CHAPTER 10

Key Management; Other public-key Cryptosystem

Key Management

- Several techniques have been proposed for the distribution of public keys.
  1. Public Announcement.
  2. Publicly Available Directory.
  3. Public-key authority.
  4. Public-key certificates.

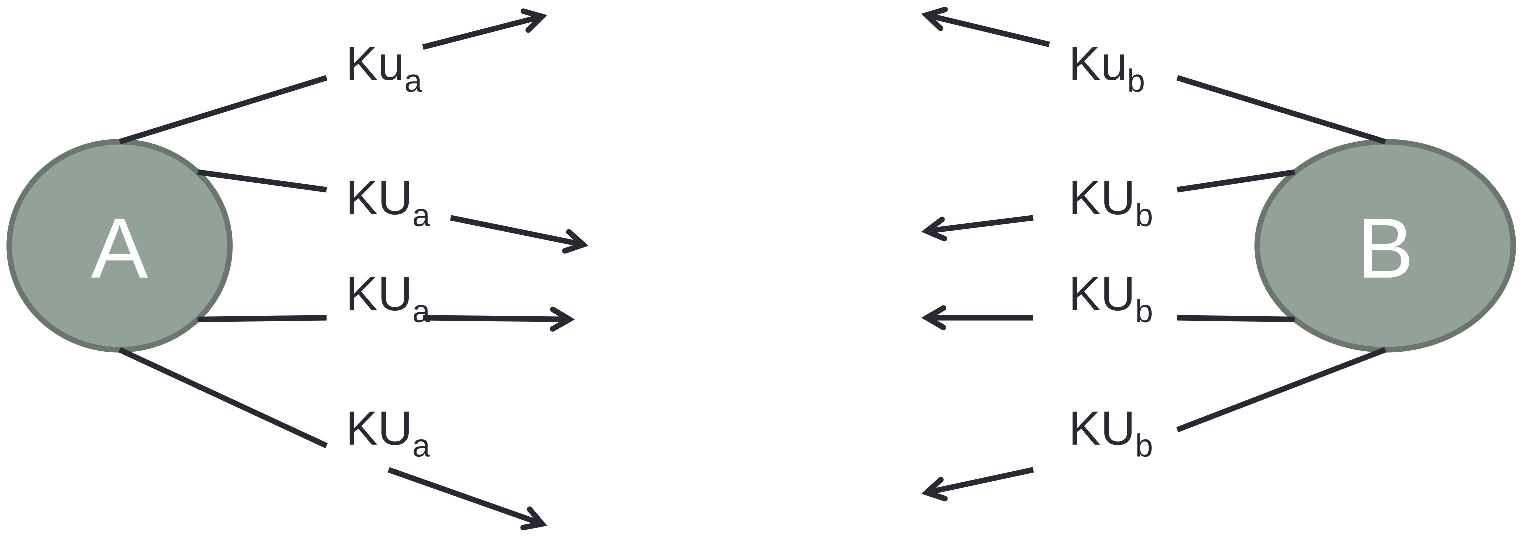
1. Public Announcement of public keys:
   a. Any participant can send his or her public key to any other participants or broadcast the key to the community at large.

   b. Many PGP (pretty good privacy) 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 list.

   c. It has a major weakness; some user could pretend to be user A and send a public key to another participant or broadcast such a key.

Public Announcement of public keys:

Uncontrolled public key distribution:



