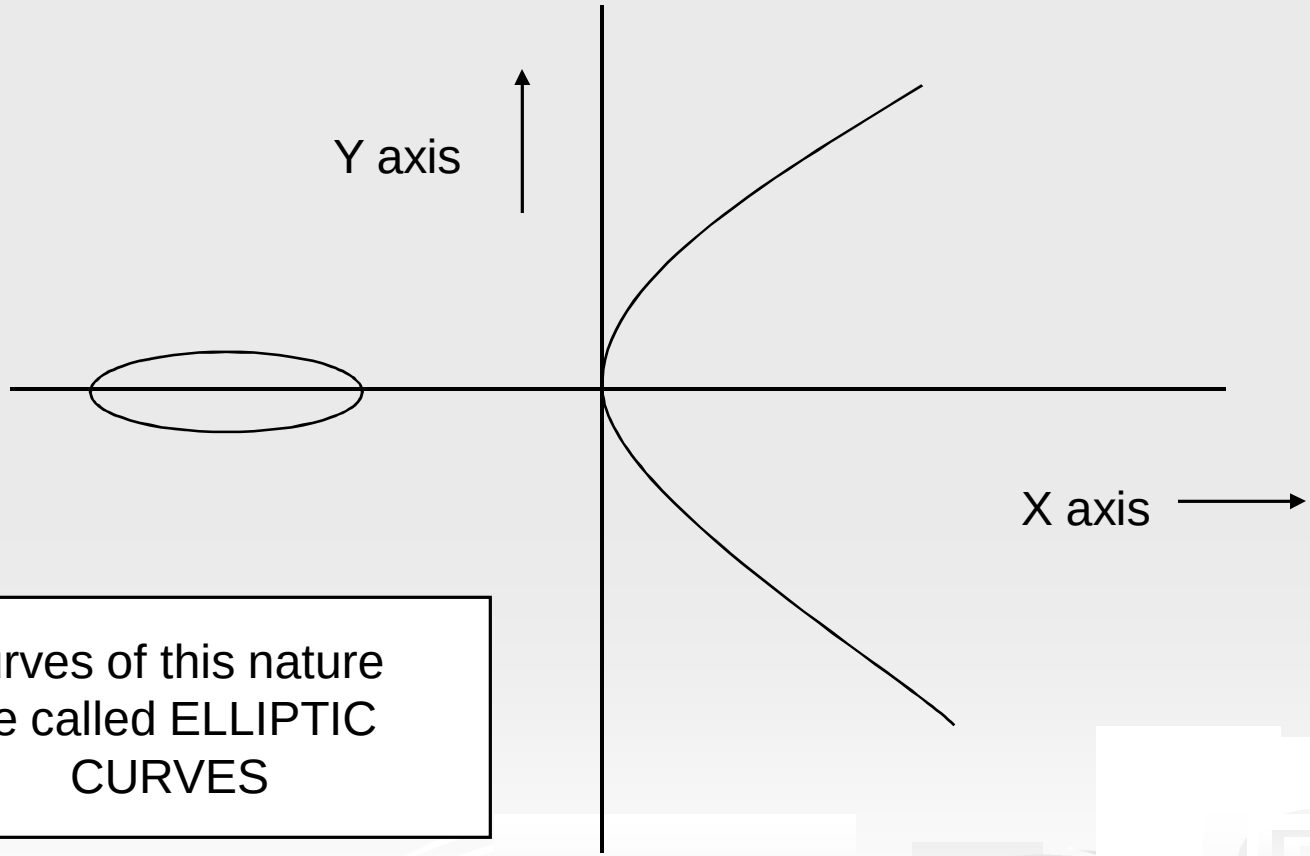
Introduction to Elliptic Curves

Graphical Representation

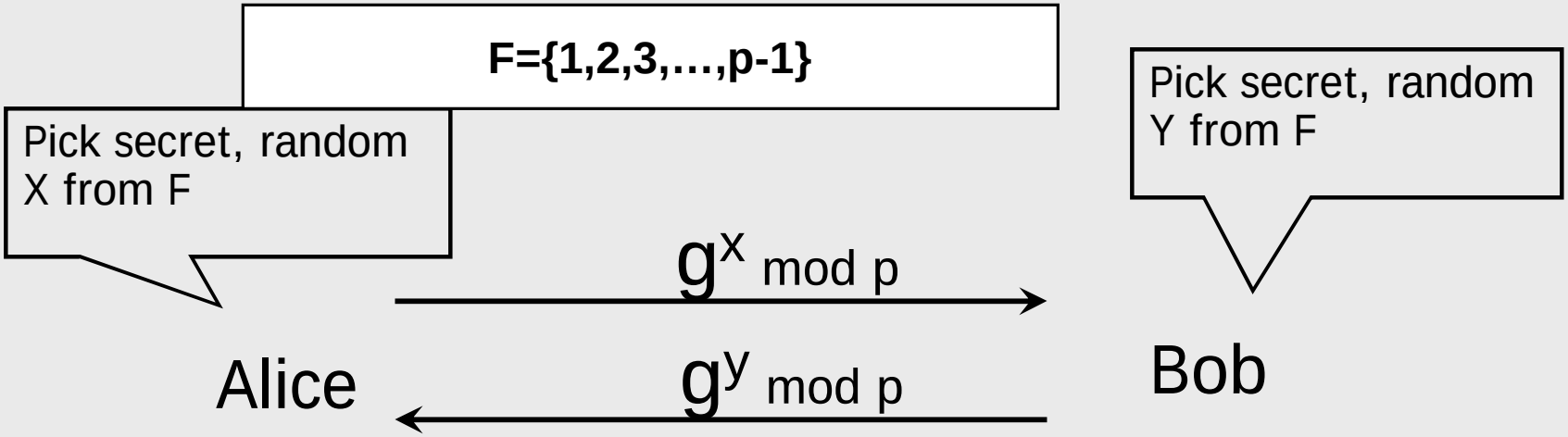


Curves of this nature are called ELLIPTIC CURVES

Elliptic curves in Cryptography

- Elliptic Curve (EC) systems as applied to cryptography were first proposed in 1985 independently by Neal Koblitz and Victor Miller.
- The discrete logarithm problem on elliptic curve groups is believed to be more difficult than the corresponding problem in (the multiplicative group of nonzero elements of) the underlying finite field.

Discrete Logarithms in Finite Fields



Compute $k=(g^y)^x=g^{xy} \bmod p$

Compute $k=(g^x)^y=g^{xy} \bmod p$

Eve has to compute g^{xy} from g^x and g^y without knowing x and y…
She faces the Discrete Logarithm Problem in finite fields

Diffie-Hellman Setup

- all users agree on global parameters:
  - large prime integer or polynomial q
  - a being a primitive root mod q
