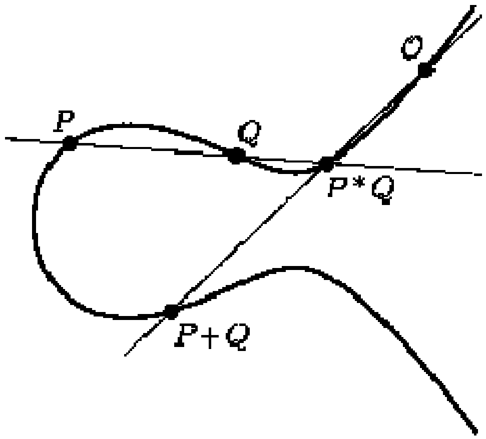



# Elliptic Curve Cryptography

**Speaker :** **Debdeep Mukhopadhyay**

**Dept of Computer Sc and Engg**

**IIT Madras**

# Outline of the Talk…

- **Introduction to Elliptic Curves**

- Elliptic Curve Cryptosystems (ECC)

- Implementation of ECC in Binary Fields

# Introduction to Elliptic Curves

# Lets start with a puzzle…

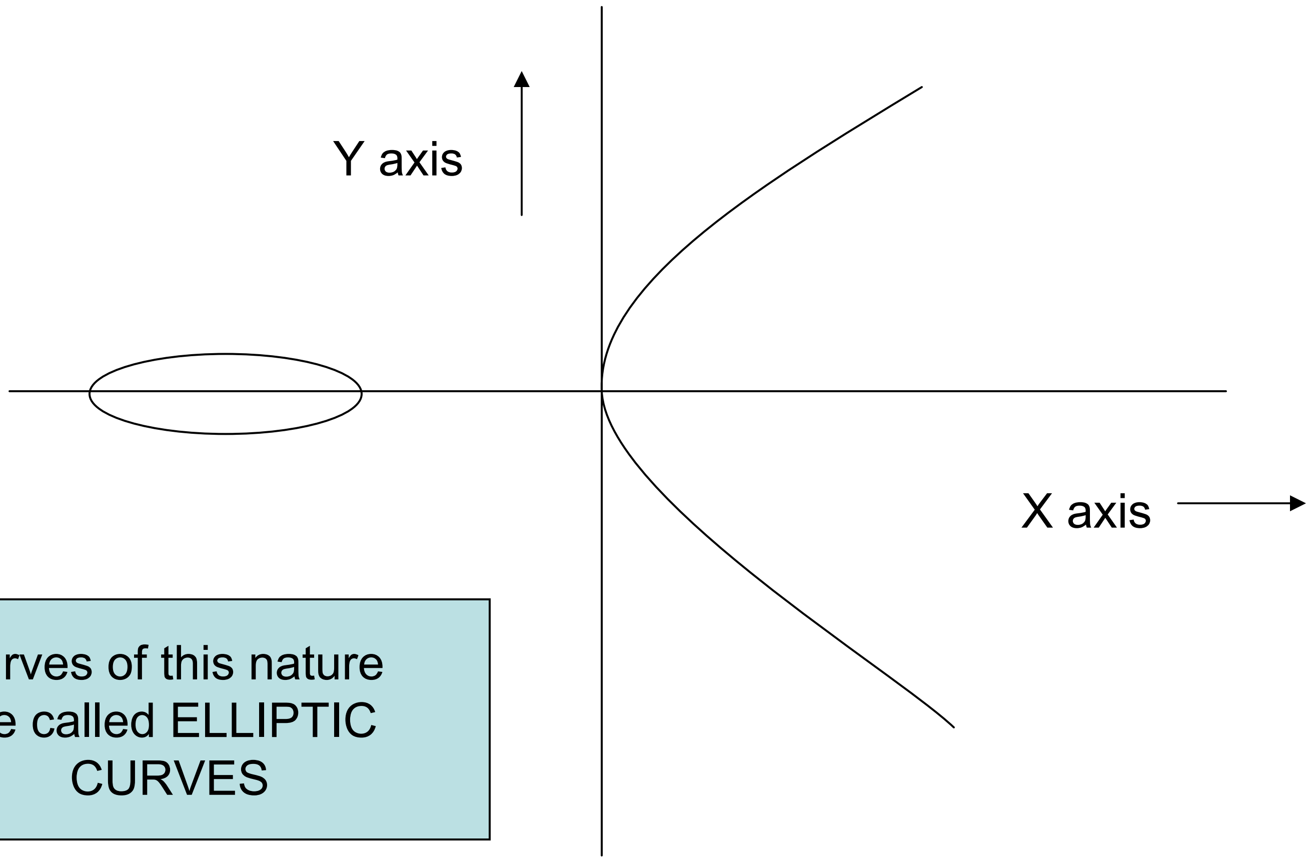
- **What is the number of balls that may be piled as a square pyramid and also rearranged into a square array?**
- **Soln:** Let x be the height of the pyramid…

Thus, $$1^2 + 2^2 + 3^2 + \ldots + x^2 = \frac{x(x+1)(2x+1)}{6}$$

We also want this to be a square:

Hence, $$\boxed{y^2 = \frac{x(x+1)(2x+1)}{6}}$$

# Graphical Representation




Curves of this nature are called ELLIPTIC CURVES

# Method of Diophantus

- Uses a set of known points to produce new points
- (0,0) and (1,1) are two trivial solutions
- Equation of line through these points is y=x.
- Intersecting with the curve and rearranging terms:

$$x^3 - \frac{3}{2}x^2 + \frac{1}{2}x = 0$$

- We know that 1 + 0 + x = 3/2 =>

  x = ½ and y = ½
- Using symmetry of the curve we also have (1/2,-1/2) as another solution

# Diophantus' Method

- Consider the line through (1/2,-1/2) and (1,1) => y=3x-2
- Intersecting with the curve we have:

$$x^3 - \frac{51}{2}x^2 + \ldots = 0$$

- Thus ½ + 1 + x = 51/2 or x = 24 and y=70

- Thus if we have 4900 balls we may arrange them in either way

# Elliptic curves in Cryptography

- Elliptic Curve (EC) systems as applied to cryptography were first proposed in 1985 independently by Neal Koblitz and Victor Miller.

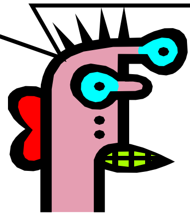
- The discrete logarithm problem on elliptic curve groups is believed to be more difficult than the corresponding problem in (the multiplicative group of nonzero elements of) the underlying finite field.

# Discrete Logarithms in Finite Fields

**F={1,2,3,…,p-1}**

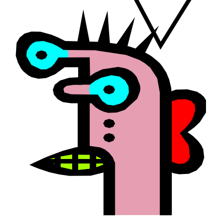
Pick secret, random X from F

Pick secret, random Y from F



$g^x$ mod p →

← $g^y$ mod p



Alice

Bob

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

Compute $k=(g^x)^y=g^{xy}$ mod 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

# Elliptic Curve on a finite set of Integers

- Consider $y^2 = x^3 + 2x + 3 \pmod 5$

  $x = 0 \Rightarrow y^2 = 3 \Rightarrow$ no solution (mod 5)

  $x = 1 \Rightarrow y^2 = 6 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 2 \Rightarrow y^2 = 15 = 0 \Rightarrow y = 0 \pmod 5$

  $x = 3 \Rightarrow y^2 = 36 = 1 \Rightarrow y = 1,4 \pmod 5$

  $x = 4 \Rightarrow y^2 = 75 = 0 \Rightarrow y = 0 \pmod 5$

- Then points on the elliptic curve are

  (1,1) (1,4) (2,0) (3,1) (3,4) (4,0)

  and the point at infinity: $\infty$

**Using the finite fields we can form an Elliptic Curve Group where we also have a DLP problem which is harder to solve…**

# Definition of Elliptic curves

- An elliptic curve over a field ***K*** is a nonsingular cubic curve in two variables, $f(x,y) = 0$ with a rational point (which may be a point at infinity).

- The field ***K*** is usually taken to be the complex numbers, reals, rationals, algebraic extensions of rationals, p-adic numbers, or a **finite field**.

