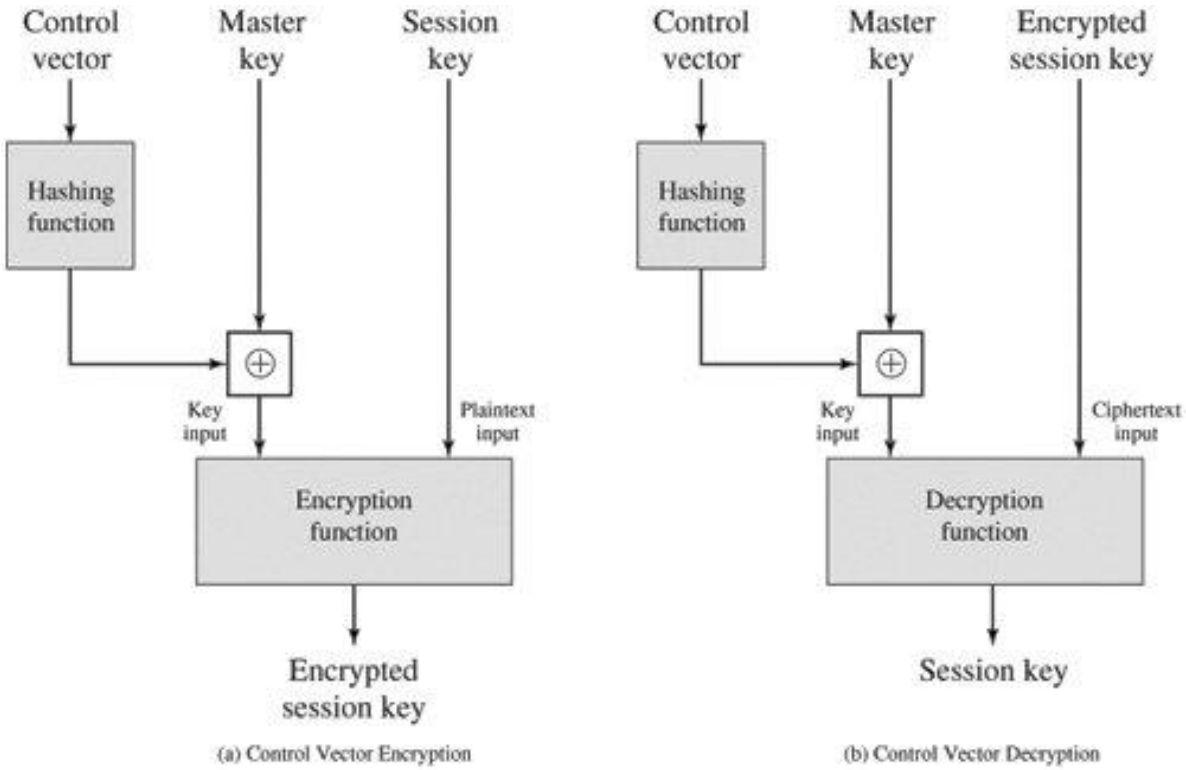


Fig: Control vector Encryption and Decryption

- The session key is recovered in plaintext by the reverse operation:

$$D([K_m \oplus H], E([K_m \oplus H], K_s))$$

- When a session key is delivered to a user from the KDC, it is accompanied by the control vector in clear form.
- The session key can be recovered only by using both the master key that the user shares with the KDC and the control vector.
- Thus, the linkage between the session key and its control vector is maintained.