- Alice: generates her keys
  - chooses a secret key (number): $x_A < q$
  - computes **public key**: $y_A = a^{x_A} \bmod q$
  - Sends y_A to Bob
- Bob: generates his keys
  - chooses a secret key (number): $x_B < q$
  - computes **public key**: $y_B = a^{x_B} \bmod q$
  - Sends y_B to Bob

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

Diffie-Hellman Example

- users Alice & Bob who wish to swap keys:
- agree on prime q=353 and a=3 (these are public values).
- 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)

Elliptic Curve Cryptography

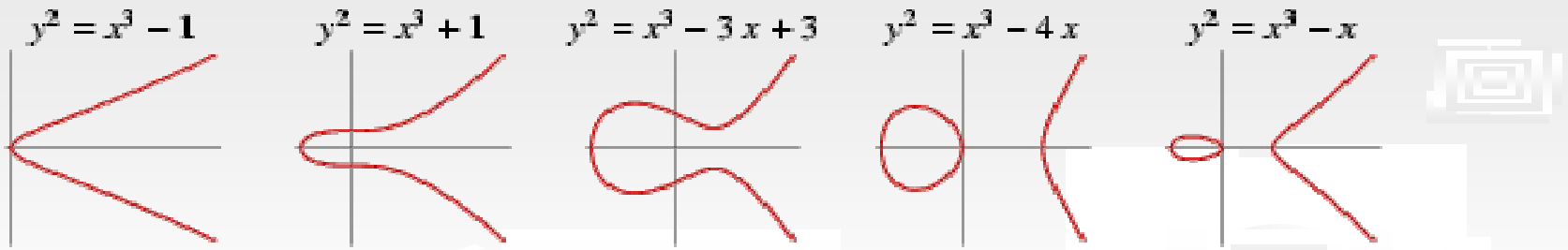
- majority of public-key crypto (RSA, D-H) use either integer or polynomial arithmetic with very large numbers/polynomials
- imposes a significant load in storing and processing keys and messages
- an alternative is to use elliptic curves
- offers same security with smaller bit sizes
- newer, but not as well analysed

