MSC-CS

CRYPTOGRAPHY
&
NETWORK SECURITY

UNIT - 4

Unit-4: Other Public – Key Cryptosystems:

Hellman Key Exchange- Elgamal Cryptographic System- Elliptic Curve Arithmetic – Elliptic Curve Cryptography

Digital Signatures:

Digital Signatures- Elgamal Digital Signature Scheme – Schnorr Digital Signature Scheme- NIST Digital Signature Algorithm – Elliptic Curve Digital Signature Algorithm

Diffie-Hellman Key Exchange

- first public-key type scheme proposed
- by Diffie & Hellman in 1976 along with the exposition of public key concepts
- 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

Discrete Logs

Given $$b = a^x \pmod q$$

Find x

We denote this as $x = Log_a(b) \pmod q$

Why is this 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

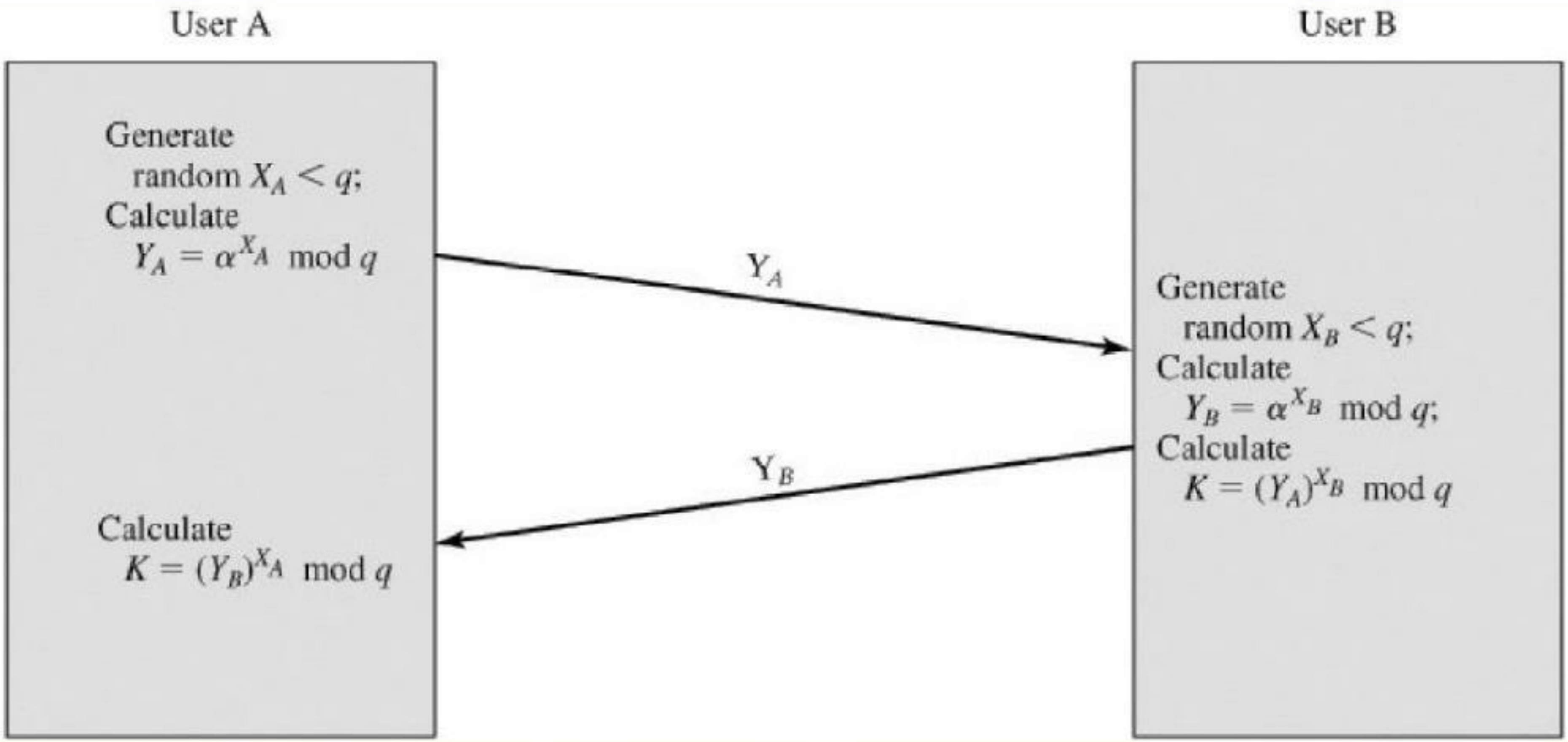
- shared session key for users A & B is K_{AB}:

  $K_{AB} = a^{x_A . x_B} \bmod q$

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

  $= y_B^{x_A} \bmod q$ (which **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


User A
User B
Generate random $X_A < q$;
Calculate $Y_A = \alpha^{X_A} \bmod q$
Y_A
Generate random $X_B < q$;
Calculate $Y_B = \alpha^{X_B} \bmod q$;
Calculate $K = (Y_A)^{X_B} \bmod q$
Y_B
Calculate $K = (Y_B)^{X_A} \bmod q$

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

Man-in-the-middle attack on Diffie-Hellman

1. Darth prepares for the attack by generating two random private keys X_{D1} *and* X_{D2} *and then computing the corresponding* public keys $Y_{D1} = a^{X_{D1}} \bmod q$ and $Y_{D2} = a^{X_{D2}} \bmod q$
2. Alice transmits Y_A *to Bob.*
3. Darth intercepts Y_A *but transmits* Y_{D1} *to Bob. Darth also calculates* $K2 = (Y_A)^{X_{D2}} \bmod q$.
4. Bob receives Y_{D1} *and calculates* $K1 = (Y_{D1})^{X_B} \bmod q$.
5. Bob transmits Y_B *to Alice.*
6. Darth intercepts Y_B *but transmits* Y_{D2} *to Alice. Darth calculates* $K1 = (Y_B)^{X_{D1}} \bmod q$.
7. Alice receives Y_{D2} *and calculates* $K2 = (Y_{D2})^{X_A} \bmod q$.

- *Allice and Bob think they share a secret key, but actually Bob and Darth share K1, and Alice and Darth share K2.*

ElGamal Cryptosystem

- Another public-key cryptosystem like RSA.
- Bob publishes (α, p, β), where 1 < m < p and $\beta=\alpha^a$
- Alice chooses secret k, computes and sends to Bob the pair (r,t) where
  - $r=\alpha^k$ (mod p)
  - $t = \beta^k m$ (mod p)
- Bob calculates: $tr^{-a}=m$ (mod p)
- Why does this decrypt?

ElGamal Cryptosystem

Bob publishes (α, p, β), where 1 < m < p and $\beta=\alpha^a$

Alice chooses secret k, computes and sends to Bob the pair (r,t) where

- $r=\alpha^k$ (mod p)
- $t = \beta^k m$ (mod p)

Bob finds: $tr^{-a}=m$ (mod p)

- Why does this work?

- Multiplying m by β^k scrambles it.
- Eve sees α, p, β, r, t. If she only knew a or k!
  - Knowing a allows decryption.
  - Knowing k also allows decryption. Why?
- Can't find k from r or t. Why?

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
- newer, but not as well analysed

Abelian Group

- A set of elements G and operation * among elements (G,*) with some
- Axioms:
  - (A1) Closure: □ a,b □ G, a*b □ G
  - (A2) associative law: (a*b)*c = a*(b*c)
  - (A3) has identity e: e*a = a*e = a
  - (A4) has inverses a^{-1}: $a*a^{-1} = e$
  - (A5) commutative law a*b = b*a