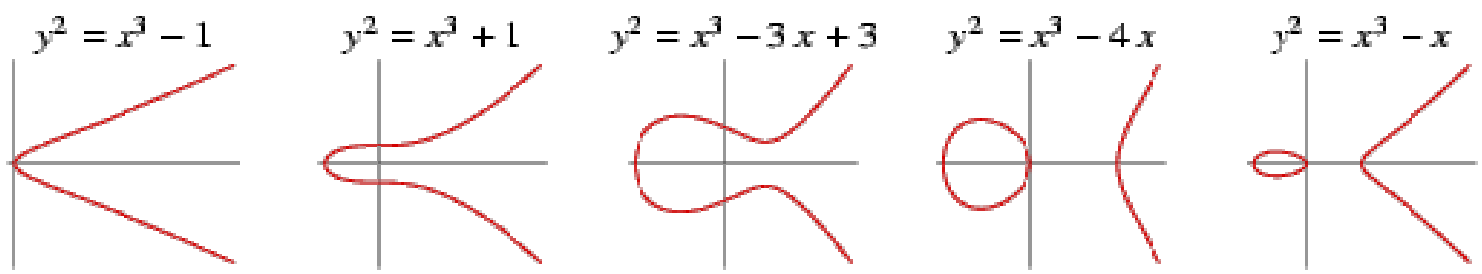
- Elliptic curves groups for cryptography are examined with the underlying fields of $\boldsymbol{F_p}$ (*where* $p>3$ *is a prime*) and $\boldsymbol{F_{2^m}}$ (*a binary representation with* $2^m$ *elements*).

# 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



# Weierstrass Equation

- A two variable equation F(x,y)=0, forms a curve in the plane. We are seeking geometric arithmetic methods to find solutions

- Generalized Weierstrass Equation of elliptic curves:

$$y^2 + a_1xy + a_3y = x^2 + a_2x^2 + a_4x + a_6$$

Here, A, B, x and y all belong to a field of say rational numbers, complex numbers, finite fields ($F_p$) or Galois Fields ($GF(2^n)$).

- **If Characteristic field is not 2:**

$$(y + \frac{a_1 x}{2} + \frac{a_3}{2})^2 = x^3 + (a_2 + \frac{a_1^2}{4})x^2 + a_4 x + (\frac{a_3^2}{4} + a_6)$$

$$\Rightarrow y_1^{\ 2} = x^3 + a_2^{'} x^2 + a_4^{'} x + a_6^{'}$$

- **If Characteristics of field is neither 2 nor 3:**

$$x_1 = x + a_2^{'} / 3$$

$$\Rightarrow y_1^{\ 2} = x_1^{\ 3} + A x_1 + B$$

# Points on the Elliptic Curve (EC)

- Elliptic Curve over field L

$$E(L) = \{\infty\} \cup \{(x, y) \in L \times L \mid y^2 + ... = x^3 + ...\}$$

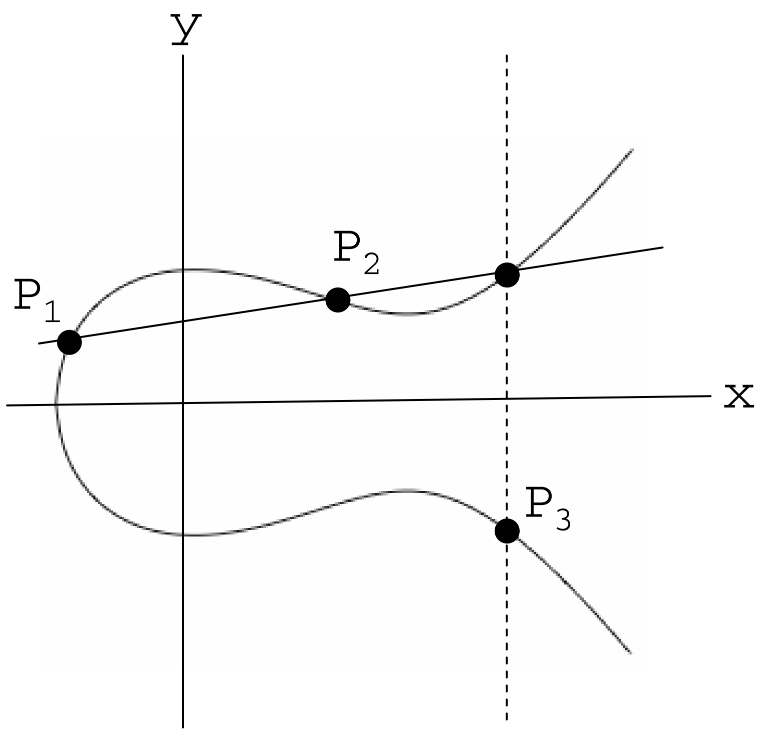
- It is useful to add the point at infinity
- The point is sitting at the top of the y-axis and any line is said to pass through the point when it is vertical
- It is both the top and at the bottom of the y-axis

# 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

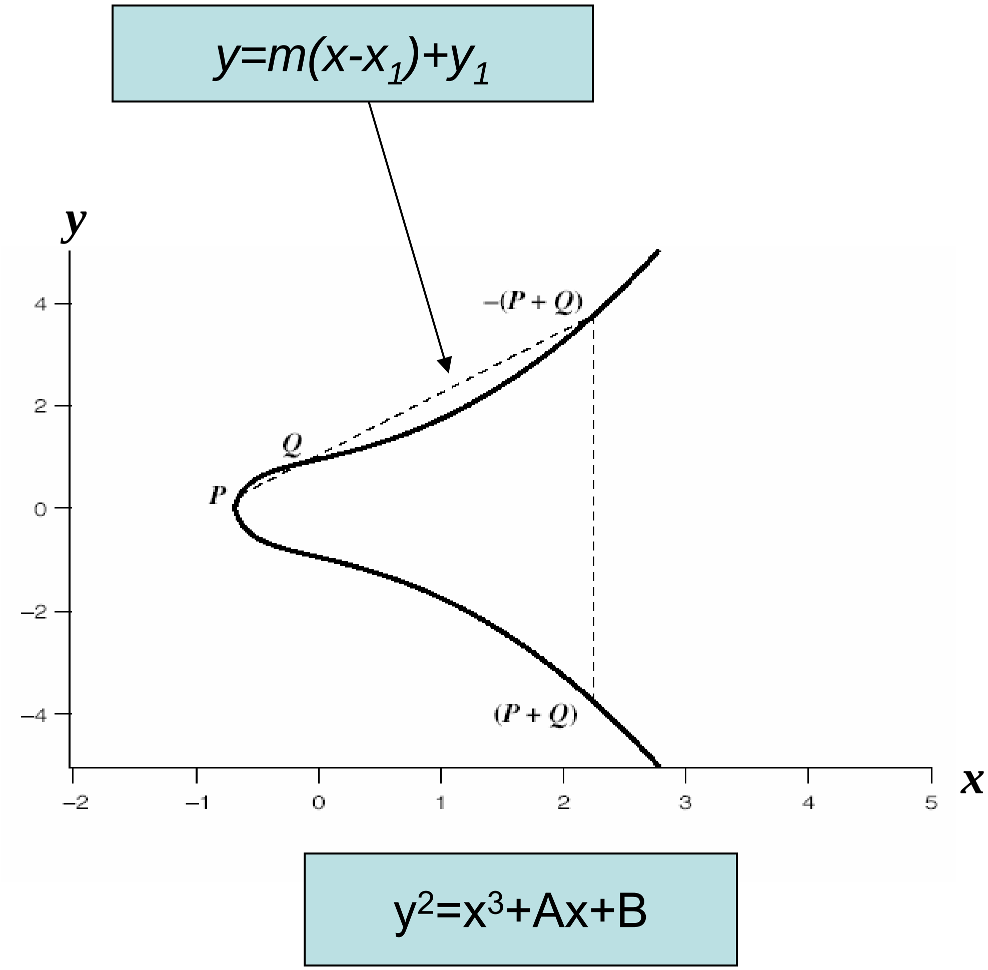
$$E: y^2 = x^3 - x + 1$$

- 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

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

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

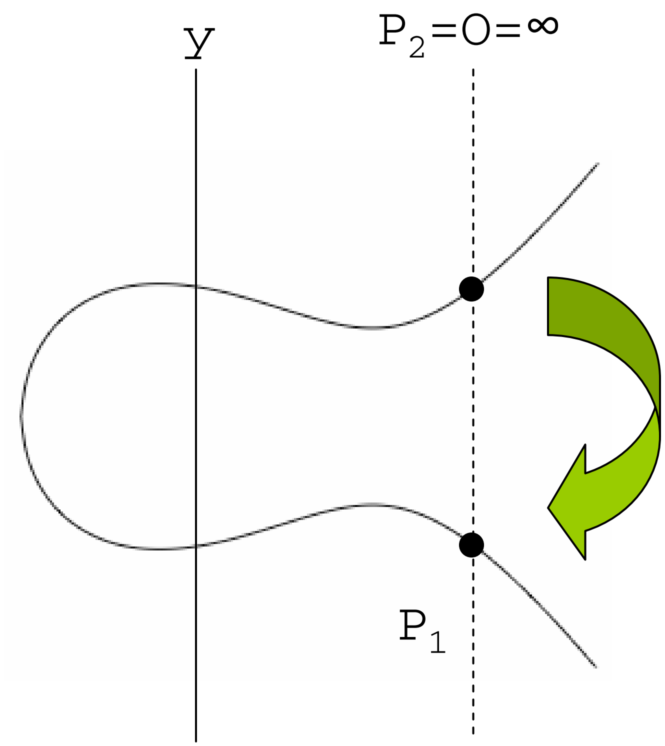
$If, y_1 \neq 0$ (since then $P_1+P_2=\infty$):

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

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

- What happens when $P_2=\infty$?