General form of a EC

- An *elliptic curve* is a plane curve defined by an equation of the form

$$y^2 = x^3 + ax + b$$

Examples

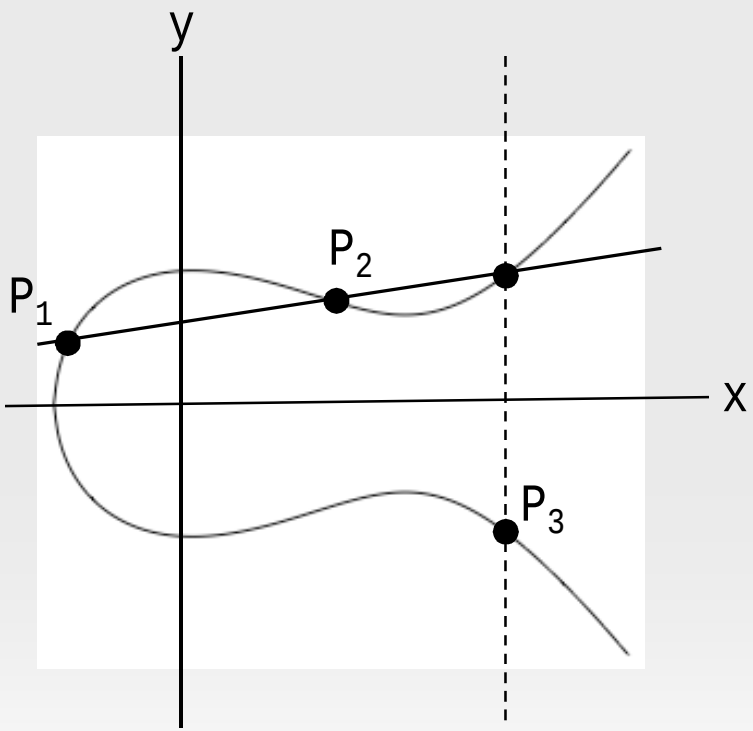


The Abelian Group

Given two points P,Q in *E(Fp)*, there is a third point, denoted by *P*+*Q* on *E(Fp)*, and the following relations hold for all P,Q,R in *E(Fp)*

- $P + Q = Q + P$ (*commutativity*)
- $(P + Q) + R = P + (Q + R)$ (*associativity*)
- $P + O = O + P = P$ (*existence of an identity element*)
- there exists $(-P)$ such that $-P + P = P + (-P) = O$ (*existence of inverses*)

Elliptic Curve Picture



- Consider elliptic curve

  $E: y^2 = x^3 - x + 1$

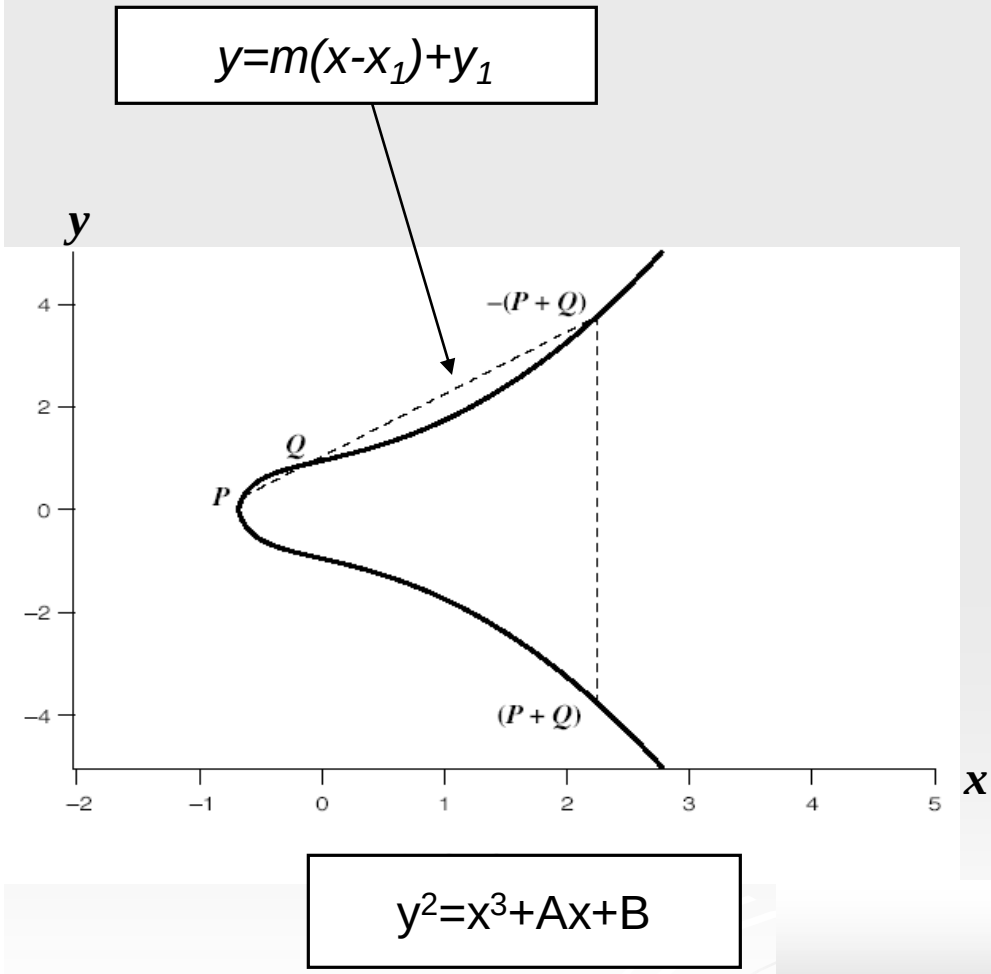
- If P_1 and P_2 are on E, we can define

