



Key Management and Public Key Cryptosystems

Key Management

- public-key encryption helps to address the key distribution problems
- have two aspects of this:
 - distribution of public keys
 - use of public-key encryption to distribute secret keys

Distribution of Public Keys

- can be considered as using one of:
 - public announcement
 - publicly available directory
 - public-key authority
 - public-key certificates

Public Announcement

- users distribute public keys to recipients or broadcast to community at large
 - eg. append PGP keys to email messages or post to news groups or email list
- major weakness is forgery
 - anyone can create a key claiming to be someone else and broadcast it
 - until forgery is discovered can masquerade as claimed user

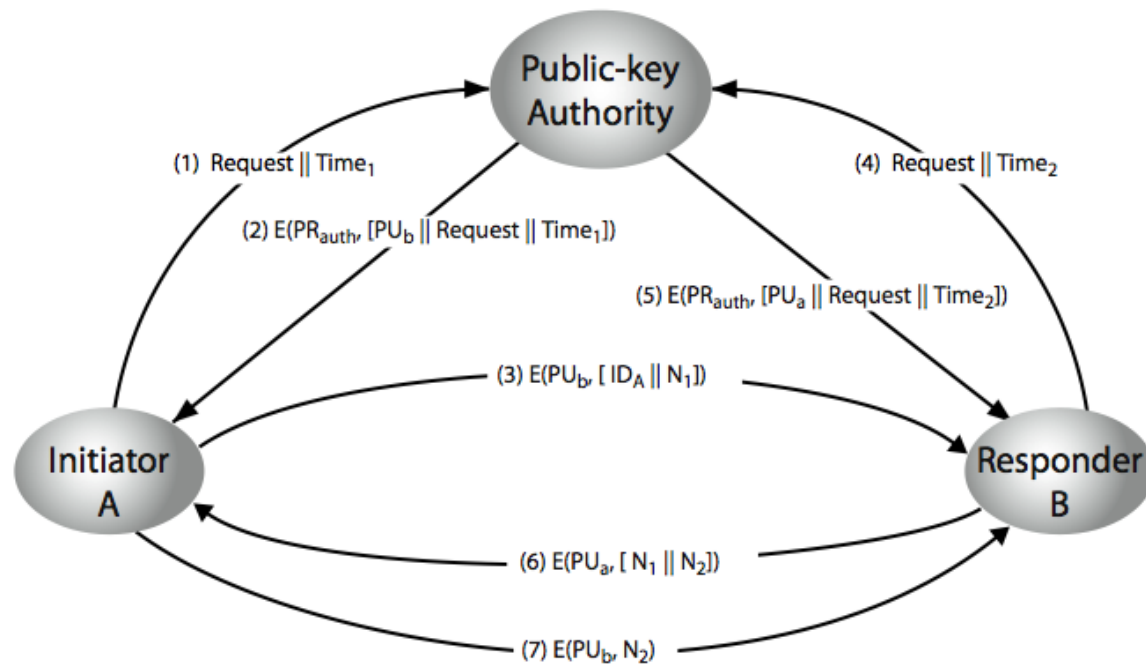
Publicly Available Directory

- can obtain greater security by registering keys with a public directory
- directory must be trusted with properties:
 - contains {name,public-key} entries
 - participants register securely with directory
 - participants can replace key at any time
 - directory is periodically published
 - directory can be accessed electronically
- still vulnerable to tampering or forgery

Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- has properties of directory
- and requires users to know public key for the directory
- then users interact with directory to obtain any desired public key securely
 - does require real-time access to directory when keys are needed

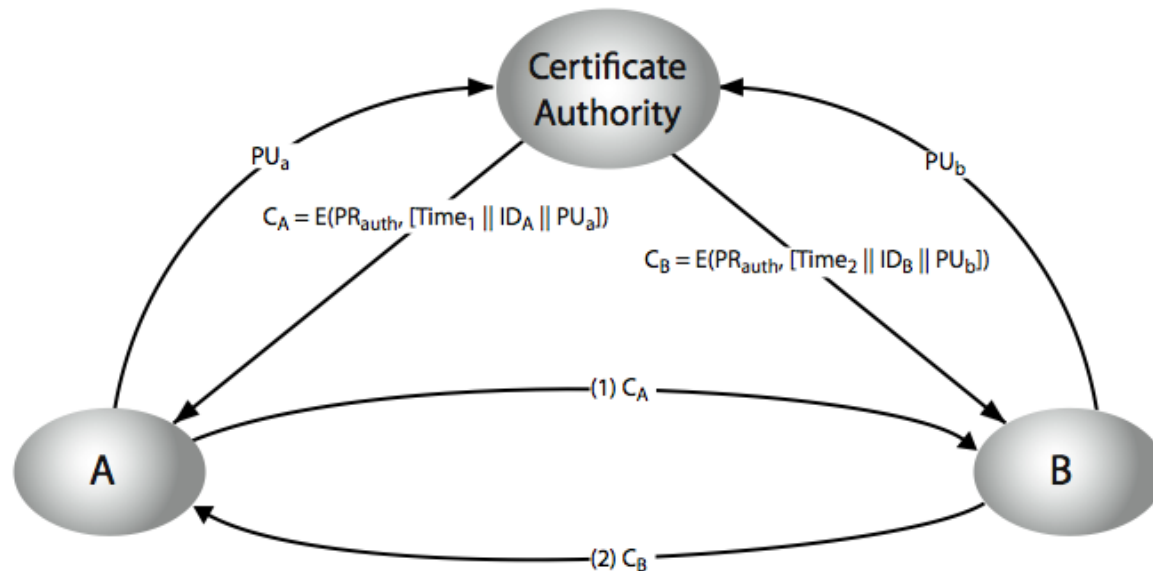
Public-Key Authority



Public-Key Certificates

- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
 - usually with other info such as period of validity, rights of use etc
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
- can be verified by anyone who knows the public-key authorities public-key