# Why do we need the reflection?




$P_1 = P_1 + O = P_1$

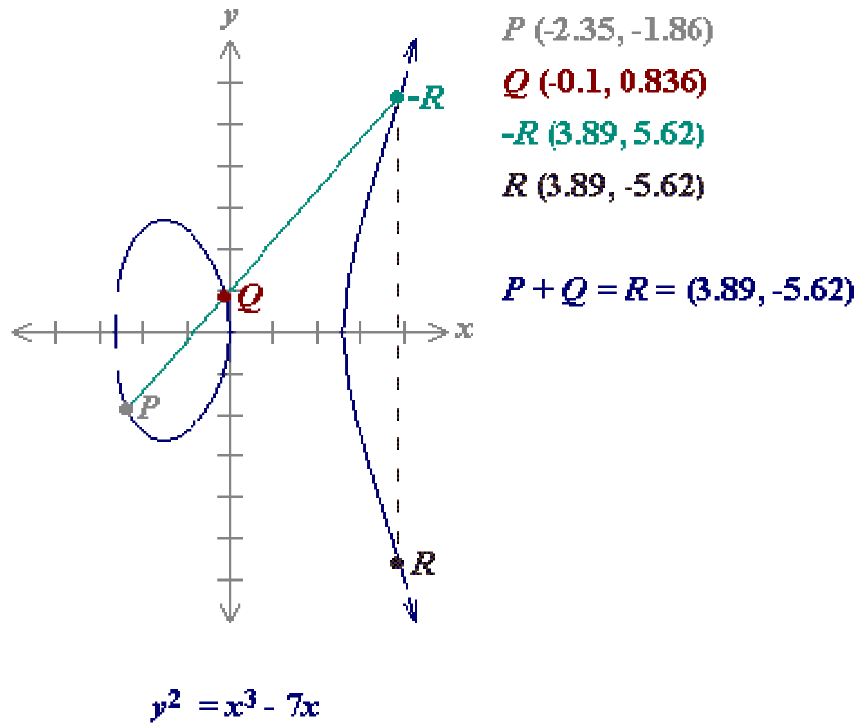
# Sum of two points

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

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

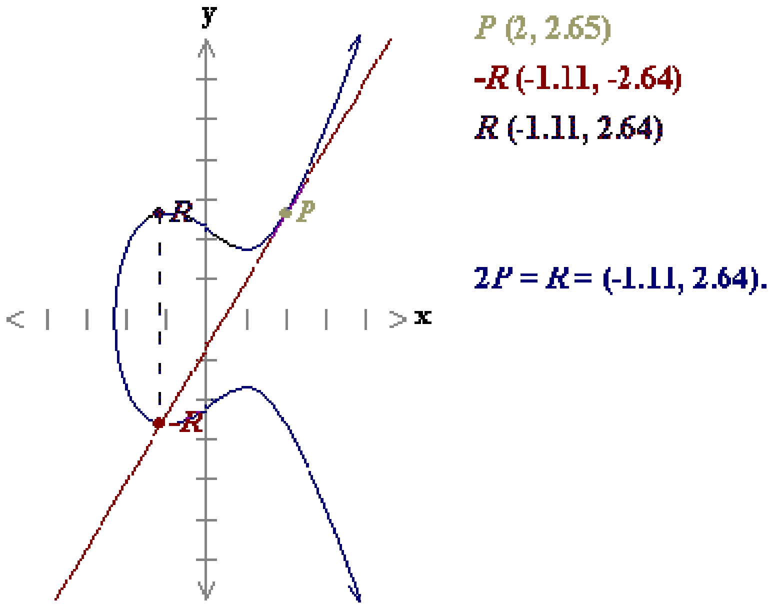
Then ***P+Q*** is given by ***R***$(x_3,y_3)$ :

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

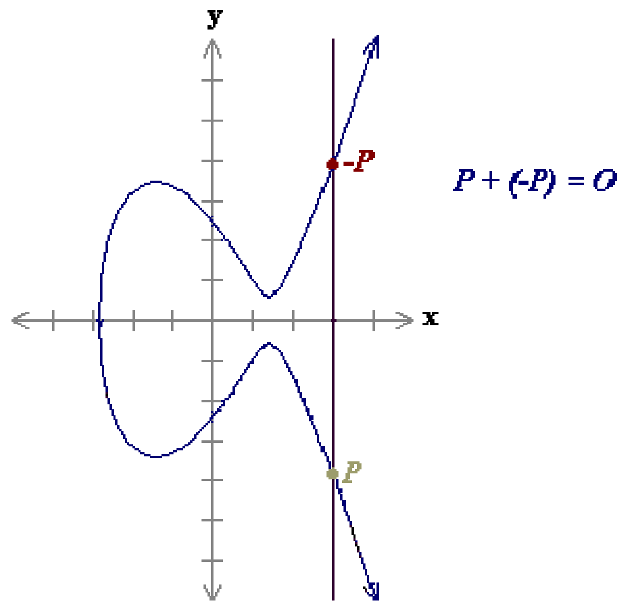


Point at infinity ***O***

***P+P = 2P***




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




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

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***.

# Projective Co-ordinates

- Two-dimensional projective space $P_K^2$ over *K* is given by the equivalence classes of triples (x,y,z) with x,y z in K and at least one of x, y, z nonzero.
- Two triples $(x_1,y_1,z_1)$ and $(x_2,y_2,z_2)$ are said to be equivalent if there exists a non-zero element λ in K, st:
  - $(x_1,y_1,z_1) = (\lambda x_2, \lambda y_2, \lambda z_2)$
  - The equivalence class depends only the ratios and hence is denoted by (x:y:z)

# Projective Co-ordinates

- If $z \neq 0$, $(x:y:z)=(x/z:y/z:1)$
- What is $z=0$? We obtain the point at infinity.
- The two dimensional affine plane over K:

$$A_K^2 = \{(x, y) \in K \times K\}$$

Hence using,

$$(x, y) \to (X : Y : 1)$$

$$\Rightarrow A_K^2 = P_K^2$$

There are advantages with projective co-ordinates
from the implementation point of view

# Singularity

- For an elliptic curve $y^2=f(x)$, define

$F(x,y)=y^2-F(x)$. A singularity of the EC is a pt $(x_0,y_0)$ such that:

$$\frac{\partial F}{\partial x}(x_0,y_0)=\frac{\partial F}{\partial y}(x_0,y_0)=0$$

$$or, 2y_0=-f'(x_0)=0$$

$$or, f(x_0)=f'(x_0)$$

$$\therefore \text{ f has a double root}$$

It is usual to assume the EC has no singular points

## If Characteristics of field is not 3:

$$y^2 = f(x) = x^3 + Ax + B$$

1. **Hence condition for no singularity is $4A^3+27B^2 \neq 0$**
2. **Generally, EC curves have no singularity**

$$\frac{\partial F}{\partial x}(x_0, y_0) = \frac{\partial F}{\partial y}(x_0, y_0) = 0$$

$$or, 2y_0 = -f'(x_0) = 0$$

$$or, f(x_0) = f'(x_0)$$

$\therefore$ f has a double root

$$y^2 = x^3 + Ax + B$$

For double roots,

$$x^3 + Ax + B = 3x^2 + A = 0$$

$$\Rightarrow x^2 = -A/3.$$

Also, $x^4 + Ax^2$+Bx=0,

$$\Rightarrow \frac{A^2}{9} - \frac{A^2}{3} + Bx = 0$$

$$\Rightarrow x = \frac{2A^2}{9B}$$

$$\Rightarrow 3(\frac{2A^2}{9B})^2 + A = 0$$

$$\Rightarrow 4A^3 + 27B^2 = 0$$

# Elliptic Curves in Characteristic 2

- Generalized Equation:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

- If $a_1$ is not 0, this reduces to the form:

$$y^2 + xy = x^3 + Ax^2 + B$$

- If $a_1$ is 0, the reduced form is:

$$y^2 + Ay = x^3 + Bx + C$$

- Note that the form cannot be:

$$y^2 = x^3 + Ax + B$$

# Outline of the Talk…

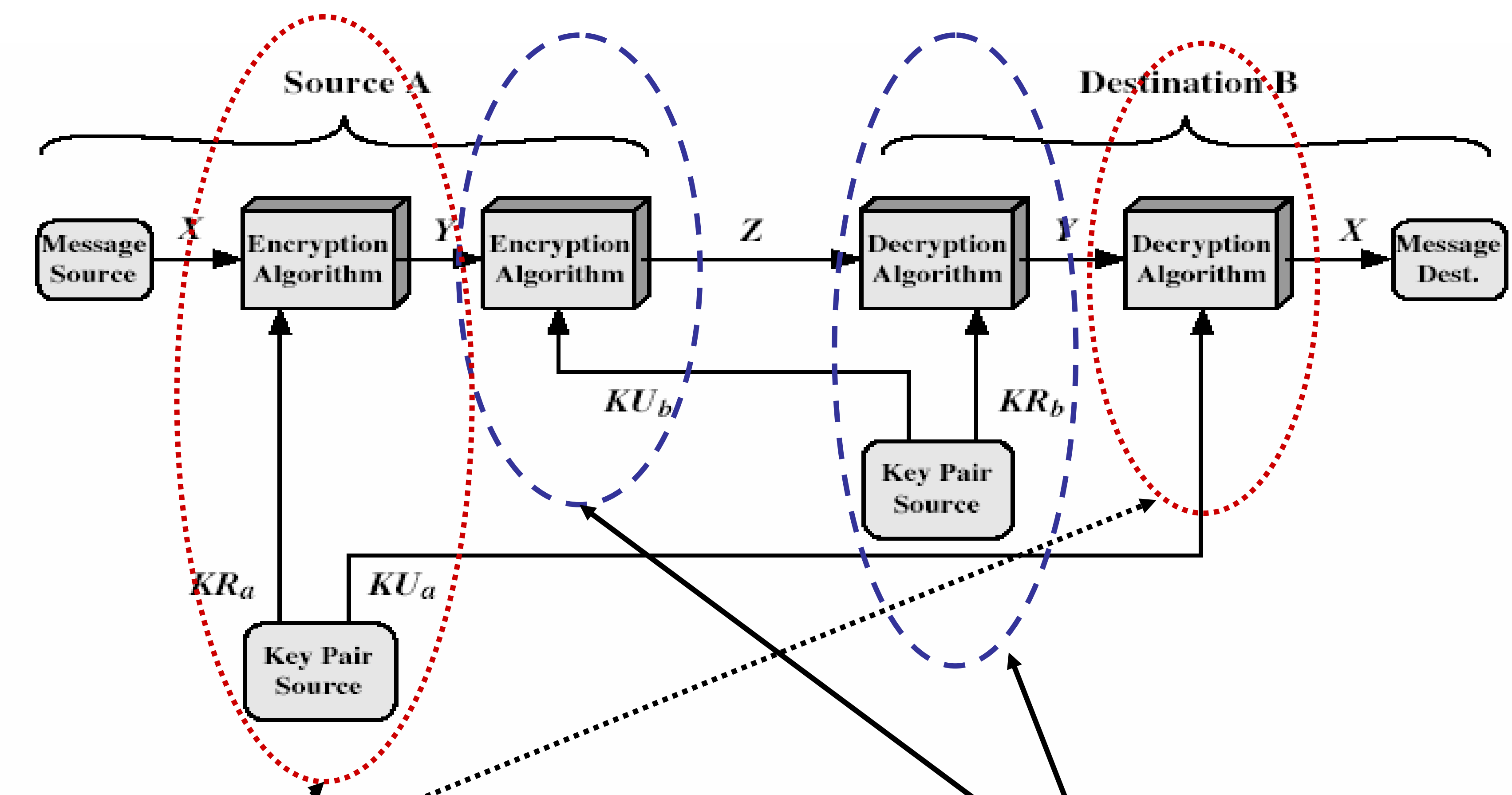
- Introduction to Elliptic Curves

- **Elliptic Curve Cryptosystems**

- Implementation of ECC in Binary Fields

# Elliptic Curve Cryptosystems (ECC)

# Public-Key Cryptosystems

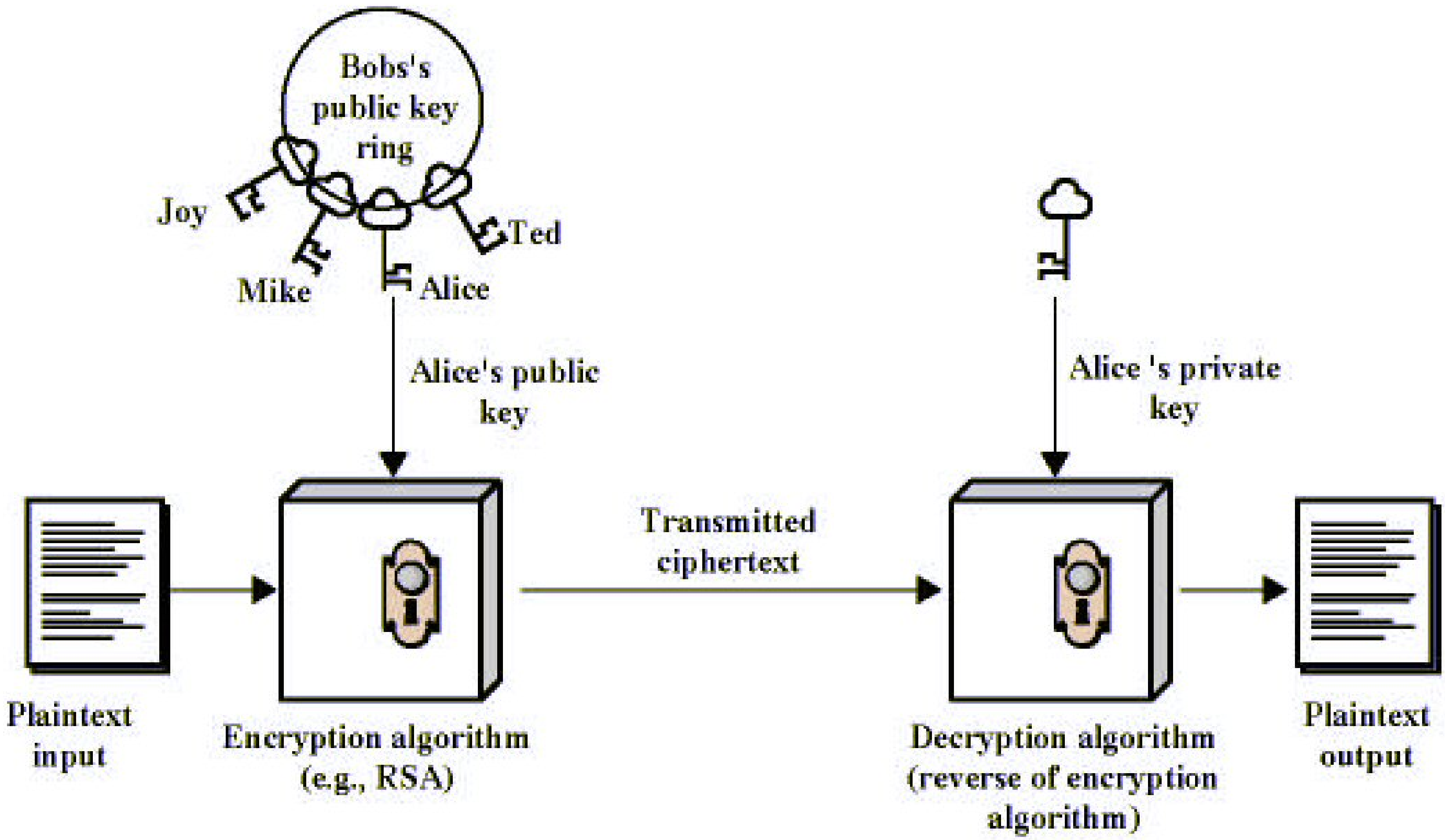



Public-Key Cryptosystem: Secrecy and Authentication

**Authentication**: Only **A** can generate the encrypted message

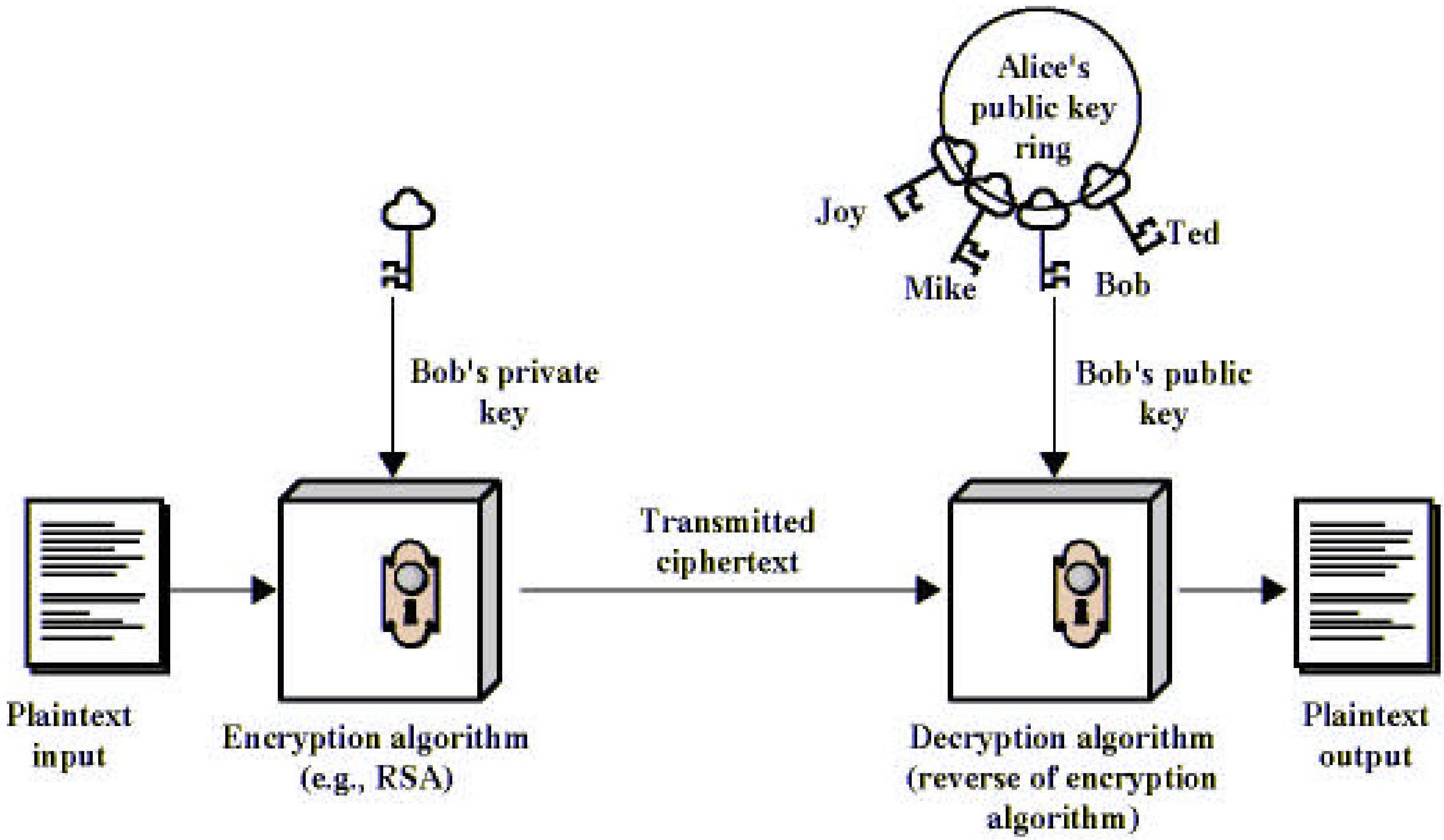
**Secrecy**: Only **B** can Decrypt the message

# Public-Key Cryptography




(a) Encryption

# Public-Key Cryptography




(b) Authentication

# What Is Elliptic Curve Cryptography (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).

# Generic Procedures of ECC

- Both parties agree to some publicly-known data items
  - The **<u>elliptic curve equation</u>**
    - values of ***a*** and ***b***
    - prime, ***p***