  $P_3 = P_1 + P_2$

  as shown in picture

- Addition is all we need

Addition in Affine Co-ordinates



$$P = (x_1, y_1), Q = (x_2, y_2)$$

$$R = (P + Q) = (x_3, y_3)$$

Let, P≠Q,

$$m = \frac{y_2 - y_1}{x_2 - x_1};$$

To find the intersection with E. we get

$$(m(x - x_1) + y_1)^2 = x^3 + Ax + B$$

$$or, 0 = x^3 - m^2 x^2 + \ldots$$

$$So, x_3 = m^2 - x_1 - x_2$$

$$\Rightarrow y_3 = m(x_1 - x_2) - y_1$$

Doubling of a point

- Let, P=Q

$$2y\frac{dy}{dx} = 3x^2 + A$$

$$\Rightarrow m = \frac{dy}{dx} = \frac{3x_1^2 + A}{2y_1}$$

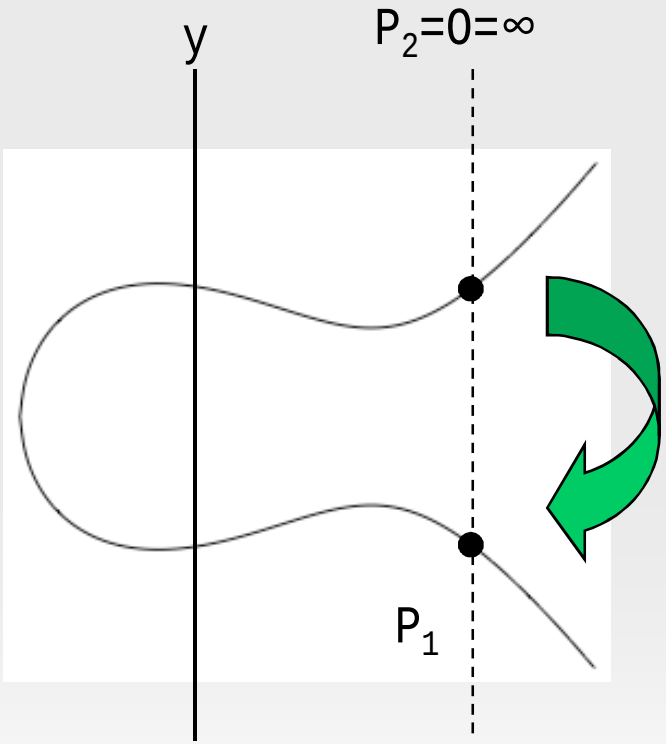
$$If\ , y_1 \neq 0 \text{ (since then } P_1 + P_2 = \infty\text{):}$$

$$\therefore 0 = x^3 - m^2x^2 + \ldots$$

$$\Rightarrow x_3 = m^2 - 2x_1, y_3 = m(x_1 - x_3) - y_1$$

- What happens when $P_2=\infty$?

Why do we need the reflection?



