Cryptography and Network Security

Third Edition

by William Stallings

Lecture slides by Lawrie Brown

Chapter 10 – Key Management; Other Public Key Cryptosystems

No Singhalese, whether man or woman, would venture out of the house without a bunch of keys in his hand, for without such a talisman he would fear that some devil might take advantage of his weak state to slip into his body.

—*The Golden Bough,* Sir James George Frazer

Key Management

- public-key encryption helps address 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
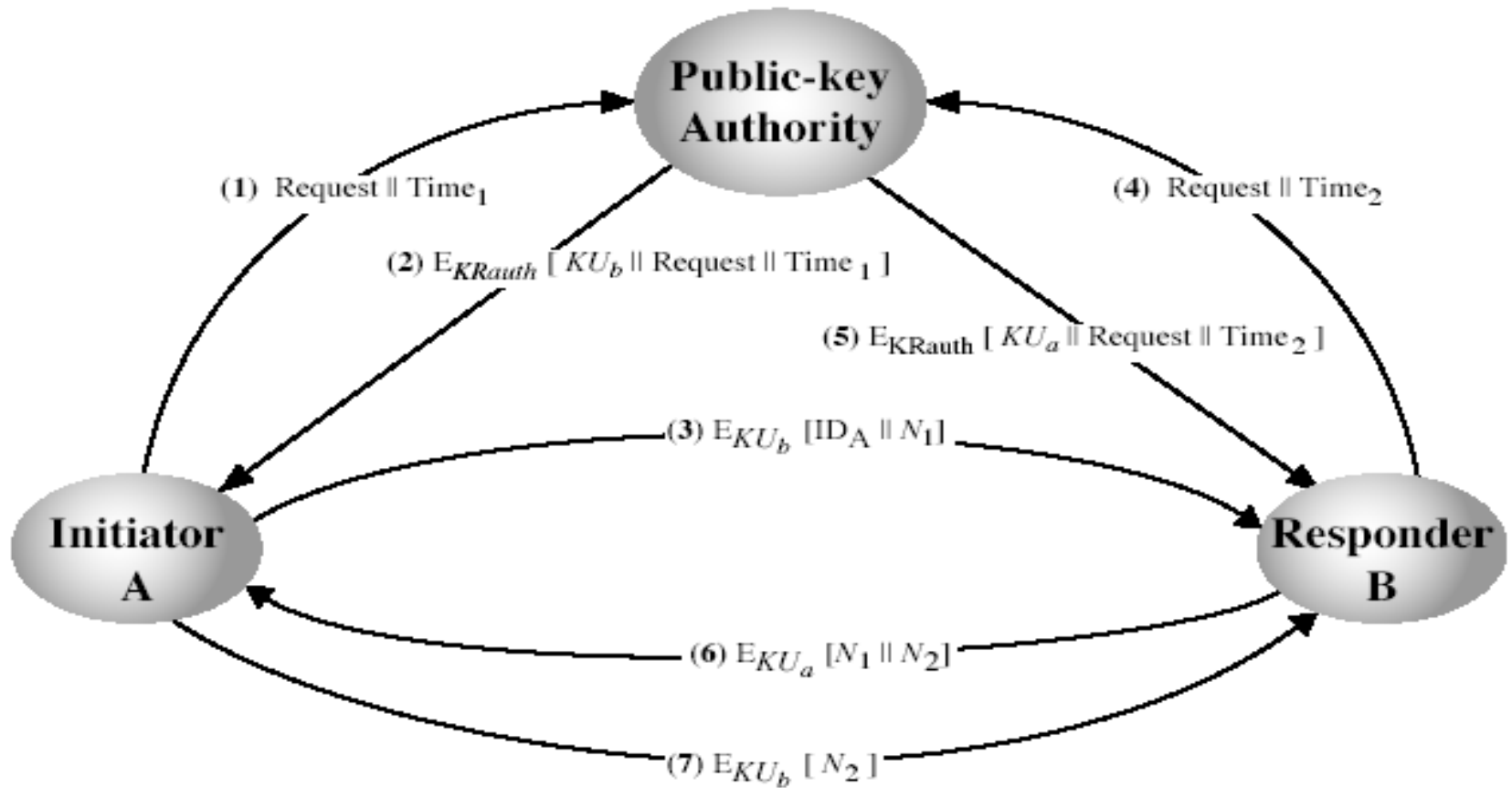
Publicly Available Directory