Publicly Available Directory

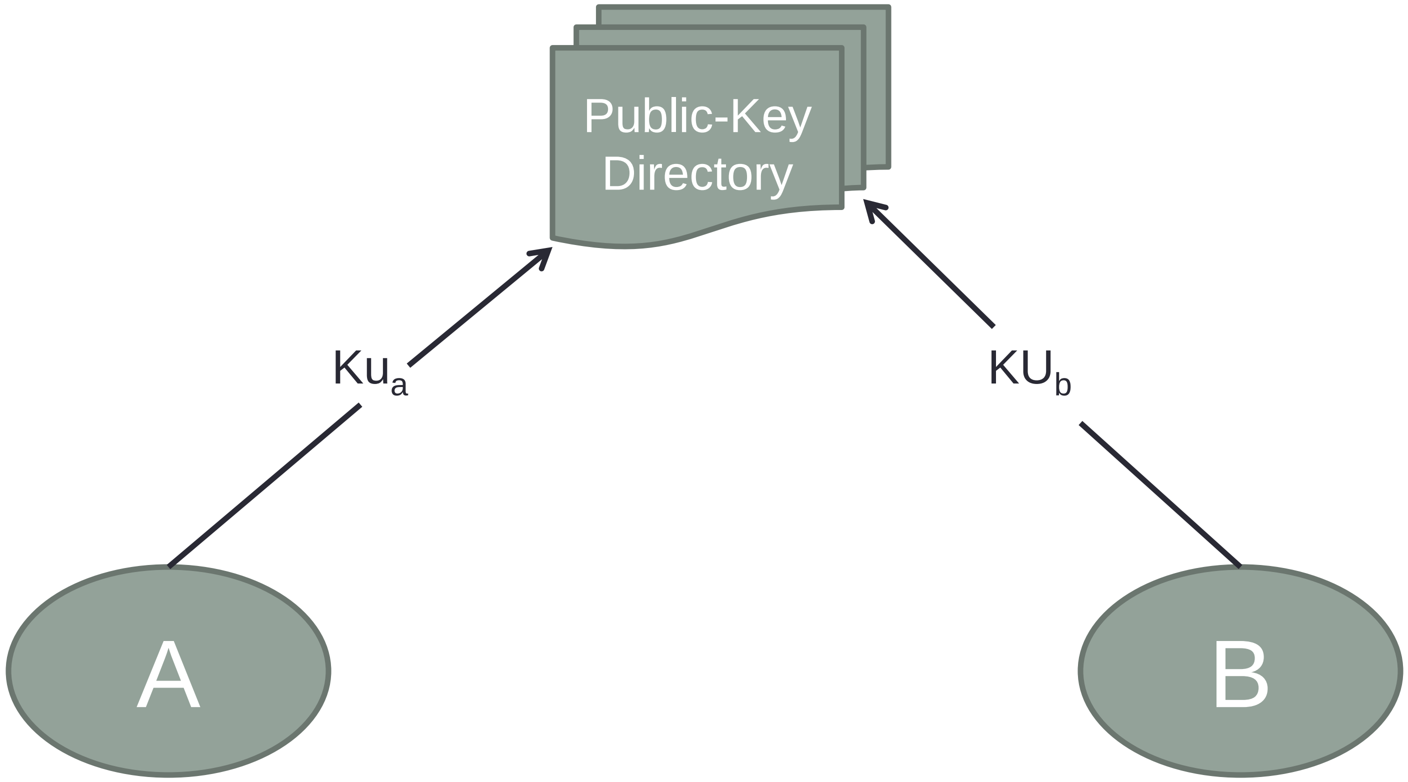
- Public key of each participants are stored on a directory which is maintained by some trusted entity or organization.
- Such a scheme would include the following elements:

1. The authority maintains a directory with a {name, public key} entry for each participant.
2. Registration would have to be in person or by some form of secure authenticated communication.
3. A participants can update the key if needed (key compromised or after large data transfer).
4. Periodically, the authority publishes (hard copy version) the entire directory or updates.
5. Secure authenticated communication from the authority to the participant is mandatory.

It has still vulnerabilities if an opponent succeeds in obtaining or computing private key of the directory authority.

Publicly Available Directory

Public-Key Publication



Public-key Authority

- Stronger security for public-key distribution can be achieved by providing a tighter control over the distribution of public keys from the directory.

- It is assume that each participant reliably knows a public key for the authority.
    1. A sends a request with a timestamp to the authority to know the current public key of B.

