Algorithms for Information Security and Privacy 2017

10.

Recap: Point Addition on Elliptic Curve

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

$P+Q = (x_3, y_3)$ where.

$$m = \frac{y_2 - y_1}{x_2 - x_1} = (y_2 - y_1) \cdot (x_2 - x_1)^{-1}$$

$$c = y_1 - m \cdot x_1$$

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

$$y_3 = -m \cdot x_3 - c$$

Computation time for field inversion $O(\lg^3 q)$

Bottleneck of Point Addition.

This class: Reduce the computation for field inversion

$$M = y_2 - y_1 \qquad Z_3 = x_2 - x_1$$

$$m = M \cdot Z_3^{-1}$$

$$c = y_1 - M \cdot Z_3^{-1} \cdot x_1$$

$$x_3 = m^2 - x_1 - x_2 = M^2 \cdot Z_3^{-2} - x_1 - x_2$$

$$= Z_3^{-2} \underbrace{\left[M^2 - x_1 \cdot Z_3^2 - x_2 \cdot Z_3^2 \right]}_{X_3}$$

$$= X_3 \cdot Z_3^{-2}$$

$$y_3 = -m x_3 - c$$

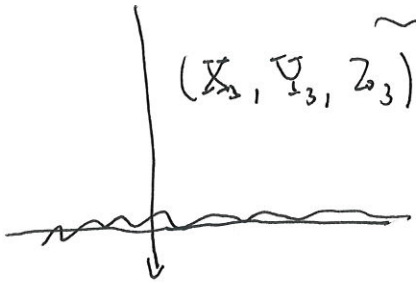
$$= -M \cdot Z_3^{-1} \cdot X_3 \cdot Z_3^{-2} - y_1 + M \cdot Z_3^{-1} \cdot x_1$$

$$= Z_3^{-3} \underbrace{\left[-M \cdot X_3 - y_1 \cdot Z_3^3 + M \cdot x_1 \cdot Z_3^2 \right]}_{Y_3}$$

$$= Y_3 \cdot Z_3^{-3}$$

$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ → before.

$(x_1, y_1) + (x_2, y_2) = (X_3, Y_3, Z_3)$ → point in Jacobian Coordinate.



$(X_3, Y_3, Z_3) \rightarrow (x_3, y_3)$ when. $x_3 = X_3 \cdot Z_3^{-2}$

$y_3 = Y_3 \cdot Z_3^{-3}$.

Complexity = $O(\lg^2 p)$ # Multiplication is the bottleneck operation.

$P + Q + R$

$(X_3, Y_3, Z_3) + (x_4, y_4) = (x_4, y_4) + (X_3, Y_3, Z_3) = (x_5, y_5)$

$$m = \frac{y_3 - y_4}{x_3 - x_4} = (Y_3 \cdot Z_3^{-3} - y_4) \cdot (X_3 \cdot Z_3^{-2} - x_4)^{-1}$$

$$= Z_3^{-3} \cdot \underbrace{(Y_3 - y_4 \cdot Z_3^3)}_{M} \cdot Z_3^3 \cdot \underbrace{(X_3 \cdot Z_3 - x_4 Z_3^3)^{-1}}_{Z_5'}$$

$$= M \cdot Z_5'^{-1}$$

$$c = y_4 - m x_4 = y_4 - M \cdot (Z_5')^{-1} \cdot x_4$$

$$x_5 = m^2 - x_4 - x_3$$

$$= (M \cdot (Z_5')^{-1})^2 - x_4 - X_3 \cdot Z_3^{-2}$$

$$= M^2 \cdot Z_5'^{-2} - x_4 - X_3 \cdot Z_3^{-2}$$

$$= Z_5'^{-2} \cdot Z_3^{-2} \left[\underbrace{M^2 \cdot Z_3^2 - x_4 Z_3^2 \cdot Z_5^2}_{X_5} - X_3 \cdot Z_5^2 \right]$$

$$= \underbrace{(Z_5' \cdot Z_3)^{-2}}_{Z_3} \cdot X_5$$

$$x_5 = Z_5^{-2} \cdot X_5.$$

$$y_5 = -M \cdot x_5 - c$$

$$= -M \cdot Z_5'^{-1} \cdot Z_5^{-2} X_5 - y_4 + M \cdot Z_5^{-1} \cdot x_4$$

$$= \frac{Z_5^{-2} \cdot Z_5'^{-1} \cdot Z_3^{-1}}{Z_5^{-1}} \Big[-M \cdot Z_3 \cdot X_5 - y_4 \cdot Z_5^2 \cdot Z_5' \cdot Z_3 + M \cdot Z_5^2 \cdot Z_3 \cdot x_4 \Big]$$

$$= Z_5^{-3} \cdot Y_5$$

Conclusion

$$(X_3, Y_3, Z_3) + (x_4, y_4) = (X_5, Y_5, Z_5)$$

when $M = Y_3 - y_4 \cdot Z_3^3$

$$Z_5' = X_3 \cdot Z_3 - x_4 \cdot Z_3^3$$

$$Z_5 = Z_5' \cdot Z_3$$

$$X_5 = M^2 \cdot Z_3^2 - x_4 Z_3^2 \cdot Z_5^2 - X_3 \cdot Z_5^2$$

$$Y_5 = -M \cdot Z_3 \cdot X_5 - y_4 \cdot Z_5^2 \cdot Z_5' \cdot Z_3 + M \cdot Z_5^2 \cdot Z_3 \cdot x_4$$

Bottleneck is field multiplication $O(\lg^2 p)$!!