- 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
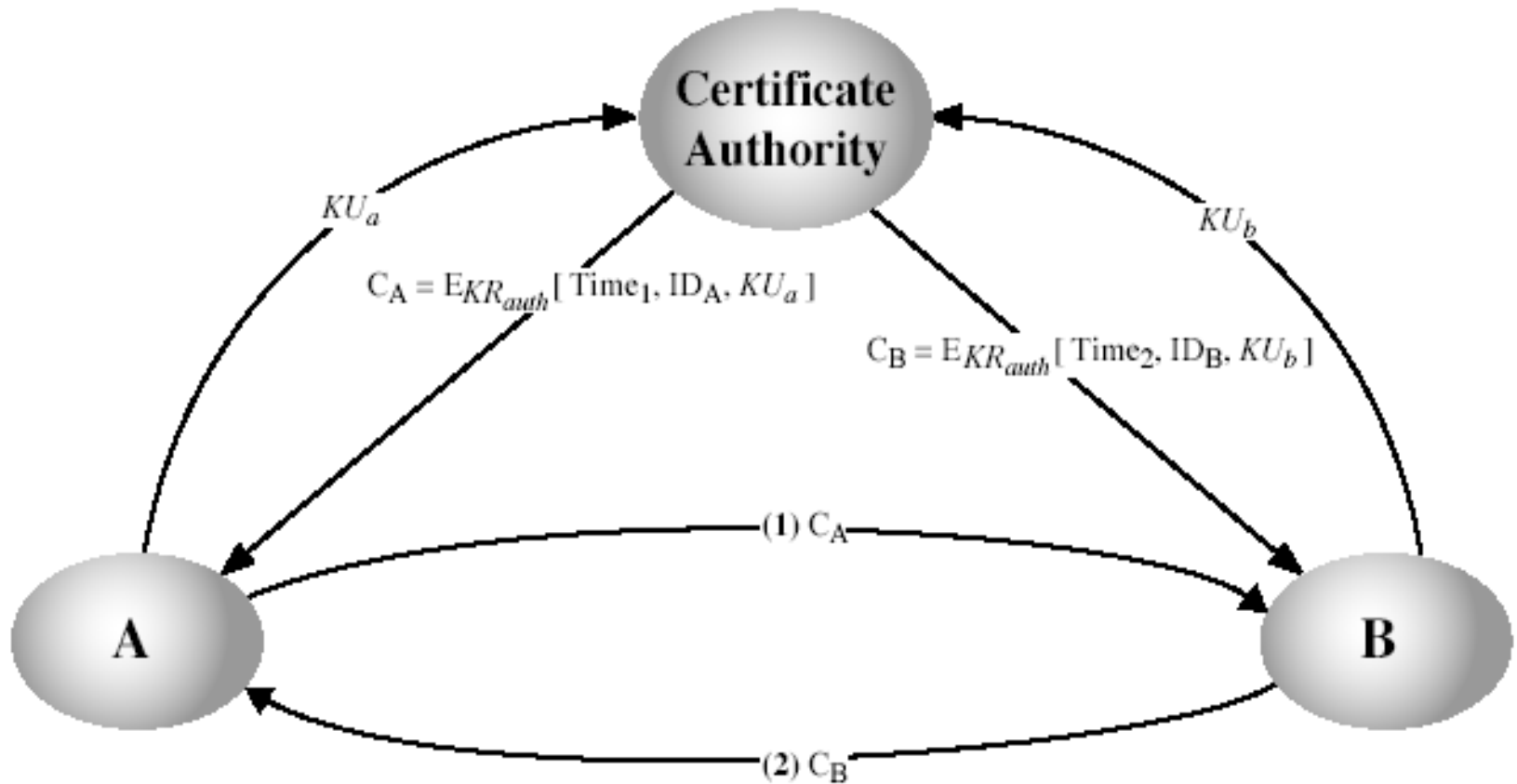 - does require real-time access to directory when keys are needed

Public-Key Authority

Public-key Authority

(1) Request $\parallel$ Time$_1$

(2) E_{KRauth} [$KU_b \parallel$ Request $\parallel$ Time$_1$]

(4) Request $\parallel$ Time$_2$

(5) E_{KRauth} [$KU_a \parallel$ Request $\parallel$ Time$_2$]

(3) E_{KU_b} [ID$_A \parallel N_1$]

Initiator A

Responder B

(6) E_{KU_a} [$N_1 \parallel N_2$]

(7) E_{KU_b} [N_2]

