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
- 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

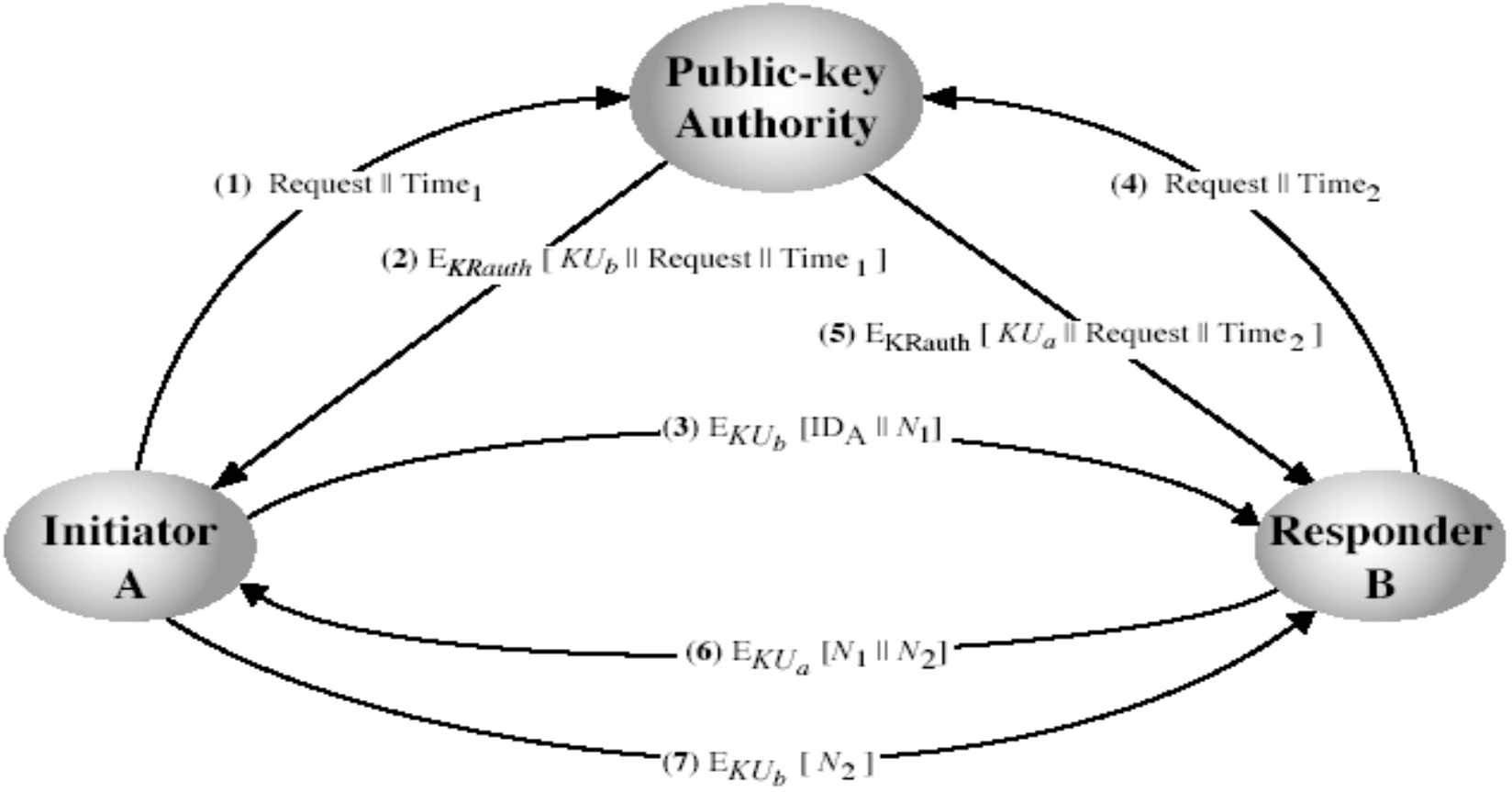
Public Announcement

- 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 for authentication

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



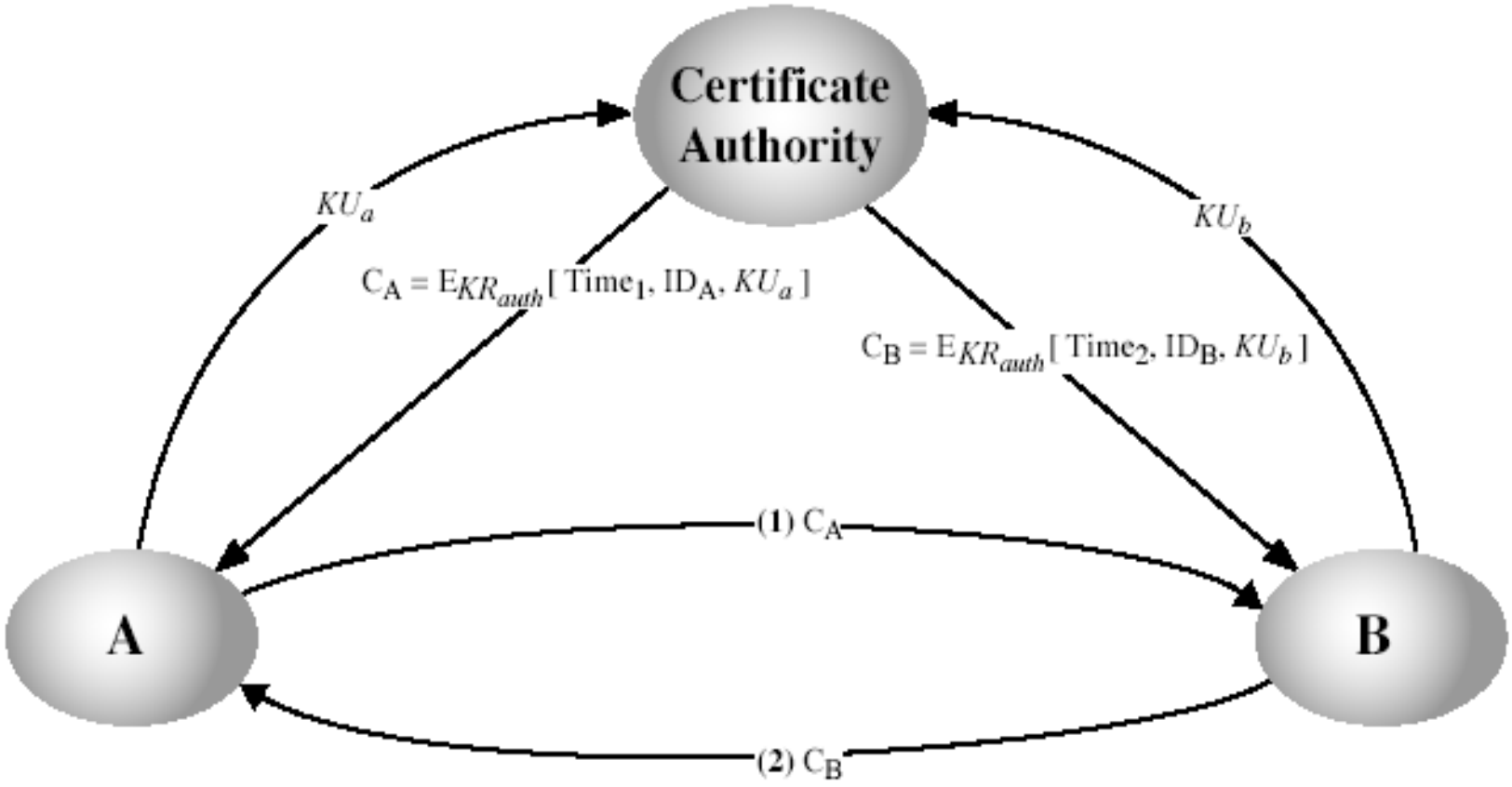
Public-Key Authority

- improve security by tightening control over distribution of keys from directory
- 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 Certificates

- The public-key authority could be a bottleneck in the system.
  - must appeal to the authority for the key of every other user
- certificates allow key exchange without real-time access to public-key authority
- a certificate binds **identity** to **public key**
- with all contents **signed** by a trusted Public-Key or Certificate Authority (CA)
  - Certifies the identity
  - Only the CA can make the certificates