Operations

- If the operation * is x, and we perform all operations mod q,

$$\underbrace{a \times a \times \ldots \times a}_{k} = a^k \bmod q$$

- If the operation * is +,

$$\underbrace{a + a + \ldots + a}_{k} = k\,a$$

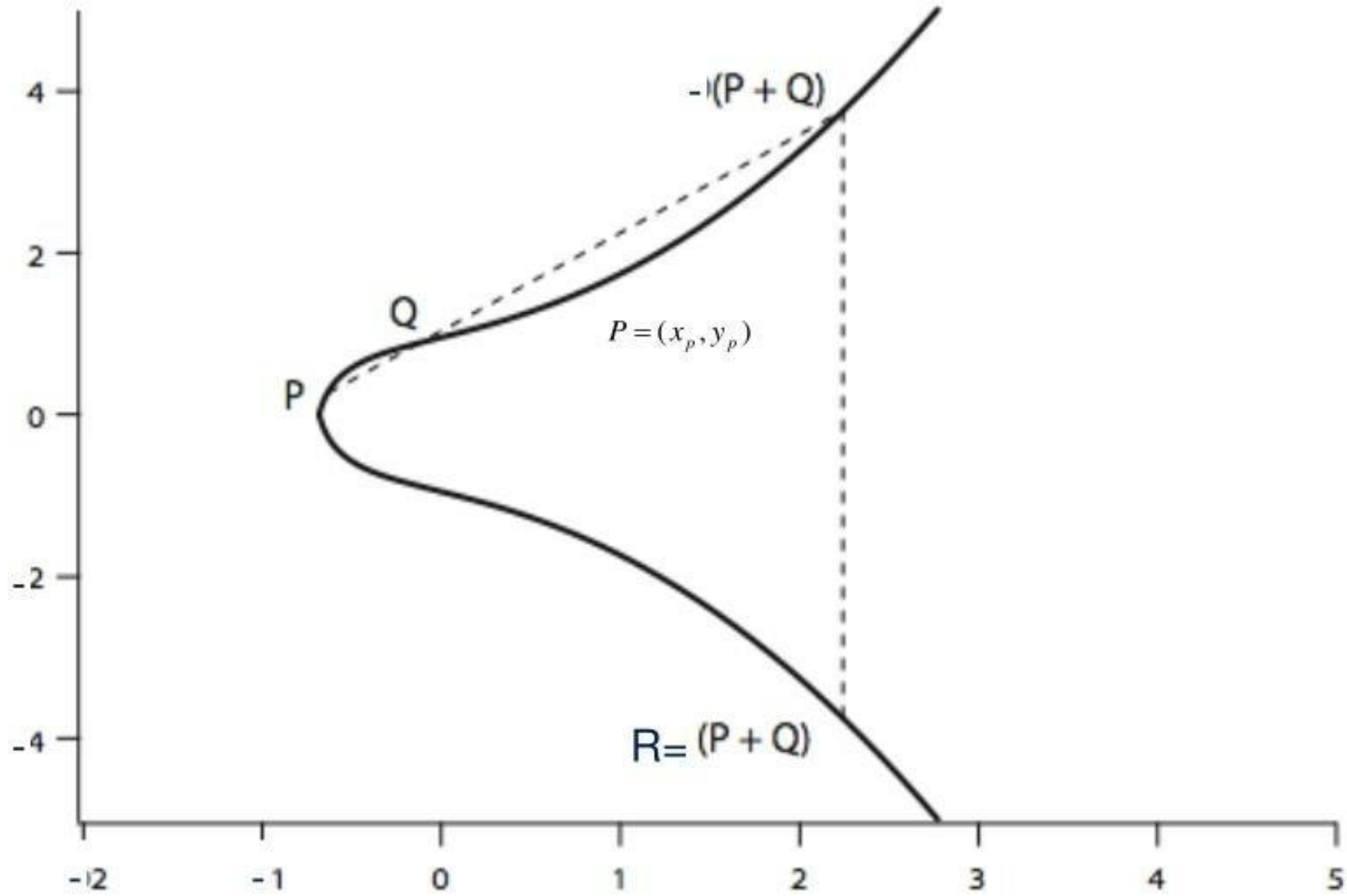
Real Elliptic Curves

- 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
  - Note: More general form of the elliptical curve (Weierstrass equation):

    $$y^2 + axy + by = x^3 + cx^2 + dx + e$$

    can be transformed to the form: $y^2 = x^3 + ax + b$
- have addition operation for elliptic curve
  - geometrically sum of Q+R is reflection of intersection R