  - The **<u>elliptic group</u>** computed from the elliptic curve equation
  - A **<u>base point</u>**, 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 Analog to El Gamal

- 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 – Elliptic Curve Cryptosystem Analog to El Gamal

- 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.

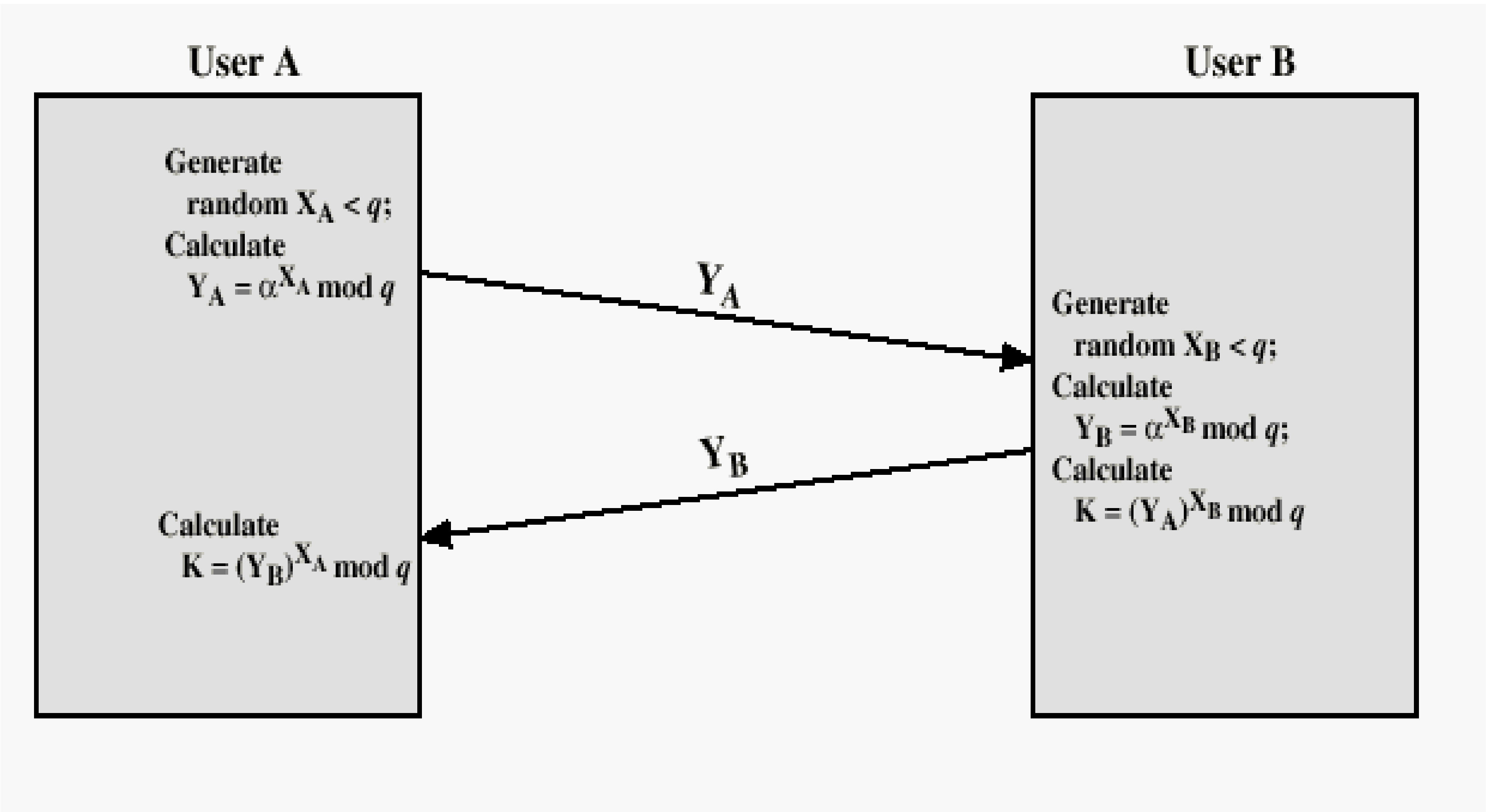
# Example – Compare to El Gamal

- The ciphertext is a pair of points
  - $P_C = [\ (kB), (P_M + kP_B)\ ]$
- The ciphertext in El Gamal is also a pair.
  - $C = (g^k \bmod p,\ mP_B{}^k \bmod p)$

------------------------------------------------------------------------

- Bob then takes this product and subtracts it from the second point from $P_C$
  - $(P_M + kP_B) – [b(kB)] = P_M + k(bB) – b(kB) = P_M$
- In El Gamal, Bob takes the quotient of the second value and the first value raised to Bob's private value
  - $m = mP_B{}^k / (g^k)^b = mg^{k*b} / g^{k*b} = m$