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

Certificate Authority

KU_a

KU_b

$C_A = E_{KR_{auth}}[\text{Time}_1, \text{ID}_A, KU_a]$

$C_B = E_{KR_{auth}}[\text{Time}_2, \text{ID}_B, KU_b]$

A

B

(1) C_A

(2) C_B

# 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:
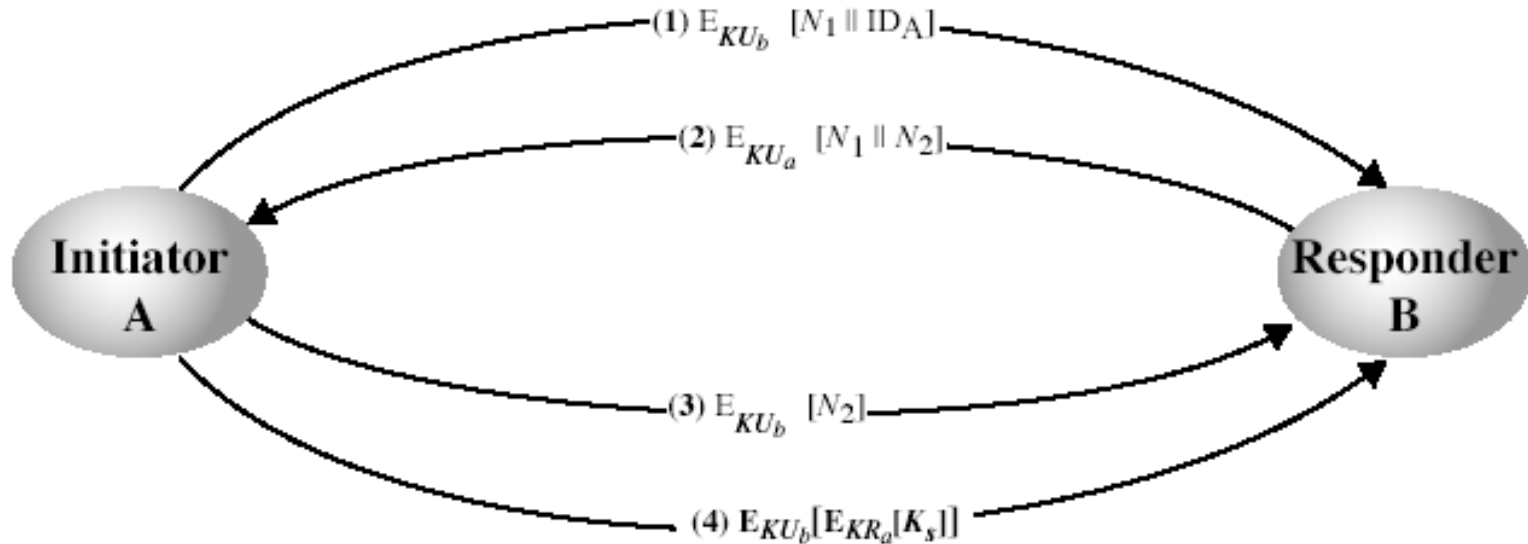
Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
  - note: now know that James Ellis (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

- a 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
  - α a primitive root mod q
- each user (eg. A) generates their key
  - chooses a secret key (number): $x_A < q$
  - compute their **public key**: $y_A = α^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

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

$$K_{AB} = \alpha^{x_A \cdot x_B} \bmod q$$
$$= y_A^{x_B} \bmod q \quad \text{(which \textbf{B} can compute)}$$
$$= y_B^{x_A} \bmod q \quad \text{(which \textbf{A} can compute)}$$

- 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 α`=3`
- select random secret keys:
  - A chooses x_A`=97`, B chooses x_B`=233`
- compute public keys:
  - y_A=3^{97} `mod 353 = 40` (Alice)
  - y_B=3^{233} `mod 353 = 248` (Bob)
- compute shared session key as:

K_{AB}= $y_B^{x_A}$ `mod 353` = 248^{97} = `160` (Alice)
K_{AB}= $y_A^{x_B}$ `mod 353` = 40^{233} = `160` (Bob)

Elliptic Curve Cryptography

- 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

Real Elliptic Curves

- 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
  - also define zero point O
- have addition operation for elliptic curve
  - geometrically sum of Q+R is reflection of intersection R