Public-Key Certificates



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

Public-Key Distribution of Secret Keys

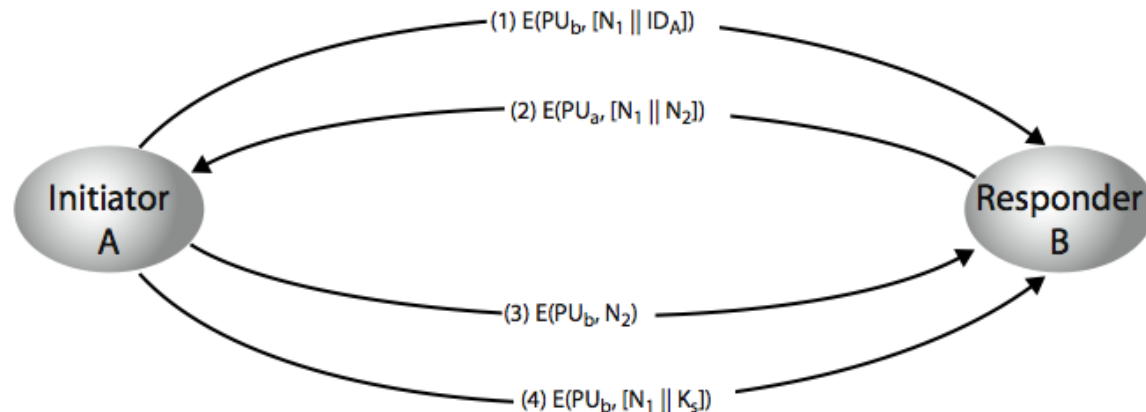
- use previous methods to obtain public-key
- can use for secrecy or authentication
- but public-key algorithms are slow
- so usually want to use private-key encryption to protect message contents
- hence need a session key
- have several alternatives for negotiating a suitable session

Simple Secret Key Distribution

- proposed by Merkle in 1979
 - A generates a new temporary public key pair
 - A sends B the public key and their identity
 - B generates a session key K sends it to A encrypted using the supplied public key
 - A decrypts the session key and both use
- problem is that an opponent can intercept and impersonate both halves of protocol

Public-Key Distribution of Secret Keys

- if have securely exchanged public-keys:



Hybrid Key Distribution

- retain use of private-key KDC
- shares secret master key with each user
- distributes session key using master key
- public-key used to distribute master keys
 - especially useful with widely distributed users
- rationale
 - performance
 - backward compatibility

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
 - note: now know that Williamson (UK CESG) secretly proposed the concept in 1970
- is a practical method for public exchange of a secret key
- used in a number of commercial products

Diffie-Hellman Key Exchange

- Public-key distribution scheme
 - cannot be used to exchange an arbitrary message
 - rather it can establish a common key
 - known only to the two participants
- value of key depends on the participants (and their private and public key information)
- based on exponentiation in a finite (Galois) field (modulo a prime or a polynomial) - easy
- security relies on the difficulty of computing discrete logarithms (similar to factoring) – hard

Diffie-Hellman Setup

- all users agree on global parameters:
 - large prime integer or polynomial q
 - a being a primitive root mod q
- each user (eg. A) generates their key
 - chooses a secret key (number): $x_A < q$
 - compute their **public key**: $y_A = a^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K_{AB} :

$$\begin{aligned} K_{AB} &= a^{x_A \cdot x_B} \bmod q \\ &= y_A^{x_B} \bmod q \quad (\text{which } \mathbf{B} \text{ can compute}) \\ &= y_B^{x_A} \bmod q \quad (\text{which } \mathbf{A} \text{ can compute}) \end{aligned}$$

- K_{AB} is used as session key in private-key encryption scheme between Alice and Bob
- if Alice and Bob subsequently communicate, they will have the **same** key as before, unless they choose new public-keys
- attacker needs an x , must solve discrete log

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime $q=353$ and $a=3$
- select random secret keys:
 - A chooses $x_A=97$, B chooses $x_B=233$
- compute respective public keys:
 - $Y_A = 3^{97} \bmod 353 = 40$ (Alice)
 - $Y_B = 3^{233} \bmod 353 = 248$ (Bob)
- compute shared session key as:
 - $K_{AB} = Y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
 - $K_{AB} = Y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Key Exchange Protocols

- users could create random private/public D-H keys each time they communicate
- users could create a known private/public D-H key and publish in a directory, then consulted and used to securely communicate with them
- both of these are vulnerable to a meet-in-the-Middle Attack
- authentication of the keys is needed