# 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

Alice, A

$\xrightarrow{a(x,y)}$

$\xleftarrow{b(x,y)}$

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

# 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

# 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**

# Outline of the Talk…

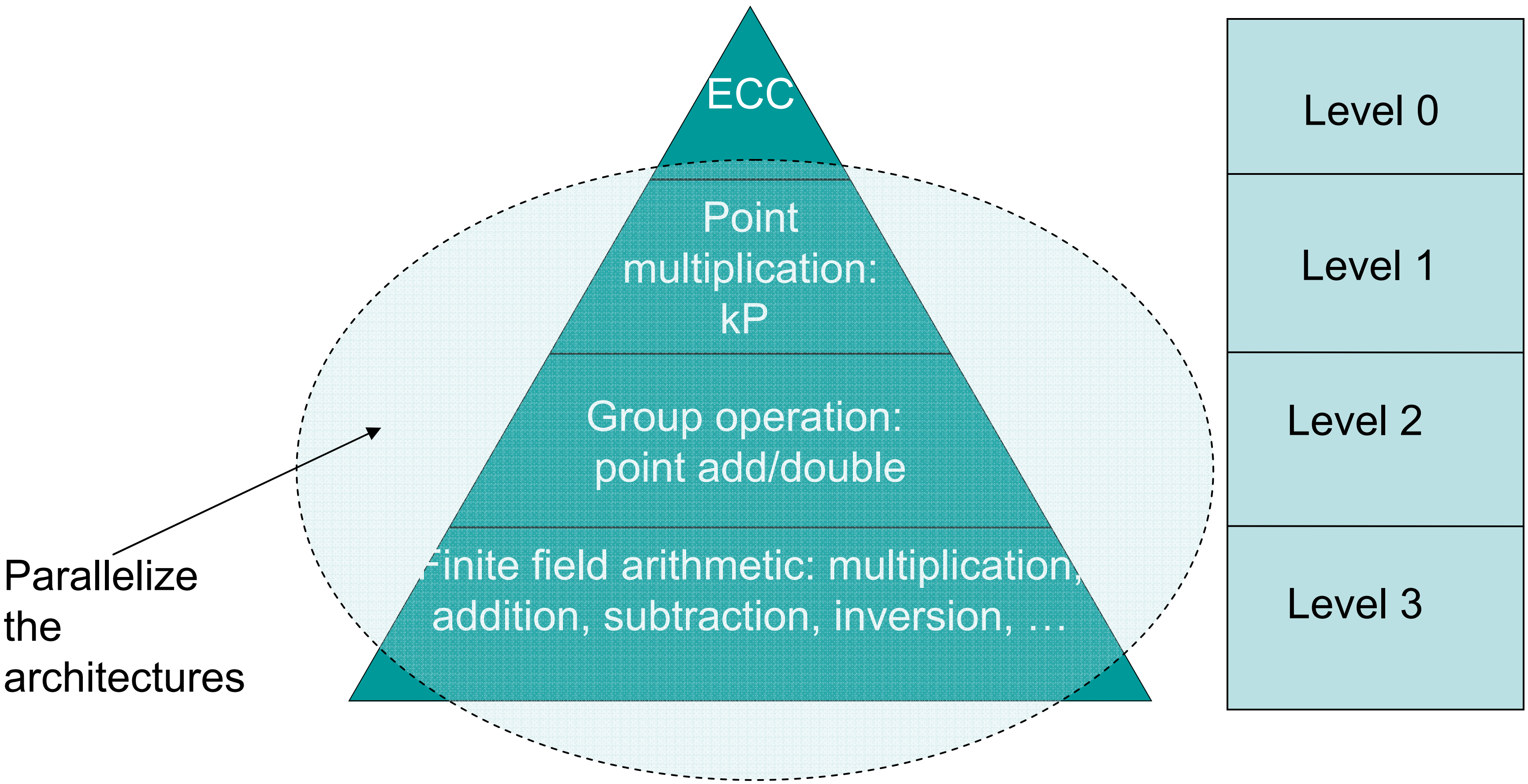
- Introduction to Elliptic Curves
- Elliptic Curve Cryptosystems
- **Implementation of ECC in Binary Fields**

# Implementation of ECC in Binary Fields

# Sub-Topics

1. Scalar Multiplication: LSB first vs MSB first
2. Montgomery Technique of Scalar Multiplication
3. Fast Scalar Multiplication without pre-computation.
4. Lopez and Dahab Projective Transformation to Reduce Inverters
5. Mixed Coordinates
6. Parallelization Techniques
7. Half and Add Technique for Scalar Multiplication

# ECC operations: Hierarchy



# Scalar Multiplication: MSB first

- Require $k=(k_{m-1},k_{m-2},\ldots,k_0)_2$, $k_m=1$
- Compute $Q=kP$
  - $Q=P$
  - For $i=m-2$ to $0$
    - $Q=2Q$
    - If $k_i=1$ then
      - $Q=Q+P$
    - End if
  - End for
  - Return Q

**Sequential** Algorithm

Requires m point doublings and (m-1)/2 point additions on the average

# Example

- **Compute 7P:**
  - $7=(111)_2$
  - 7P=2(2(P)+P)+P=> 2 iterations are required
  - Principle: First double and then add (accumulate)
- **Compute 6P:**
  - $6=(110)_2$
  - 6P=2(2(P)+P)

# Scalar Multiplication: LSB first

- Require $k=(k_{m-1},k_{m-2},...,k_0)_2$, $k_m=1$
- Compute Q=kP
  - Q=0, R=P
  - For i=0 to m-1
    - If $k_i=1$ then
      - Q=Q+R
    - End if
    - R=2R
  - End for
  - Return Q

Can **Parallelize**…

What you are doubling and what you are accumulating are different…

On the average m/2 point Additions and m/2 point doublings

# Example

- **Compute 7P**, $7=(111)_2$, Q=0, R=P
  - Q=Q+R=0+P=P, R=2R=2P
  - Q=P+2P=3P, R=4P
  - Q=7P, R=8P
- **Compute 6P**, $6=(110)_2$, Q=0, R=P
  - Q=0, R=2R=2P
  - Q=0+2P=2P, R=4P
  - Q=2P+4P=6P, R=8P

# Compute 31P…

$31=(11111)_2$

| MSB First | LSB First |
|---|---|
| 1. Q=2P | 1. Q=P, R=2P |
| 2. Q=3P | 2. Q=3P, R=4P |
| 3. Q=6P | 3. Q=7P, R=8P |
| 4. Q=7P | 4. Q=15P, R=16P |
| 5. Q=14P | 5. Q=31P, R=32P |
| 6. Q=15P | |
| 7. Q=30P | |
| 8. Q=31P | |

# Weierstrass Point Addition

$$y^2 + xy = x^3 + ax^2 + b,\ (x, y) \in GF(2^m) \times GF(2^m)$$

- Let, $P=(x_1,y_1)$ be a point on the curve.
- $-P=(x_1,x_1+y_1)$
- Let, $R=P+Q=(x_3,y_3)$

$$x_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)^2 + \dfrac{y_1+y_2}{x_1+x_2} + x_1 + x_2 + a; P \neq Q \\ x_1^2 + \dfrac{b}{x_1^2}; P = Q \end{cases}$$

$$y_3 = \begin{cases} \left(\dfrac{y_1+y_2}{x_1+x_2}\right)(x_1+x_3) + x_3 + y_1; P \neq Q \\ x_1^2 + (x_1 + \dfrac{y_1}{x_1})x_3 + x_3; P = Q \end{cases}$$

1. Point addition and doubling each require 1 inversion & 2 multiplications
2. We neglect the costs of squaring and addition
3. ***Montgomery noticed that the x-coordinate of 2P does not depend on the y-coordinate of P***

# Montgomery's method to perform scalar multiplication

- Input: $k>0$, $P$
- Output: $Q=kP$

1. Set $k \leftarrow (k_{l-1},\ldots,k_1,k_0)_2$
2. Set $P_1=P$, $P_2=2P$
3. For $i$ from $l-2$ to $0$

   If $k_i=1$,

   Set $P_1=P_1+P_2$, $P_2=2P_2$

   else

   Set $P_2=P_2+P_1$, $P_1=2P_1$
4. Return $Q=P_1$

**Invariant Property:**
**$P=P_2-P_1$**

**Question:** How to implement the Operation efficiently?

# Example

Compute 7P

- $7=(111)_2$
- Initialization:

  $P_1=P; P_2=2P$
- Steps:
  - $P_1=3P$, $P_2=4P$
  - $\mathbf{P_1=7P}$, $P_2=8P$

Compute 6P

- $7=(110)_2$
- Initialization:

  $P_1=P; P_2=2P$
- Steps:
  - $P_1=3P$, $P_2=4P$
  - $P_2=7P$, $\mathbf{P_1=6P}$

# Fast Multiplication on EC without pre-computation

# Result-1

- Let $P_1 = (x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Then the x-coordinate of $P_1+P_2$, $x_3$ can be computed as:

$$x_3 = \frac{x_1 y_2 + x_2 y_1 + x_1 x_2^2 + x_2 x_1^2}{(x_1 + x_2)^2}$$

**Hint: Remember that the field has a characteristic 2 and that $P_1$ and $P_2$ are points on the curve**

# Result-2

- Let P=(x,y), $P_1$ = ($x_1$,$y_1$) and $P_2$=($x_2$,$y_2$) be elliptic points. Let *$P=P_2-P_1$* be an invariant.