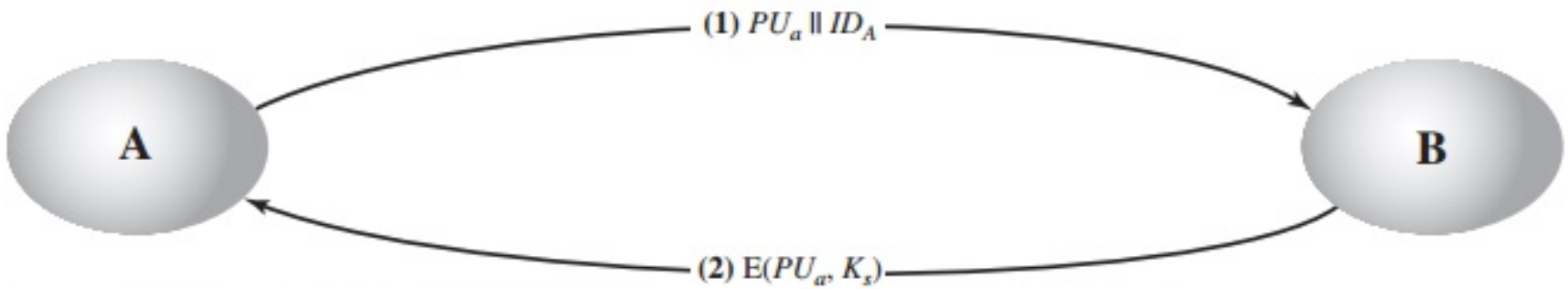
- Use of the control vector has two advantages over use of an 8-bit tag.

  1. There is no restriction on length of the control vector, which enables arbitrarily complex controls to be imposed on key use.
  2. 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

- One of the most important uses of a public-key cryptosystem is to encrypt secret keys for distribution

Simple Secret Key Distribution



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.

- A and B can now securely communicate using conventional encryption and the session key K_s.
- At the completion of the exchange, both A and B discard K_s.
- Advantges

1. Simple
2. 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.
3. The communication is secure from eavesdropping.

- Drawback

- Prone to man-in-the-middle attack. if an adversary, D, has control of the intervening communication channel, then D can compromise the communication in the following fashion without being detected

  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 || *IDA* 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.