$$P_1=P_1+O=P_1$$

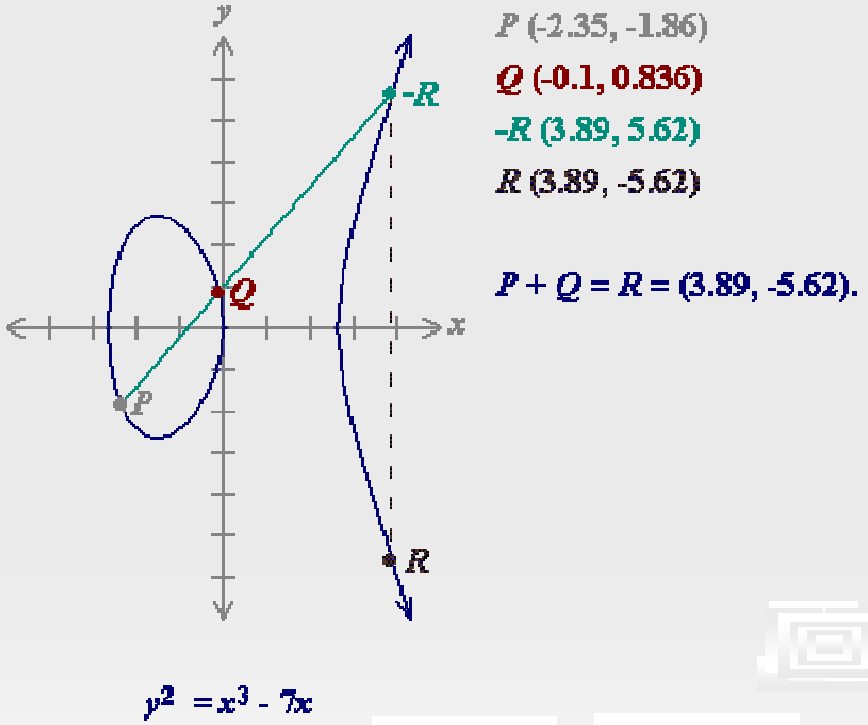
Sum of two points

Define for two points $P(x_1,y_1)$ and $Q(x_2,y_2)$ in the Elliptic curve

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1} \; for \; x_1 \neq x_2 \\ \dfrac{3x_1^{\,2} + a}{2y_1} \; for \; x_1 = x_2 \end{cases}$$

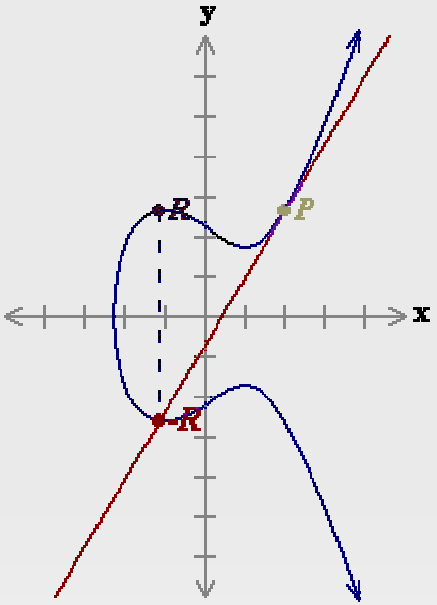
Then $P+Q$ is given by $R(x_3,y_3)$:

$$\boxed{\begin{aligned} x_3 &= \lambda - x_1 - x_2 \\ y_3 &= \lambda(x_3 - x_1) + y_1 \end{aligned}}$$



Point at infinity ***O***

P+P = 2P



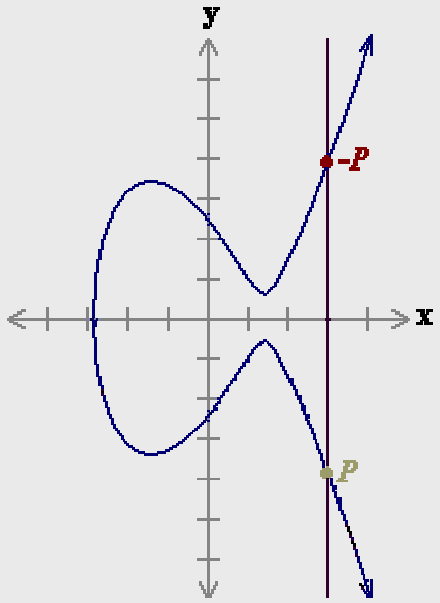
P (2, 2.65)

$-R$ (-1.11, -2.64)

R (-1.11, 2.64)

$2P = R = (-1.11, 2.64)$.

$y^2 = x^3 - 3x + 5$



$P + (-P) = O$

$y^2 = x^3 - 6x + 6$

As a result of the above case ***P=O+P***

***O* is called the additive identity of the elliptic curve group.**

Hence all elliptic curves have an additive identity ***O***.

ECC over Finite Field Fp

P = (xP, yP) is the point -P = (xP, -yP mod p).

- Adding two Distinct Points

  If P and Q are distinct points such that P is not -Q, then

  P + Q = R where

  s = (yP - yQ) / (xP - xQ) mod p

  xR = s2 - xP - xQ mod p and yR = -yP + s(xP - xR) mod p

  Note that s is the slope of the line through P and Q.

ECC over Finite Field Fp

- Doubling of a point

  Provided that yP is not 0,

  2P = R where

  $s = (3xP2 + a) / (2yP) \bmod p$

  $xR = s2 - 2xP \bmod p$ and $yR = -yP + s(xP - xR) \bmod p$

ECC over $F2^m$

- Elements of the field F_2m are m-bit strings
- The equation is adjusted to
  - $y^2 + xy = x^3 + ax^2 + b$
- here, only condition is: b is not zero
- For example, consider the field F_2^4, defined by using polynomial representation with the irreducible polynomial $f(x) = x^4 + x + 1$.