Then **the x-coordinate of $P_1$+$P_2$, $x_3$ can be computed in terms of the x-coordinates** as:

$$x_3 = \begin{cases} x + \left(\dfrac{x_1}{x_1+x_2}\right)^2 + \dfrac{x_1}{x_1+x_2}; P_1 \neq P_2 \\ x_1^2 + \dfrac{b}{x_1^2}; P_1 = P_2 \end{cases}$$

# Result-3

Let $P=(x,y)$, $P_1=(x_1,y_1)$ and $P_2=(x_2,y_2)$ be elliptic points. Assume that *$P_2$-$P_1$=$P$* and x is not 0. Then **the y-coordinates of $P_1$ can be expressed in terms of P, and the x-coordinates of $P_1$ and $P_2$** as follows:

$$y_1 = (x_1 + x)\{(x_1 + x)(x_2 + x) + x^2 + y\} / x + y$$

# Final Algorithm

Input: $k>0$, $P=(x,y)$
Output: $Q=kP$

1. If $k=0$ or $x=0$ then output(0,0)
2. Set $k = (k_{l-1},k_{l-2},\ldots,k_0)_2$
3. Set $x_1=x$, $x_2=x^2+b/x^2$
4. For i from *l-2* to 0
   1. Set $t=x_1/(x_1+x_2)$
   2. If $k_i=1$,
      $x_1=x+t^2+t$, $x_2=x_2^2+b/x_2^2$
      else
      $x_1=x_1^2+b/x_1^2$, $x_2=x+t^2+t$
5. $r_1=x_1+x$, $r_2=x_2+x$
6. $y_1=r_1(r_1r_2+x^2+y)/x+y$
7. Return $Q=(x_1,y_1)$

- #INV:2(l-2)+1;
- #MULT: 2(l-2)+4
- #ADD: 4(l-2)+6
- #SQR: 2(l-2)+2

# How to reduce inversions?

1. In affine coordinates Inverses are very expensive
2. For $n \geq 128$ each inversion requires around 7 multipliers (in hardware designs)
3. Lopez Dahab Projective coordinates:
   - (X,Y,Z), Z≠0, maps to $(X/Z, Y/Z^2)$
   - Motivation is to replace inversions by the multiplication operations and then perform one inversion at the end (to obtain back the affine coordinates)

# Doubling

- **Remember:**

$$x_3 = \begin{cases} x + \left(\dfrac{x_1}{x_1 + x_2}\right)^2 + \dfrac{x_1}{x_1 + x_2}; P_1 \neq P_2 \\ x_1^2 + \dfrac{b}{x_1^2}; P_1 = P_2 \end{cases}$$

- 2 inverses
- 1 general field multiplication
- 4 additions
- 2 squarings

- **In Projective Coordinates**:

$$P_1 = P_2,\ X_3 = X_1^4 + b.Z_1^4$$

$$Z_3 = Z_1^2.X_1^2$$

$$P_1 \neq P_2,\ Z_3 = (X_1.Z_2 + X_2.Z_1)^2$$

$$X_3 = x.Z_3 + (X_1.Z_2).(X_2.Z_1)$$

- 0 inverses
- 4 general field multiplications
- 3 additions
- 5 squarings

# Montgomery Algorithm

- Input: $k>0$, $P=(x,y)$
- Output: $Q=kP$
- Set $k \leftarrow (k_{l-1},\ldots,k_1,k_0)_2$
- Set $X_1=x$, $Z_1=1$; $X_2=x^4+b$, $Z_2=x^2$
- For i from *l-2* to 0
  - If $k_i=1$,

    Madd($X_1,Z_1,X_2,Z_2$), Mdouble($X_2,Z_2$)

    else

    Madd($X_2,Z_2,X_1,Z_1$), Mdouble($X_1,Z_1$)
- Return $Q=(Mxy(X_1,Y_1,X_2,Y_2))$

# Mxy: Projective to Affine

$$x_3 = X_1 / Z_1$$

$$y_3 = (x + X_1 / Z_1)[(X_1 + xZ_1)(X_2 + xZ_2) + (x^2 + y)(Z_1 Z_2)](xZ_1 Z_2)^{-1} + y$$

Requires 10 multiplications and one inverse operation

# Final Comparison

| *Affine Coordinates* | *Projective Coordinates* |
| --- | --- |
| **Inv: 2logk + 1** | **Inv: 1** |
| **Mult: 2logk + 4** | **Mult: 6logk + 10** |
| Add: 4logk + 6 | Add: 3logk + 7 |
| Sqr: 2logk + 2 | Sqr: 5logk + 3 |

**Hence, final decision depends upon the I:M ratio of the finite field operators**

# Addition in Mixed Coordinates

- **Theorem:** Let $P_1=(X_1/Z_1, Y_1/Z_1^2)$ and $P_2=(X_2/Z_2, Y_2/Z_2^2)$ be two points on the curve. If $Z_1=1$, then $P_1+P_2=(X_3/Z_3, Y_3/Z_3^2)$ st.

$$U = Z_2^2 Y_1 + Y_2, S = Z_2 X_1 + X_2, T = Z_2 S, Z_3 = T^2,$$

$$V = Z_3 X_1, X_3 = U^2 + T(U + S^2 + Ta),$$

$$Y_3 = (V + X_3)(TU + Z_3) + Z_3^2 C$$

**Number of multiplications are further reduced**.
Squaring is increased a bit, but they are cheap in $GF(2^n)$
Improvement by 10 % if a≠0, otherwise 12 %...

# Parallel Strategies for Scalar Point Multiplication

- **Point Doubling**
  - Cycle 1: $T=X_1^2$, $M=cZ_1^2$, $Z_2=T.Z_1^2$
  - Cycle 1a: $X_2=T^2+M^2$

1 multiplier

- Point Addition
  - Cycle 1: $t_1=(X_1.Z_2)$; $t_2=(Z_1.X_2)$
  - Cycle 1a: $M=(t_1+t_2)$, $Z_1=M^2$
  - Cycle 2: $N=t_1.t_2$, $M=xZ_1$
  - Cycle 2a: $X_1=M+N$

2 multipliers

We assume that squarings and multiplications with constants can be performed without multipliers…

# Parallelizing Montgomery Algorithm

1. Input: $k>0$, $P=(x,y)$
2. Output: $Q=kP$
3. Set $k \leftarrow (k_{l-1},\ldots,k_1,k_0)_2$
4. Set $X_1=x$, $Z_1=1$; $X_2=x^4+b$, $Z_2=x^2$
5. For i from *l-2* to 0