Public-Key Certificates



Public-Key Distribution of Secret Keys

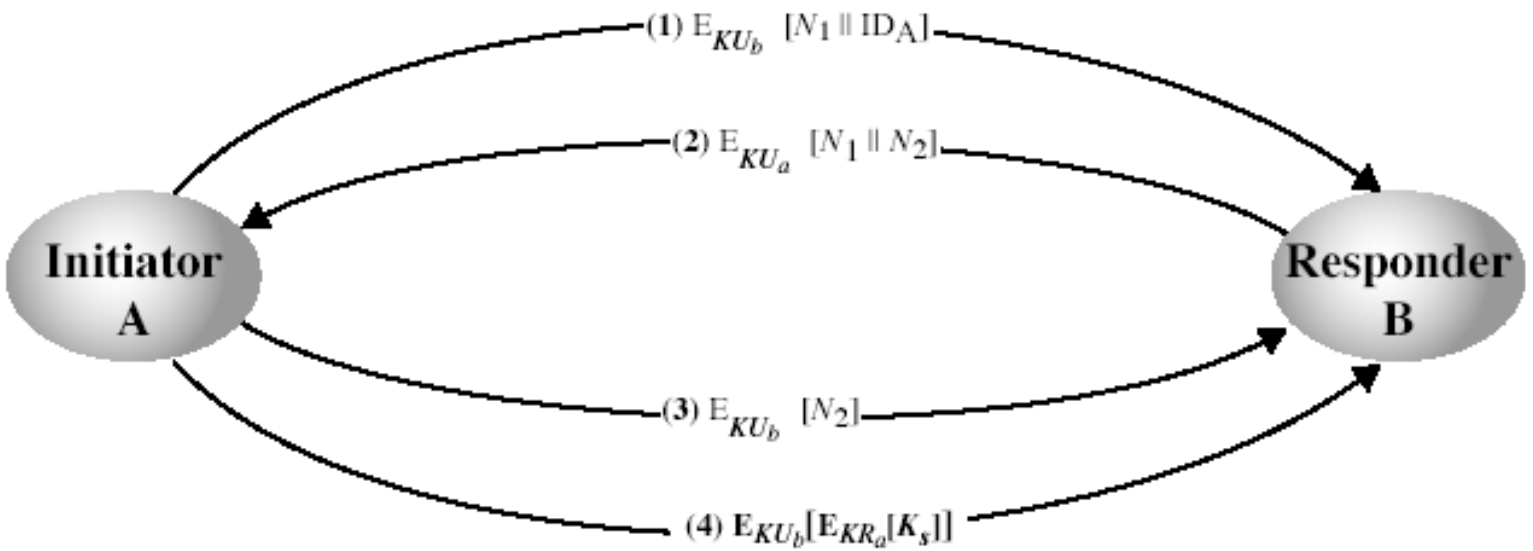
- public-key algorithms are slow
- so usually want to use private-key encryption to protect message contents
- hence need a session key
- have several alternatives for negotiating a suitable session using public-key

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
  - The scenario

Public-Key Distribution of Secret Keys

- First securely exchanged public-keys using a previous method



Diffie-Hellman Key Exchange

- first public-key type scheme proposed
  - For key distribution only
- 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**: $y_A = \alpha^{x_A} \bmod q$
- each user makes public that key y_A

Diffie-Hellman Key Exchange

- shared session key for users A & B is K:

  $K = y_A^{x_B} \bmod q$ (which **B** can compute)

  $K = y_B^{x_A} \bmod q$ (which **A** can compute)

  (example)

- K 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 $\alpha=3$
- select random secret keys:
  - A chooses $x_A=97$, B chooses $x_B=233$
- compute 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)