ECC over F2m cont…

- Adding Two distinct points P and Q
  The negative of the point P = (xP, yP) is the point
  -P = (xP, xP + yP)

  If P and Q are distinct points such that P is not -Q, then

  P + Q = R where

  s = (yP - yQ) / (xP + xQ)

  xR = s2 + s + xP + xQ + a and yR = s(xP + xR) + xR + yP

- **Doubling the point P**

  If xP = 0, then 2P = O

  Provided that xP is not 0,

  2P = R where

  s = xP + yP / xP

  xR = s2+ s + a and yR = xP2 + (s + 1) * xR

What Is ECC?

- Elliptic curve cryptography [ECC] is a **public-key** cryptosystem just like RSA, Rabin, and El Gamal.
- Every user has a **public** and a **private** key.
  - Public key is used for encryption/signature verification.
  - Private key is used for decryption/signature generation.
- Elliptic curves are used as an extension to other current cryptosystems.
  - Elliptic Curve Diffie-Hellman Key Exchange
  - Elliptic Curve Digital Signature Algorithm

Using Elliptic Curves In Cryptography

- The central part of any cryptosystem involving elliptic curves is the **elliptic group**.
- All public-key cryptosystems have some underlying mathematical operation.
  - RSA has exponentiation (raising the message or ciphertext to the public or private values)
  - ECC has point multiplication (repeated addition of two points).

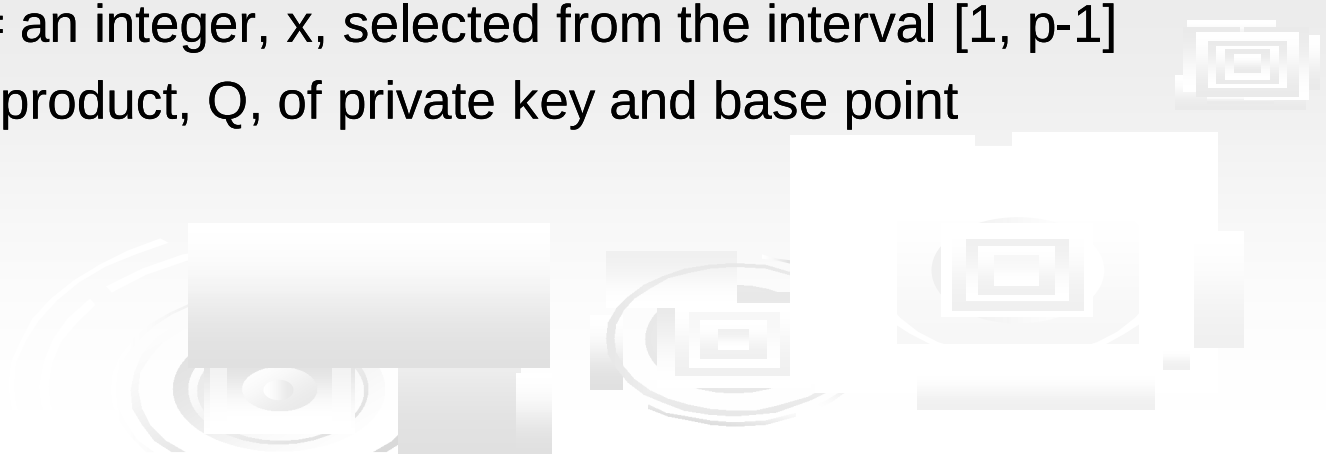
Example Repeated Addition

- Let k be a scalar that is multiplied with the point P to obtain another point Q on the curve. i.e. to find Q = kP.
- If k = 23 then

$$kP = 23.P = 2(2(2(2P) + P) + P) + P.$$

Generic Procedures of ECC

- Both parties agree to some publicly-known data items
  - The **elliptic curve equation**
    - values of ***a*** and ***b***
    - prime, ***p***
  - The **elliptic group** computed from the elliptic curve equation
  - A **base point**, B, taken from the elliptic group
    - Similar to the generator used in current cryptosystems
- Each user generates their public/private key pair
  - Private Key = an integer, x, selected from the interval [1, p-1]
  - Public Key = product, Q, of private key and base point
    - (Q = x*B)