Real Elliptic Curve Example
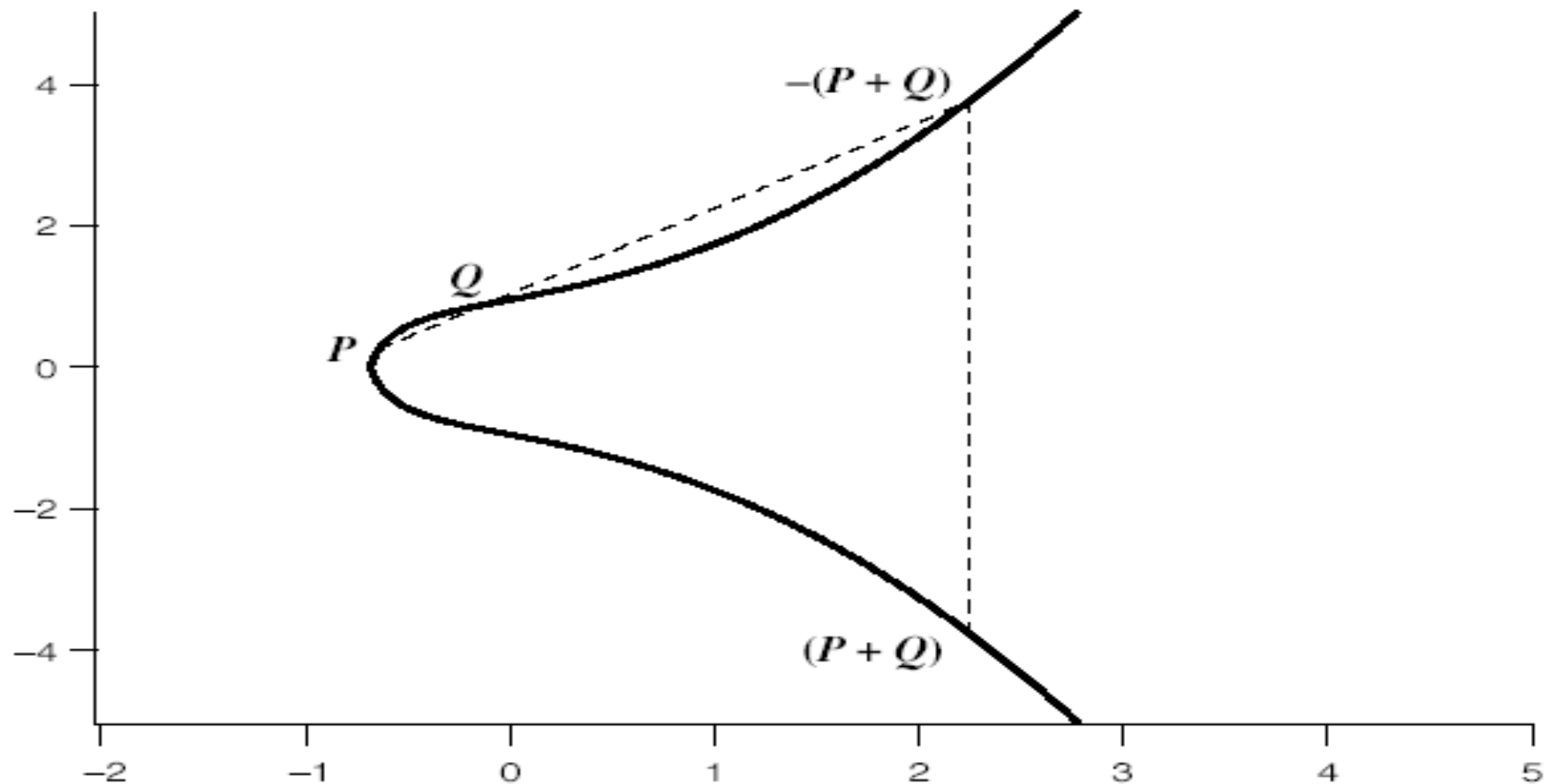
(b) $y^2 = x^3 + x + 1$

Finite Elliptic Curves

- 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 integers modulo a prime
    - best in software
  - binary curves $E_{2m}(a,b)$ defined over $GF(2^n)$
    - use polynomials with binary coefficients
    - best in hardware

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
- Certicom example: $E_{23}(9,17)$

ECC Diffie-Hellman

- can do key exchange analogous to D-H
- users select a suitable curve $E_p(a,b)$
- 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: $P_A=n_A \times G$, $P_B=n_B \times G$
- compute shared key: $K=n_A \times P_B$, $K=n_B \times P_A$
  - same since $K=n_A \times n_B \times G$

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_A×G
- to encrypt P_m : C_m={kG, P_m+k P_b}, k random
- decrypt C_m compute:

$$P_m + k P_b - n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

ECC Security

- relies on elliptic curve logarithm problem
- fastest method is "Pollard rho method"
- compared to factoring, can use much smaller key sizes than with RSA etc
- for equivalent key lengths computations are roughly equivalent
- hence for similar security ECC offers significant computational advantages

Summary

- have considered:
  - distribution of public keys
  - public-key distribution of secret keys
  - Diffie-Hellman key exchange
  - Elliptic Curve cryptography