Elliptic Curve Cryptography

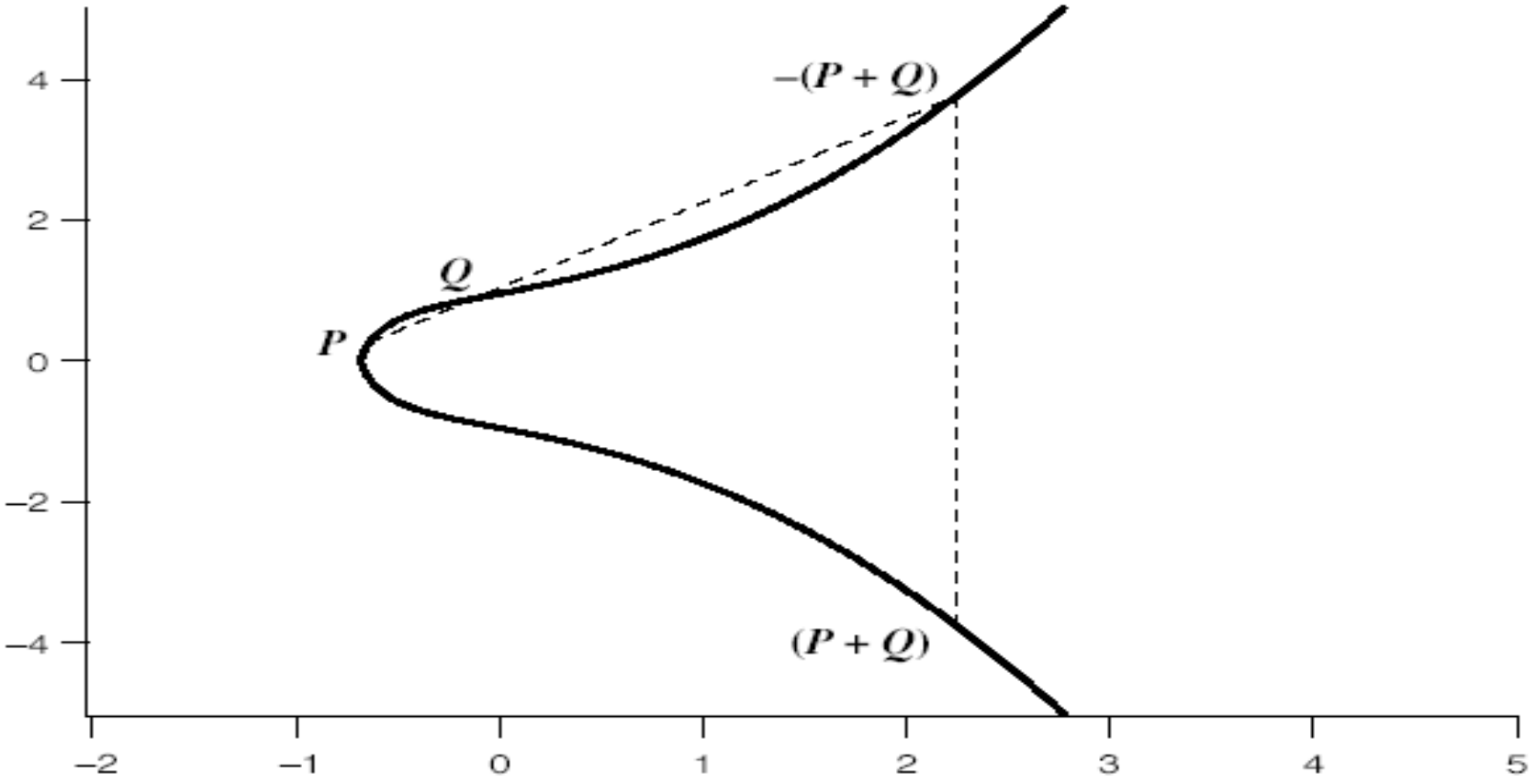
- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes

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
  - Q+R is reflection of intersection R
  - Closed form for additions
    - (10.3) and (10.4) P.300-301

Real Elliptic Addition

Rule 1-5 in P.300



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

Finite Elliptic Curves

- Elliptic curve cryptography uses curves whose variables & coefficients are finite integers
- have two families commonly used:
  - prime curves $E_p(a,b)$ defined over Z_p
    - $y^2 \bmod p = (x^3+ax+b) \bmod p$
    - use integers modulo a prime for both variables and coeff
    - best in software
  - Closed form of additions: P.303
  - Example: P=(3,10), Q=(9,7), in $E_{23}(1,1)$
    - P+Q = (17,20)
    - 2P = (7,12)