Example – Elliptic Curve Cryptosystem

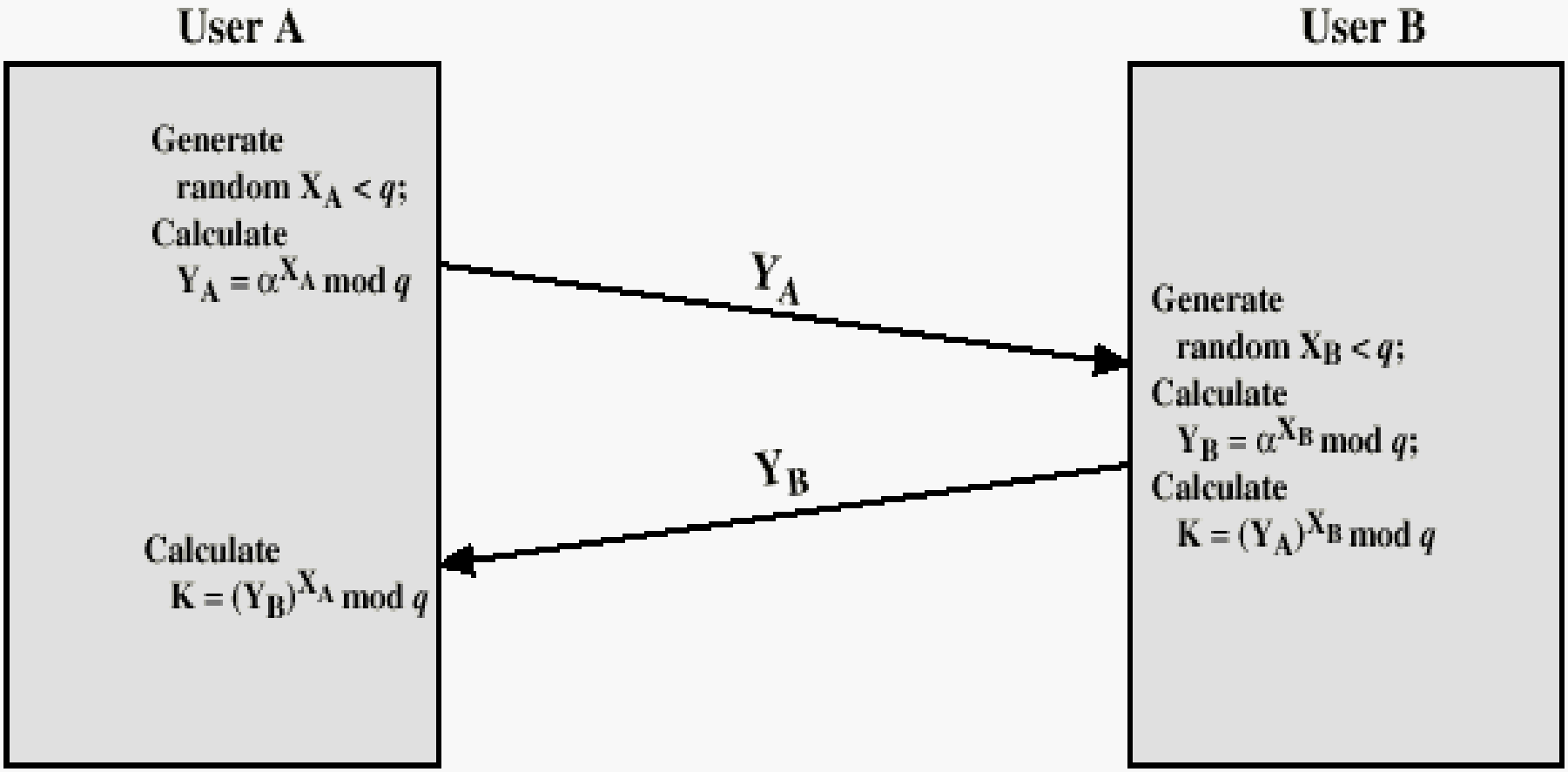
- Suppose Alice wants to send to Bob an encrypted message.
  - Both agree on a base point, B.
  - Alice and Bob create public/private keys.
    - Alice
      - Private Key = a
      - Public Key = $P_A = a * B$
    - Bob
      - Private Key = b
      - Public Key = $P_B = b * B$
  - Alice takes plaintext message, M, and encodes it onto a point, P_M, from the elliptic group

Example – cont…

- Alice chooses another random integer, k from the interval [1, p-1]
- The ciphertext is a pair of points
  - $P_C = [(kB), (P_M + kP_B)]$

- To decrypt, Bob computes the product of the first point from P_C and his private key, b
  - $b * (kB)$
- Bob then takes this product and subtracts it from the second point from P_C
  - $(P_M + kP_B) – [b(kB)] = P_M + k(bB) – b(kB) = P_M$
- Bob then decodes P_M to get the message, M.