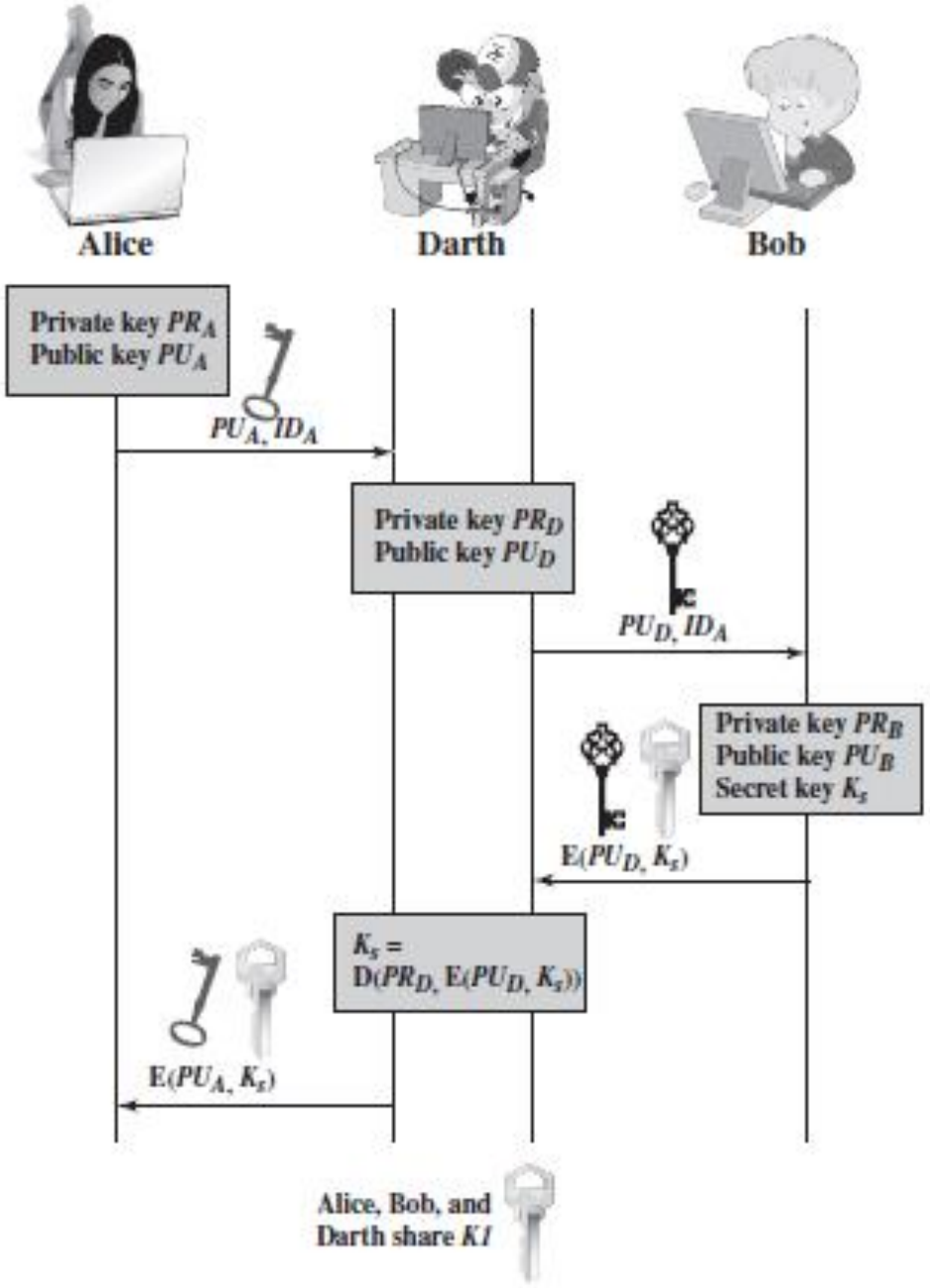


Fig: Man-in-the-Middle Attack

- A and B know *Ks* and are unaware that K_s has also been revealed to D.
- A and B can now exchange messages using K_s.
- D no longer actively interferes with the communications channel but simply eavesdrops.
- Knowing K_s, D can decrypt all messages, and both A and B are unaware of the problem.
- Thus, this simple protocol is only useful in an environment where the only threat is eavesdropping.

Secret Key Distribution with Confidentiality and Authentication

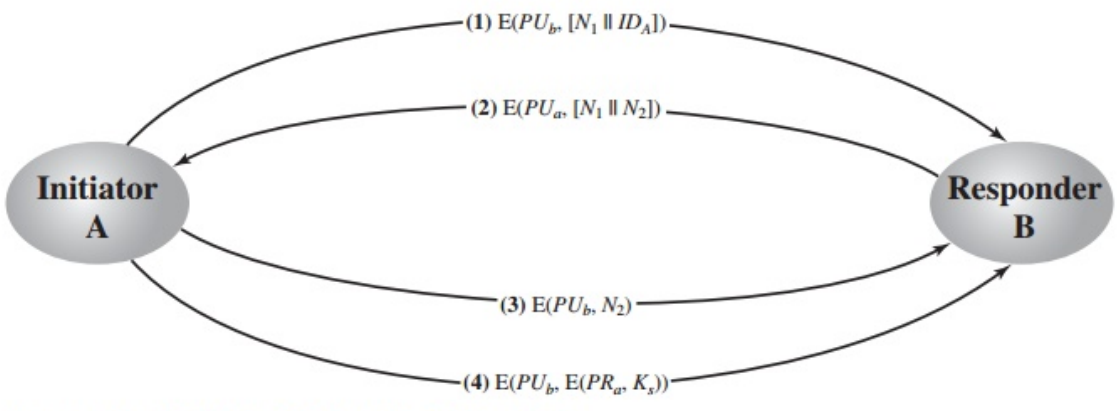


Fig: Public-Key Distribution of Secret Keys

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 N_2, encrypted using B's public key, to assure B that its correspondent is A.
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.
5. B computes D(PU_a, D(PR_b, M)) to recover the secret key.