Real Elliptic Curve Example



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

Elliptic Curve Cryptography

$$P = (x_P, y_P), \quad Q = (x_Q, y_Q)$$

Slope of the line between P and Q

$$\Delta = \frac{y_Q - y_P}{x_Q - x_P}$$

$$R = P + Q$$

$$R = (x_R, y_R)$$

$$x_R = \Delta^2 - x_P - x_Q$$

$$y_R = -y_P + \Delta(x_P - x_R)$$

$$R = P + P = 2P, \quad y_P \neq 0$$

$$R = (x_R, y_R)$$

$$x_R = \left(\frac{3x_P^2 + a}{2y_P}\right)^2 - 2x_P$$

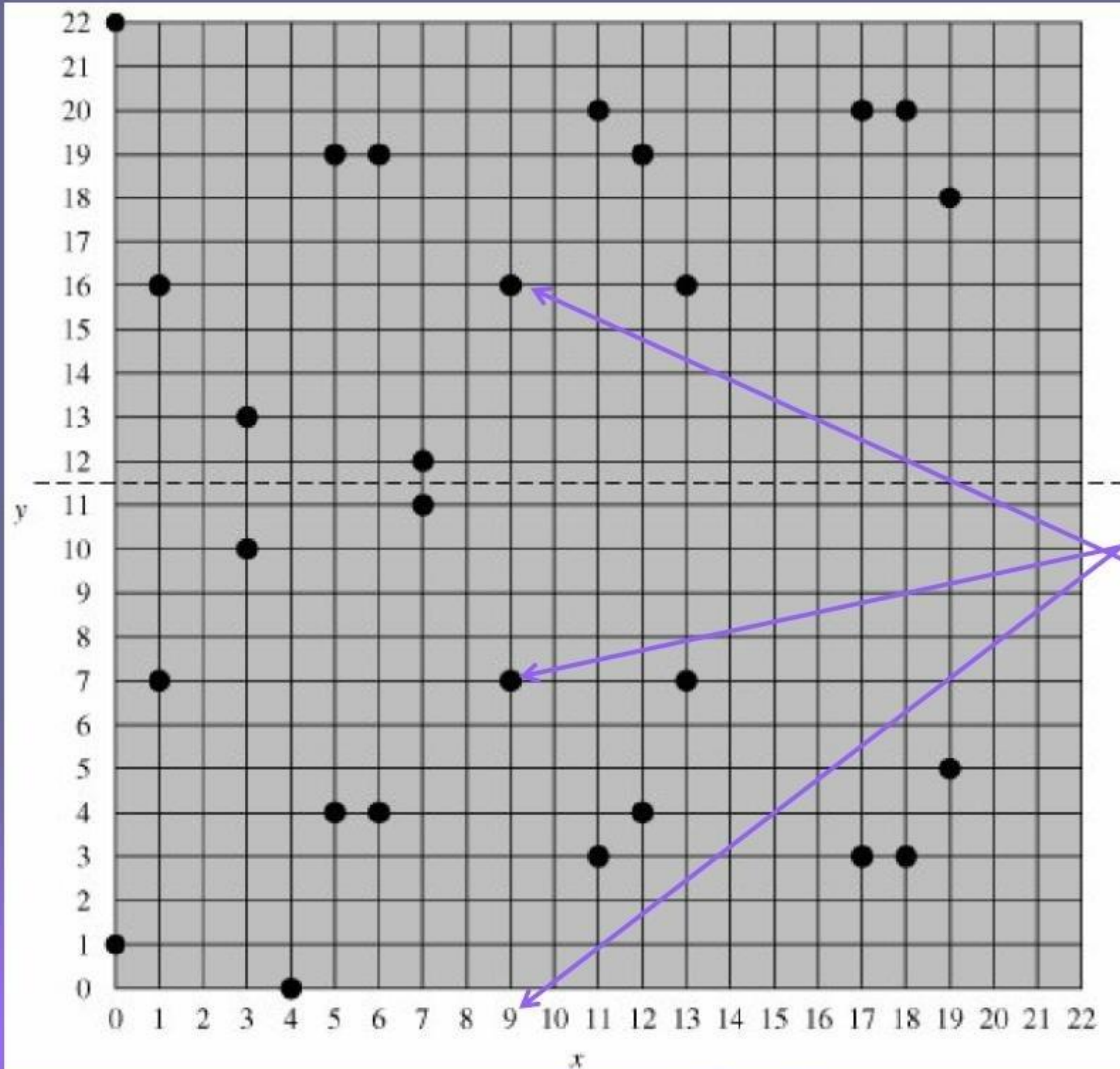
$$y_R = \left(\frac{3x_P^2 + a}{2y_P}\right)(x_P - x_R) - y_P$$

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 = P+P+...+P`, 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

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



Certicom example:
$E_{23}(1,1)$

All operations work mod 23.

Not all values for x and y satisfy
$y^2 = x^3 + x + 1$

For x=9, only
y=7
and
y=16

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: $P_A=n_AG$, $P_B=n_BG$
- compute shared key: $K=n_AP_B$, $K=n_BP_A$
  - same since $K=n_An_BG$

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:

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

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

Comparable Key Sizes for Equivalent Security

Symmetric scheme (key size in bits)	ECC-based scheme (size of *n* in bits)	RSA/DSA (modulus size in bits)
56	112	512
80	160	1024
112	224	2048
128	256	3072
192	384	7680
256	512	15360

Legal use of ECC

- A lot of ECC techniques are patented by the company Certicom
- NSA bought some patents from them and made the royalty free via NIST standardisation