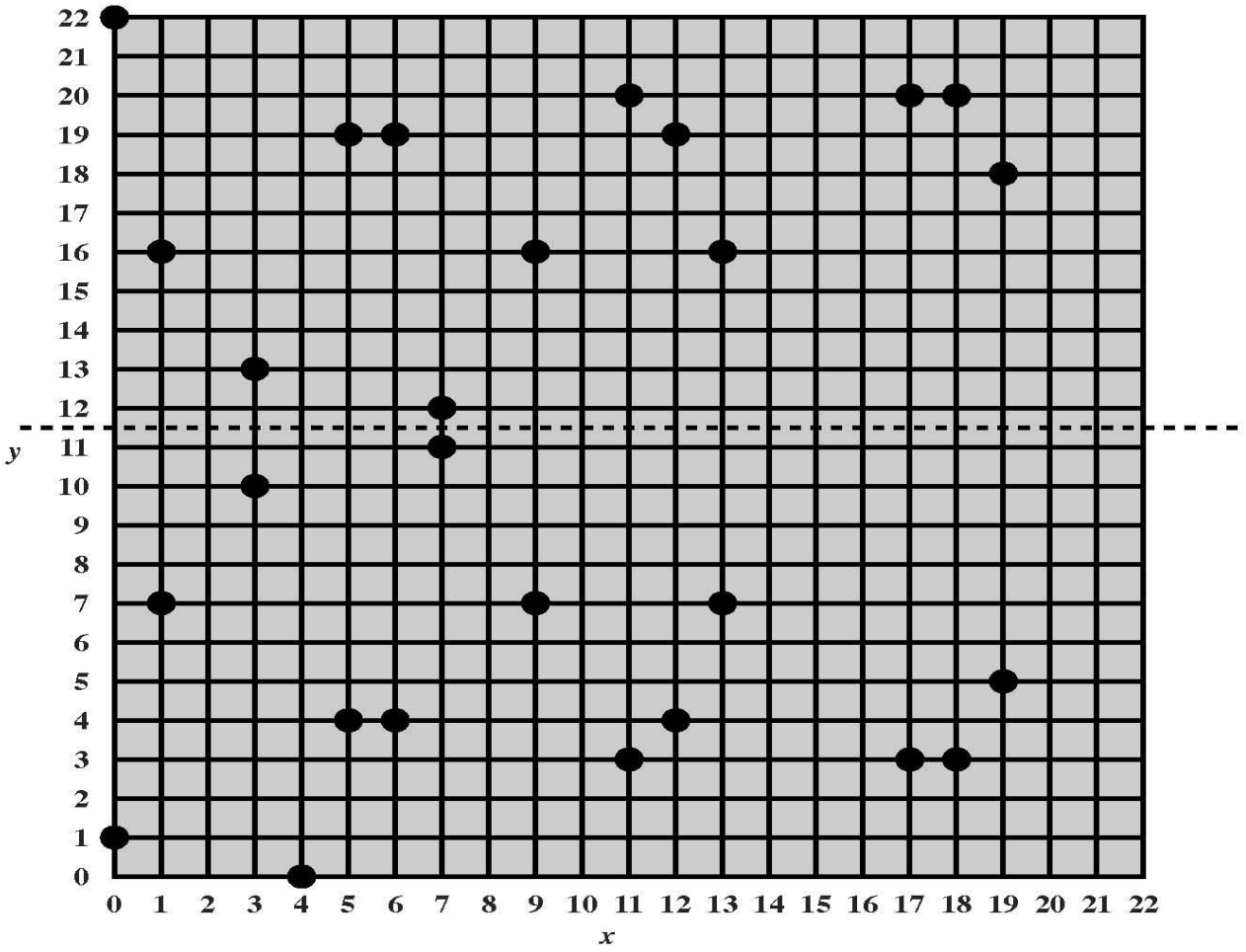


Figure 10.10 The Elliptic Curve $E_{23}(1,1)$

Finite Elliptic Curves

- have two families commonly used:
  - binary curves $E_{2m}(a,b)$ defined over GF(2^m)
    - use polynomials with binary coefficients
    - best in hardware
  - Take a slightly different form of the equation
  - Different close forms for addition (P.304)

Elliptic Curve Cryptography

- ECC addition is analog of multiply
- ECC repeated addition is analog of exponentiation
- need “hard” problem equiv to discrete log
  - $Q=kP$, where Q,P are points in an elliptic 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: $E_{23}(9,17)$ (P.305)
  - k could be so large as to make brute-force fail

ECC Key Exchange

- can do key exchange similar to D-H
- users select a suitable curve $E_p(a,b)$
  - Either a prime curve, or a binary curve
- 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 $K=n_A \times n_B \times G$
- Example: P.305

ECC Encryption/Decryption

- select suitable curve & point G as in D-H
- encode any message M as a point on the elliptic curve $P_m=(x,y)$
- each user chooses private key $n_A<n$
- and computes public key $P_A=n_A \times G$
- to encrypt pick random k: $C_m=\{kG,\ P_m+k\ P_b\}$,
- decrypt C_m compute:

$$P_m+kP_b-n_B(kG) = P_m+k(n_BG)-n_B(kG) = P_m$$

- Example: P.307

Table 10.2 Computational Effort for Cryptanalysis of Elliptic Curve Cryptography Compared to RSA

Key Size	MIPS-Years
150	3.8×10^{10}
205	7.1×10^{18}
234	1.6×10^{28}

(a) Elliptic Curve Logarithms using the Pollard rho Method

Key Size	MIPS-Years
512	3×10^{4}
768	2×10^{8}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

(b) Integer Factorization using the General Number Field Sieve

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