A Hybrid 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:

1. **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.
2. **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

- 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

Fig: Uncontrolled Public-Key Distribution

Drawback:

- Anyone can forge such a public announcement
- 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

- Maintenance and distribution of the public directory would have to be the responsibility of some trusted entity or organization

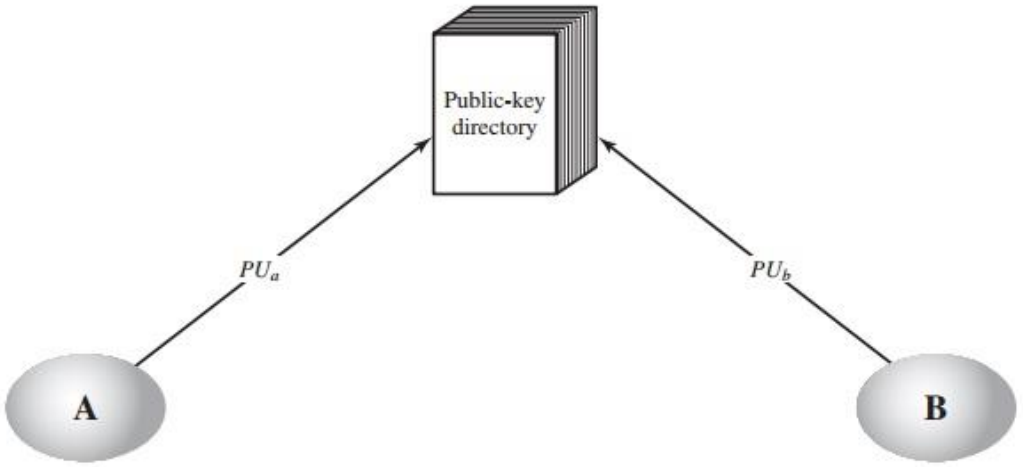


Fig: Public-Key Publication

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.

Advantages:

- more secure than individual public announcements

Disadvantages:

- 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.
- 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
- Assume 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.