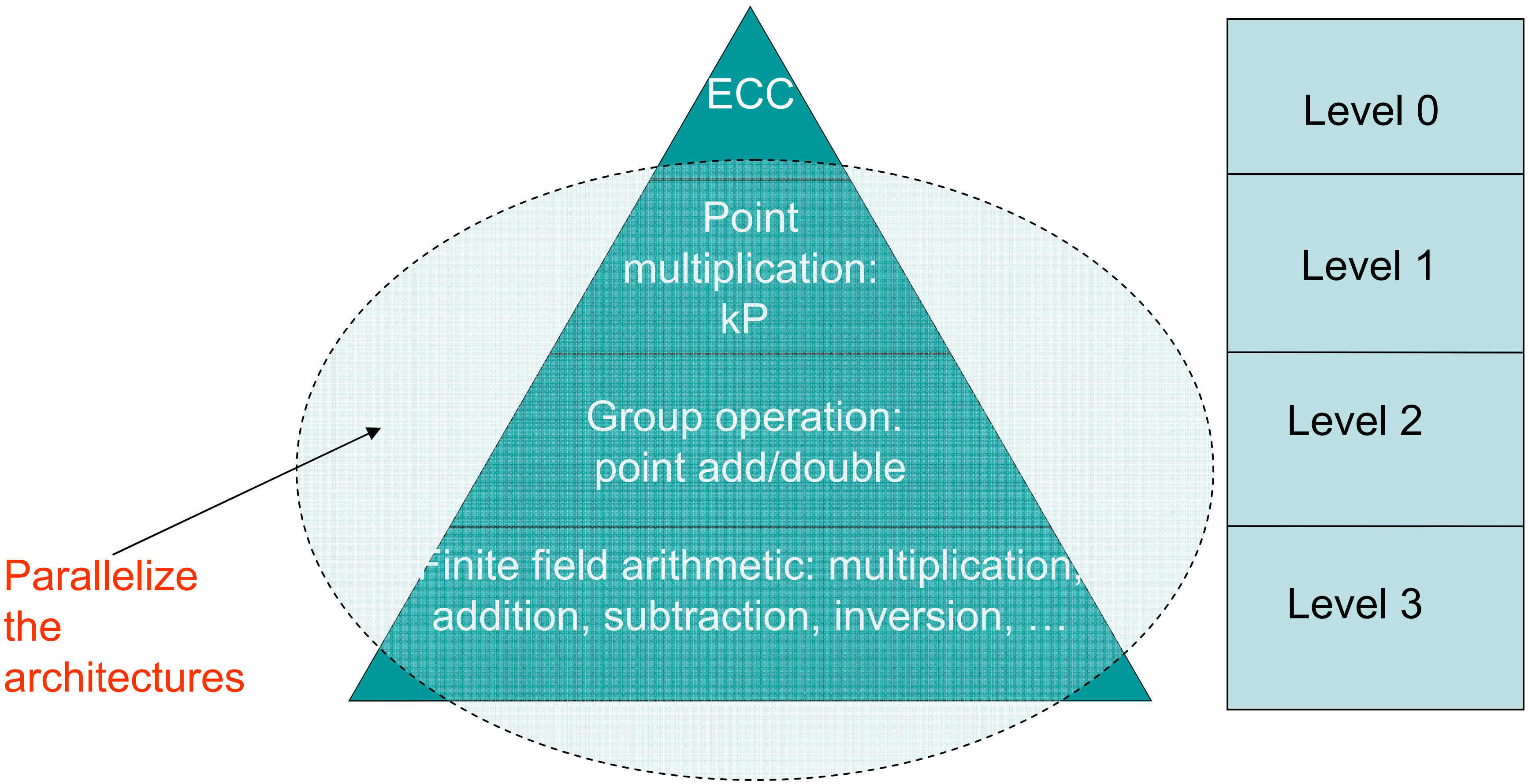
   If $k_i=1$,

   5a) Madd($X_1,Z_1,X_2,Z_2$), Mdouble($X_2,Z_2$)

   else

   5b) Madd($X_2,Z_2,X_1,Z_1$), Mdouble($X_1,Z_1$)
6. Return $Q=(Mxy(X_1,Y_1,X_2,Y_2))$

# Looking back at our Design Hierarchy



# Parallelizing Strategies

- **Parallelize level 1:** If we allocate one multiplier to each of  Madd and Mdouble, then we can parallelize steps 5a and 5b. Thus 4 clock cycles are required for each iteration. Total time is nearly 4l.
- **Parallelize level 2:** If we can parallelize the underlying Madd and Mdouble, then we cannot parallelize level 1, if we have constraint of 2 multipliers. So, we have a sequential step 5a and 5b. Total time is 3l.

# Parallelizing Strategies

- **Parallelize both the levels:** Total time is *2l* clock cycles. Require 3 multipliers.
- Thus Montgomery algorithm is highly parallelizable
- Helpful in high performance designs (*low power, high thoughput etc*)

# Point Halving

- In 1999 Scroeppel and Knudsen proposed further speed up
- Idea is to replace point doubling by halving
- Point Halving is three times as fast than doubling
- The scalar k, has to be expressed in the negative powers of 2

# Computing the Half

- **Problem:** Let E be the Elliptic Curve, defined by the equation: $y^2 + xy = x^3 + ax^2 + b, b \neq 0$
- Let Q=(u,v)=2P
- Compute P=(x,y)
- Remember :

$$u = x^2 + \frac{b}{x^2}$$

$$v = x^2 + (x + \frac{y}{x})u + u$$

# Halving (contd.)

$$Let, \lambda = x + \frac{y}{x}$$

$$\therefore v = x^2 + (\lambda + 1)u \Rightarrow x = \sqrt{v + (\lambda + 1)u}$$

$$Note : \lambda^2 + \lambda = u + a$$

| |
|---|
| Square Root |
| Solving Quadratics |

- Thus, we have to solve the above equations
- λ-representation: $(x, \lambda_x)$

# Trace of a point

- Define: $Tr(C) = C + C^2 + ... + C^{2^{m-1}}$
- Properties of Trace:
  - $Tr(c)=Tr(c^2)=Tr(c)^2$, Tr(c) can be 0 or 1
  - Tr(c+d)=Tr(c)+Tr(d)
  - NIST Curves : Tr(a)=1
  - If x,y belongs to the Elliptic Curve, Tr(x)=Tr(a)

# Computing λ

- The roots of $\lambda^2 + \lambda = u + a$ are $\lambda_1 = \lambda$ or $\lambda+1$
- **Theorem:**

Let, $P = (x, y), Q = (u, v) \in G, st.\ Q = 2P$ and denote $\lambda = x + y/x$. Let $\hat{\lambda}$ be a solution to $\lambda^2 + \lambda = u + a$ and $t = v + u\hat{\lambda}$. Suppose that $Tr(a) = 1$. Then $\hat{\lambda} = \lambda$ if and only if $Tr(t) = 0$.

# Halving Algorithm

- Input: (u,v) , Output: (x,y)

1. Solve $\lambda^2 + \lambda = u + a$ for λ. Let the root be $\hat{\lambda}$
2. Compute $t = v + u\hat{\lambda}$
3. If Tr(t)=0, then $\lambda_P = \hat{\lambda}$ , $x=(t+u)^{1/2}$
   else $\lambda_P = \hat{\lambda} + 1$, $x=(t)^{1/2}$
4. Return $(x,\lambda_P)$

# Implementation of Trace

- **Trace :** $Tr(C) = Tr(\sum_{i=0}^{m-1} c_i x^i) = \sum_{i=0}^{m-1} c_i Tr(x^i)$

- Can be evaluated in O(1) time
- Example: GF($2^{163}$), with reduction polynomial $p(x)=x^{163}+x^7+x^6+x^3+1$, $Tr(x^i)=1$, iff i=0 or 159.
- Thus, the implementation is only one xor gate to add the 0th and the 159th bits of the register storing C.

# Solving a Quadratic over GF($2^m$)

- Solve $x^2+x=c+Tr(c)$, c is an element of GF($2^m$)
- Define Half Trace:

$$H(C)=\sum_{i=0}^{(m-1)/2} C^{2^{2i}}$$

1. $H(C+D)=H(C)+H(D)$

2. $H(C)$ is a root for $x^2+x=C+Tr(C)$, as

$$H(C)=H(C^2)+C+Tr(C)$$

H(C) gives a root for the quadratic equation. A simple method to find H(C) requires storage for m elements and m/2 field additions on an average

# Obtaining Square Root

- Field squaring in binary field is linear
- Hence squaring can be rephrased as:
  - $C=MA=A^2$
- We require to compute D st. $D^2=A$
- Let, $D=M^{-1}A$=> $A=MD$
- $D^2=MD$ (as M is the squaring matrix)

  $=M(M^{-1}A)=A$
- Hence, $D=(A)^{1/2}$

# An Example

Compute: $763\mathrm{R}_7$, where order of $\mathrm{R}_7 = 1013$

$\Rightarrow m = 10$

$2^{10-1}(763) = 651(\mathrm{mod}\ 1013) = (1010001011)_2$

$\therefore 763 = (\frac{1}{2^9} + \frac{1}{2^8} + \frac{1}{2^6} + \frac{1}{2^2} + 1)\,\mathrm{mod}(1013)$

$\therefore 763R_7$ may be computed using the following steps:

Step 1: $\frac{1}{2}R_7 + R_7$

Step 2: $\frac{1}{2}(\frac{1}{2}R_7 + R_7) + R_7$

Step 3: Similarly continue...

# Half and Add Algorithm

1. Input: $0<k<n$, $P=(x,y)$
2. Output: $Q=kP$
3. Compute: $t=\lfloor \log_2 n \rfloor + 1$ , $k_1=(2^{t-1}k) \bmod n$
4. $Q=O$
5. for $i=0$ to $m-1$ do
    1. $Q=[1/2]Q$
    2. If, $k_1^i=1$, then $Q=Q+P$
6. return Q

**No method is currently known to perform point halving in projective Coordinates. Keep Q in affine coordinates and P in Projective Coordinates. Then step 5.2 is a mixed operation, giving further efficiency.**

# Key References

- **Papers:**
  - *J. Lopez and R. Dahab, "Fast Multiplication on Elliptic Curves over GF($2^m$) without pre-computation", CHES 1999*
  - *K. Fong etal, "Field Inversion and Point Halving Revisited", IEEE Trans on Comp, 2004*
  - *G. Orlando and C. Paar, "A High Performance Reconfigurable Elliptic Curve Processor for GF($2^m$)", CHES 2000*
  - *N. A. Saqib etal, "A Parallel Architecture for Fast Computation of Elliptic Curve Scalar Multiplication over GF($2^m$)", Elsevier Journal of Microprocessors and Microsystems, 2004*
  - *Sabiel Mercurio etal, " An FPGA Arithmetic Logic Unit for Computing Scalar Multiplication using the Half-and-Add Method", IEEE ReConfig 2005*

# Key References

- **Books:**
  - *Elliptic Curves: Number Theory and Cryptography, by Lawrence C. Washington*
  - *Guide to Elliptic Curve Cryptography, Alfred J. Menezes*
  - *Guide to Elliptic Curve Cryptography, Darrel R. Hankerson, A. Menezes and A. Vanstone*
  - *http://cr.yp.to/ecdh.html ( Daniel Bernstein)*

Thank You