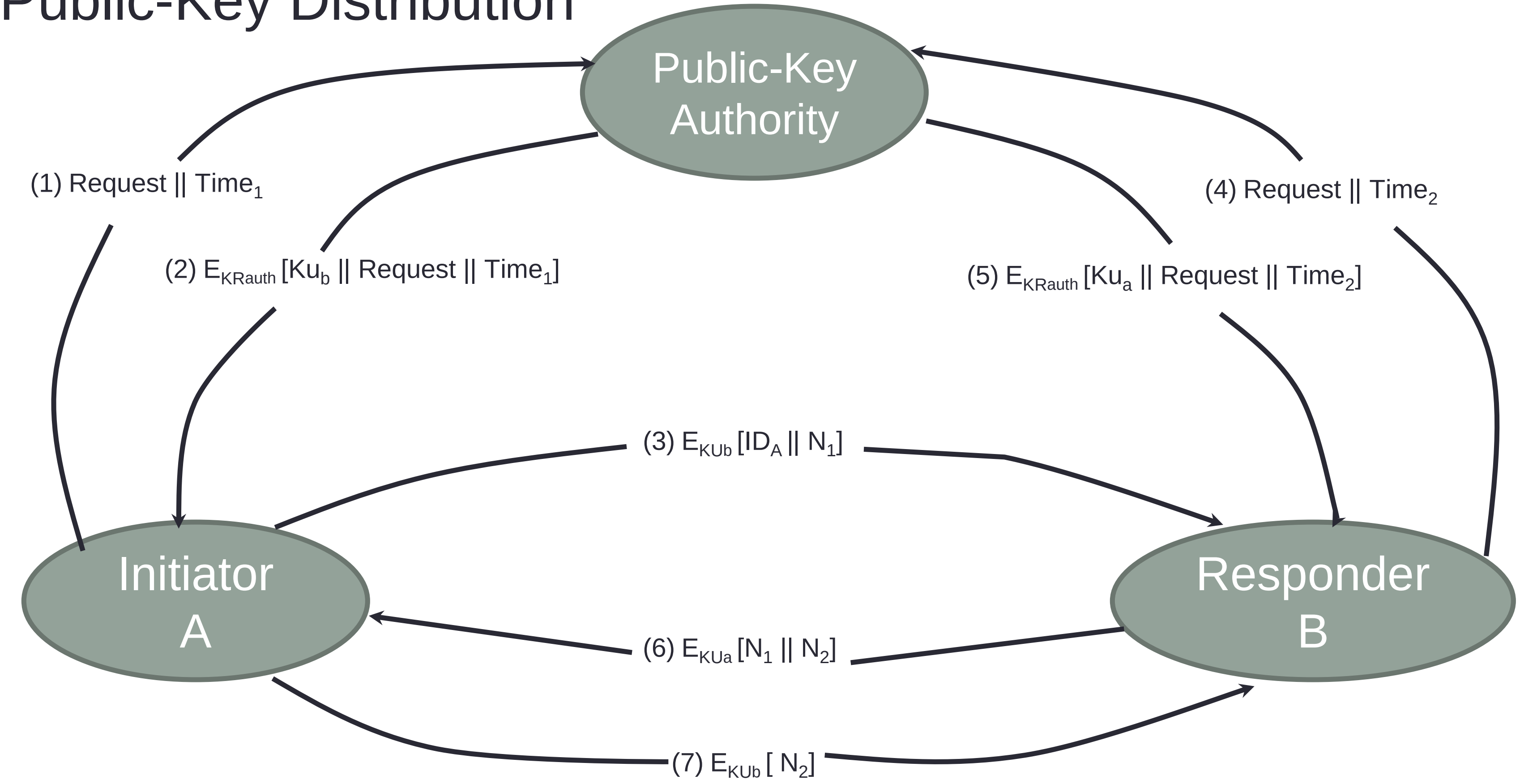
    2. The authority responds by encrypting a message with its own private key KR_{auth}. A can verify the response by decrypting the message with the public key of the authority. The message has:

        a. B's public key, KU_b.
        b. The original request, to enable A that the request was not altered.
        c. The original timestamp to ensure that it is the last request.

    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).

Public-key Authority

Public-Key Distribution


Public-Key Authority
Initiator A
Responder B
(1) Request || $Time_1$
(2) E_{KRauth} [Ku_b || Request || $Time_1$]
(3) E_{KUb} [ID_A || N_1]
(4) Request || $Time_2$
(5) E_{KRauth} [Ku_a || Request || $Time_2$]
(6) E_{KUa} [N_1 || N_2]
(7) E_{KUb} [N_2]

Public-key Authority Continue…

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 key have been securely delivered to A and B, and they may begin their protected exchange. However the two additional steps are desirable:

6. B sends a message to A encrypted with KU_a and containing A's nonce (N_1) as well as a new nonce (N_2).

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

The initial four messages need be used infrequently because both A and B can save the other's public key for future use.