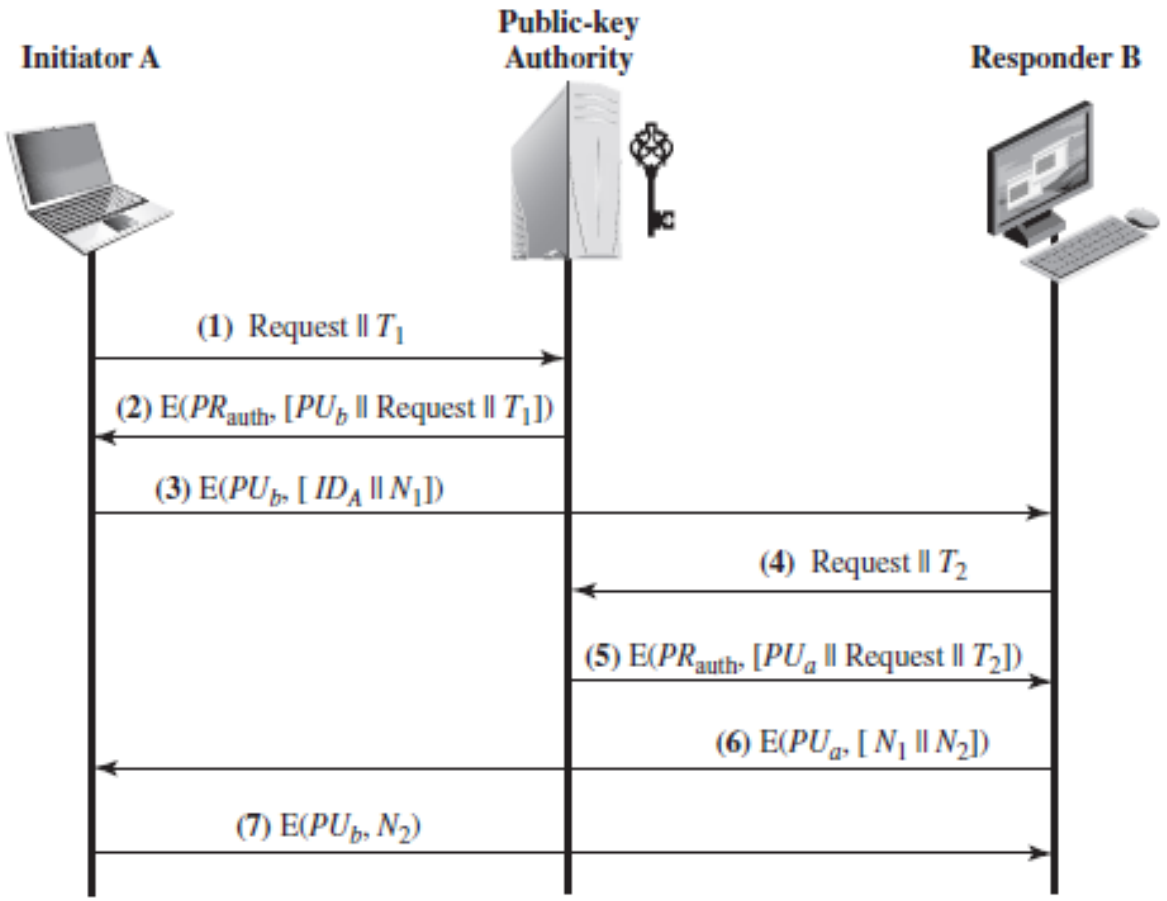


Fig: Public-Key Distribution Scenario

2. The authority responds with a message that is encrypted using the authority's private key, PR_{auth}. 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, PU_b, 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

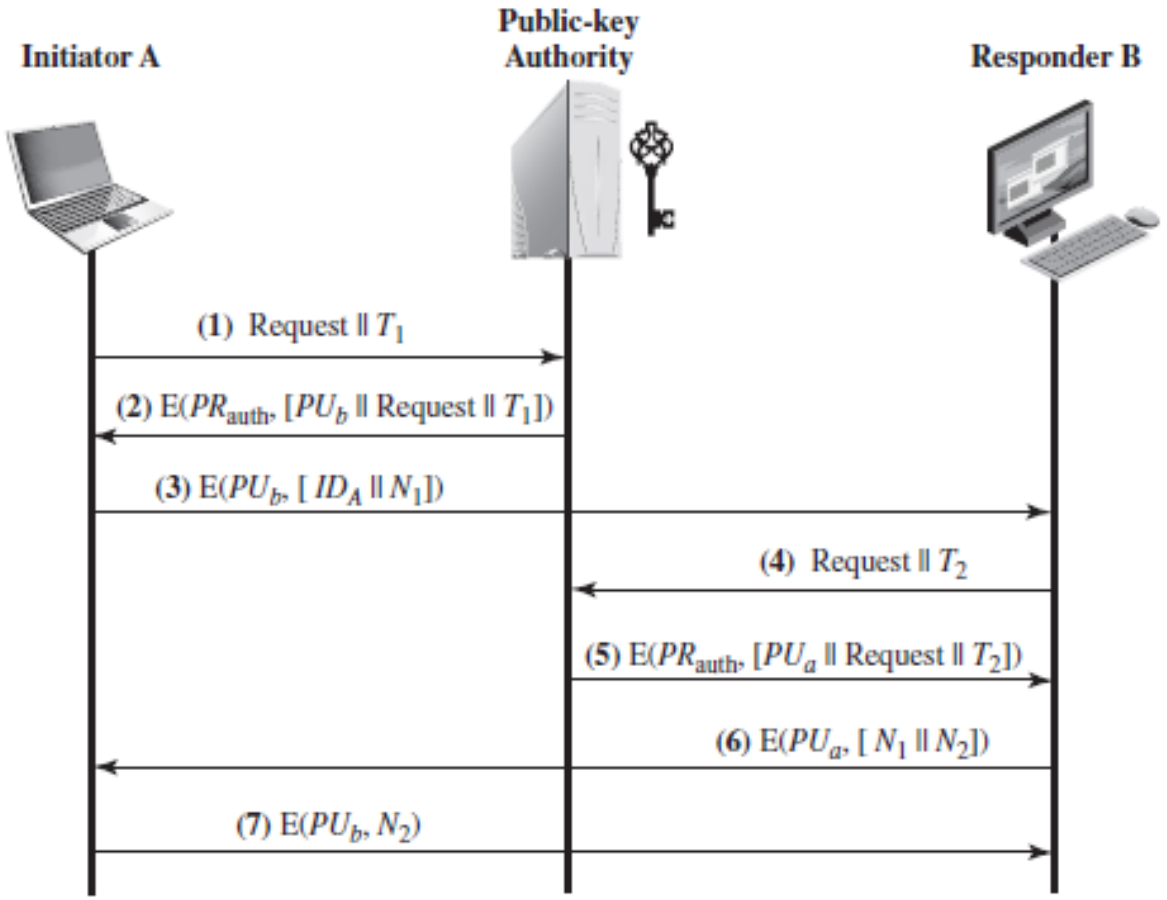


Fig: Public-Key Distribution Scenario

3. A stores B's public key and also uses it 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.

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 (N_1) as well as a new nonce generated by B (N_2). Because only B could have decrypted message (3), the presence of N_1 in message (6) assures A that the correspondent is B.