Diffie-Hellman (DH) Key Exchange



ECC Diffie-Hellman

- **Public:** Elliptic curve and point B=(x,y) on curve
- **Secret:** Alice's a and Bob's b

a(x,y)

b(x,y)

Alice, A

Bob, B

- Alice computes a(b(x,y))
- Bob computes b(a(x,y))
- These are the same since ab = ba

Example – Elliptic Curve Diffie-Hellman Exchange

- Alice and Bob want to agree on a shared key.
  - Alice and Bob compute their public and private keys.
    - Alice
      - Private Key = a
      - Public Key = $P_A = a * B$
    - Bob
      - Private Key = b
      - Public Key = $P_B = b * B$
  - Alice and Bob send each other their public keys.
  - Both take the product of their private key and the other user's public key.
    - Alice → $K_{AB} = a(bB)$
    - Bob → $K_{AB} = b(aB)$
    - **Shared Secret Key = $K_{AB} = abB$**

Why use ECC?

- How do we analyze Cryptosystems?
  - How difficult is the underlying problem that it is based upon
    - RSA – Integer Factorization
    - DH – Discrete Logarithms
    - ECC - Elliptic Curve Discrete Logarithm problem
  - How do we measure difficulty?
    - We examine the algorithms used to solve these problems

Security of ECC

- To **protect** a 128 bit AES key it would take a:
  - RSA Key Size: 3072 bits
  - ECC Key Size: 256 bits
- How do we strengthen RSA?
  - Increase the key length
- **Impractical?**

NIST guidelines for public key sizes for AES

ECC KEY SIZE (Bits)	RSA KEY SIZE (Bits)	KEY SIZE RATIO	AES KEY SIZE (Bits)
163	1024	1 : 6	
256	3072	1 : 12	128
384	7680	1 : 20	192
512	15 360	1 : 30	256

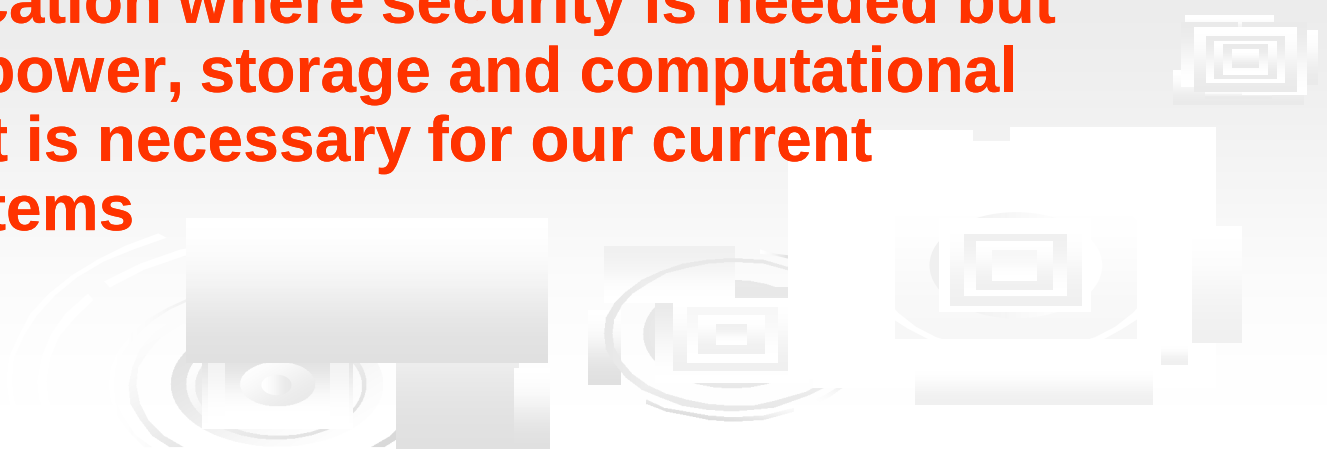
Supplied by NIST to ANSI X9F1

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

Applications of ECC

- Many devices are small and have limited storage and computational power
- Where can we apply ECC?
  - **Wireless communication devices**
  - Smart cards
  - Web servers that need to handle many encryption sessions
  - **Any application where security is needed but lacks the power, storage and computational power that is necessary for our current cryptosystems**



Benefits of ECC

- Same benefits of the other cryptosystems: confidentiality, integrity, authentication and non-repudiation but…
- Shorter key lengths
  - Encryption, Decryption and Signature Verification speed up
  - Storage and bandwidth savings

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

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

Summary of ECC

- "**Hard problem**" analogous to discrete log
  - **Q=kP, where Q, P belong to a prime curve**
    given k, P → "easy" to compute Q
    given Q, P → "hard" to find k
  - **known as the elliptic curve logarithm problem**
    - **k must be large enough**
- ECC security relies on elliptic curve logarithm problem
  - compared to factoring, can use much smaller key sizes than with RSA etc
    - **for similar security ECC offers significant computational advantages**