Public-Key Certificate

- Public-Key authority has some drawbacks:
  - The key authority could be bottleneck, since before any data transfer a sender (user) appeal to the authority for receiver(another user) a public key.
- An alternative approach is to use Certificates that can be used by participants to exchange keys without contacting a public key authority.
- Any participants can read a certificate to determine the name and public key of the certificate's owner.
- Any participants can verify that the certificate originated from the certificate authority and is not counterfeit.
- Only the certificate authority can create and update certificate.
- For participant A, the authority provides a certificate of the form.

$$C_A = E_{KRauth} [T, ID_A, KU_a]$$

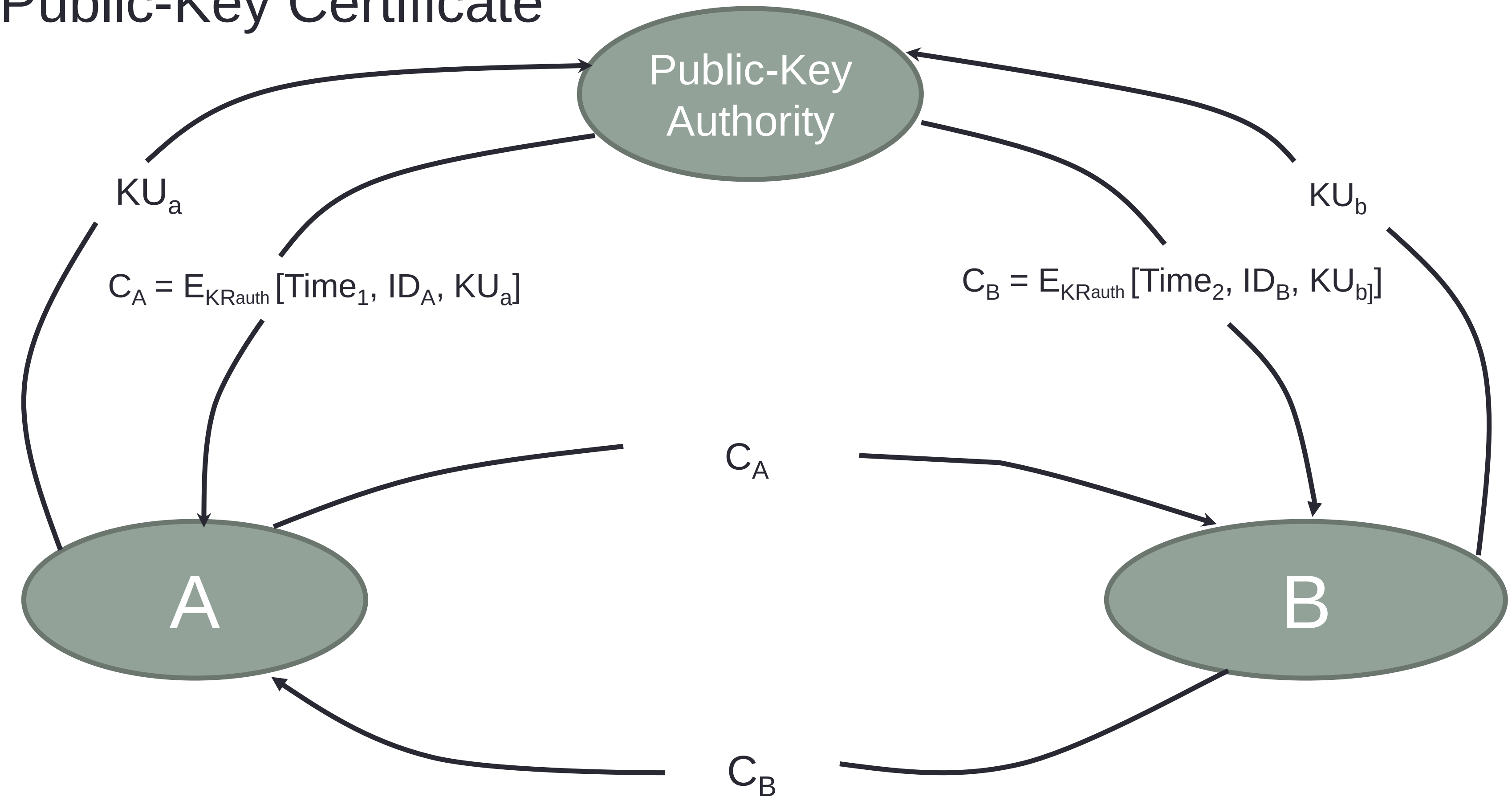
Where KR_{auth} is the private key used by the authority. A may then pass this certificate on to any other participant, who reads and verifies the certificates as follows:

$$D_{KUauth} [C_A] = D_{KUauth} [E_{KRauth} [T, ID_A, KU_a]] = (T, ID_A, KU_a)$$

- The recipient uses the authority's public key KU_{auth} to decrypt the certificate.

Public-Key Certificate

Public-Key Certificate


Public-Key Authority
KU_a
KU_b
$C_A = E_{KRauth}$ [Time$_1$, ID_A, KU_a]
$C_B = E_{KRauth}$ [Time$_2$, ID_B, KU_b]
C_A
C_B
A
B

Diffie-Hellman Key Exchange

- The algorithm is named after Diffie and Hellman.
- The purpose is to enable two users to exchange a key securely that can then be used for subsequent encryption of messages.

- First we define a primitive root of a prime number p as one whose powers generate all the integers from 1 to p-1. That is, if a is a primitive root of the prime number p, then the numbers

  $a \bmod p, a^2 \bmod p, \ldots ,a^{p-1} \bmod p$

  are distinct and consist of the integers from 1 through p-1 in some permutation. Example: prime 13 has a primitive root of 2.

Primitive root Example

$2 \bmod 13 = 2$

$2^2 \bmod 13 = 4$

$2^3 \bmod 13 = 8$

$2^4 \bmod 13 = 3$

$2^5 \bmod 13 = 6$

$2^6 \bmod 13 = 12$

$2^7 \bmod 13 = 11$

$2^8 \bmod 13 = 9$

$2^9 \bmod 13 = 5$

$2^{10} \bmod 13 = 10$

$2^{11} \bmod 13 = 7$

$2^{12} \bmod 13 = 1$

Diffie-Hellman Key Exchange Continue…

For any integer b and a primitive root a of prime number p, we can find a unique exponent i such that

$b \equiv a^i \bmod p$ where $0 \leq i \leq (p-1)$

The exponent i is referred to as the discrete logarithm or index of b for the base a mod p. This value is denoted as $ind_{a,p}(b)$.

- Suppose q is a prime number and α is a primitive root of q.
- User A selects a random integer $X_A<q$ and computes $Y_A = \alpha^{X_A} \bmod q$.

- Similarly B selects a random integer $X_B<q$ and computes $Y_B = \alpha^{X_B} \bmod q$.
- Each side keeps X value as private and Y value as public.

- User A computes the key as $K = (Y_B)^{X_A} \bmod q$ and B computes the key as $K=(Y_A)^{X_B} \bmod q$.

Diffie-Hellman Key Exchange Continue…

- These two calculations produce identical results:

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

$=(\alpha^{X_B} \bmod q)^{X_A} \bmod q$

$=(\alpha^{X_B})^{X_A} \bmod q$ by modular rules.

$=\alpha^{X_B X_A} \bmod q$

$=(\alpha^{X_A})^{X_B} \bmod q$

$=(\alpha^{X_A} \bmod q)^{X_B} \bmod q$

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

The attacker may compute $X_B = ind_{\alpha,q}(Y_B)$ to get secret key of B.

The security of Diffie-Hellman key exchange lies in the fact that, while it is relatively easy to calculate exponentials modulo a prime, it is very difficult to calculate discrete logarithms. For large prime the later task is infeasible.

Diffie-Hellman Key Exchange

- Suppose prime q=353, primitive root α =3, A's secret key X_A=97 and B's secret X_B=233.

  A computes $Y_A = 3^{97} \bmod 353 = 40$

  B computes $Y_B = 3^{233} \bmod 353 = 248$

After the exchange of public key, each can compute the common key as follows:

  A computes $K = (Y_B)^{X_A} \bmod 353 = 248^{97} \bmod 353 = 160$

  B computes $K = (Y_A)^{X_B} \bmod 353 = 40^{233} \bmod 353 = 160$