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

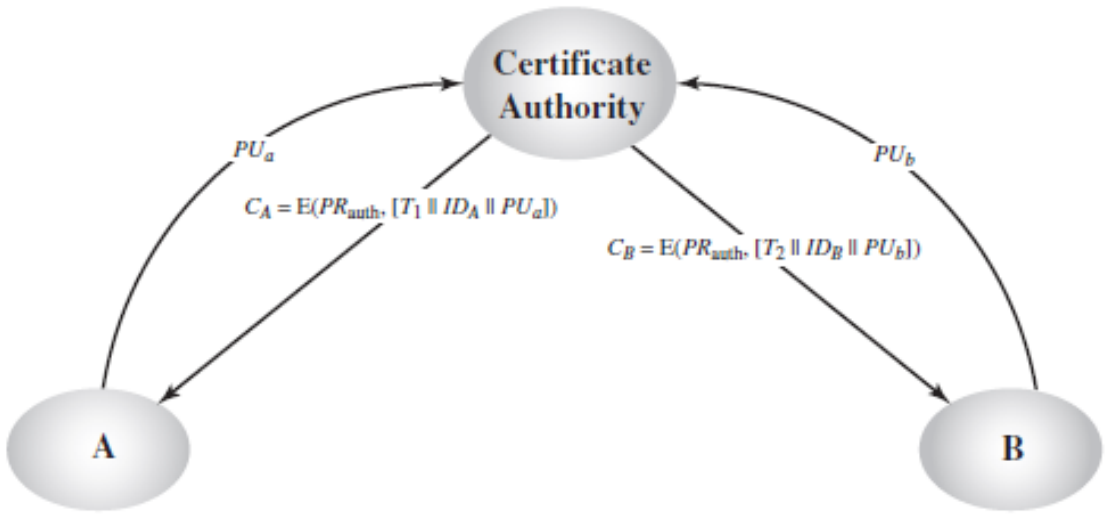
Drawbacks:

- 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.
- The directory of names and public keys maintained by the authority is vulnerable to tampering.

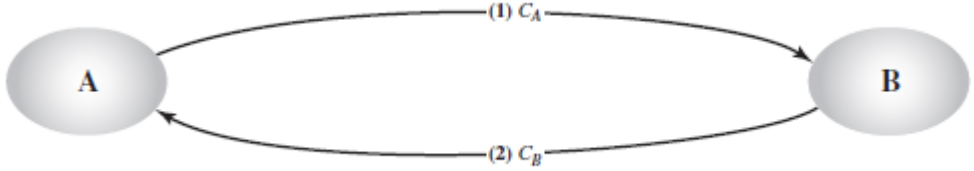
Public-Key Certificates

- 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.
- A certificate consists of a public key, an identifier of the key owner, and the whole block signed by a trusted third party.
- A user can present his or her public key to the authority in a secure manner and obtain a certificate.
- The user can then publish the certificate. Anyone needing 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

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
3. Only the certificate authority can create and update certificates.
4. Any participant can verify the currency of the certificate.



Obtaining certificates from CA



Exchanging certificates

- Each participant applies to the certificate authority, supplying a public key and requesting a certificate.
- For participant A, the authority provides a certificate of the form

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

PR_{auth}: private key used by the authority , T : timestamp.

- A may then pass this certificate on to any other participant, who reads and verifies the certificate as

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

- The recipient uses the authority's public key, $P_{U\text{auth}}$, 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 ID_A and Pu_a provide the recipient with the name and public key of the certificate's holder.
- The timestamp T validates the currency of the certificate.