Before $(x_1, y_1) + (x_2, y_2) + (x_4, y_4)$

↑ one point inversion ↑ one point inversion

with Jacobian coordinate

$(x_1, y_1) + (x_2, y_2) + (x_4, y_4)$

no point inversion ↑ $(X_3, Y_3, Z_3) + (x_4, y_4)$

no point inversion ↑ (X_5, Y_5, Z_5) ← one point inversion

↓

$(x_5, y_5) = (X_5 \cdot Z_5^{-2}, Y_5 \cdot Z_5^{-3})$

2 point inversions

↓

1 point inversion

Recap: Scalar Multiplication

$aP = \underbrace{P + P + \dots + P}_{a \text{ times}}$ $\quad a$ is as big as 2^p

$57 = 32 + 16 + 8 + 1$

$= 2^5 + 2^4 + 2^3 + 2^0$

$57P = 2^5 P + 2^4 P + 2^3 P + 2^0 P$

Calculate each point separately and add them together

Faster way: $2^0P \to 2^1P \to 2^2P \to 2^3P \to 2^4P \to [2^5P]$ about r — $O(\lg p)$ additions

Add together to have $57P$ — $O(\lg p)$ additions.

Without Jacobian Coordinate: $O(\lg^2 p)$ per addition $\to$ $O(\lg^4 p)$ per scalar multiplication

With Jacobian Coordinate: $O(\lg^2 p)$ per addition + $O(\lg^3 p)$ for point inversion $O(\lg^3 p)$ $\to$ $O(\lg^3 p)$ per scalar multiplication

(converting Jacobian to normal.)

Double-Based Number Representation

$57 = 2^5 + 2^4 + 2^3 + 2^0$

binary representation

$57 = 2 \cdot 3^2 + 3^1 + 3^0$

ternary representation

[Not that better in practice, still $O(\lg^3 p)$ per scalar multiplication]

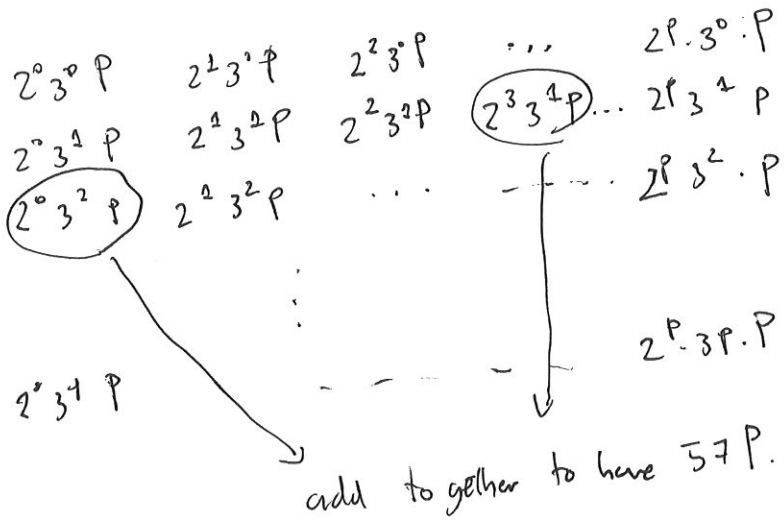
merge

$57 = 2^3 \cdot 3^1 + 3^2 = 48 + 9$

Double-Based Number Representation.

Situation : P (our own private key) is fixed, and a (value obtained from partner) is different on each Scalar Point Multiplication.

Pre-compute



How to find Double-based number representation?

- We want to find a representation with smallest #terms → smallest # additions.
- greedy algorithm.
  - Suppose that we want to calculate $a \cdot P$

1: $Q \leftarrow O$

2: Find the largest $2^i 3^j$ that is no larger than a.

3: $Q \leftarrow Q + 2^i 3^j P$

4: $a \leftarrow a - 2^i \cdot 3^j$

5: If $a = 0$, then terminates

Otherwise, go to Step 2.

Theorem. [Dimitrov, Imbert, Mishra, Maths of Computation '08]

terms obtained from greedy algorithm $= O(\lg p / \lg \lg p)$

$\simeq$ # additions.

Computation time using Double-based number system $= O(\lg^2 p / \underbrace{\lg \lg p}_{\uparrow})$

↑ 16 when $p \approx 2^{256}$

Question : Is it possible to have something better than $O(\lg p / \lg\lg p)$ terms.?

Theorem [Saranurak, Suppakitpaisarn, unpublished]

No representation has #terms $= o(\lg p / \lg\lg p)$.

Elliptic Curve Digital Signature

Alice

Message M ⟶ Bob

Message M.

1: Was it Alice who send M?
(authentication)

2. Was the message edited by someone?
(integrity)

3. Alice cannot refuse that she did send the message (non-repudiation)

Rough Idea

Alice

1: Message M ⟶ Bob

Message M.

2: Encrypt M with her private key (sign)

$E_{k_{pri}}(M)$

3: Send the encrypted message ⟶ $E_{k_{pri}}(M)$ [signature of Alice on M]

Certificate Authority

4: Receive public key of Alice k_{pub}

5: Decrypt using k_{pub} (verify)

$M' = E_{k_{pub}}(E_{k_{pri}}(M))$

6: If $M = M'$, then message M is not false message.

El Gamal's Digital Signature [Elgamal 1985]

Alice

Private key $a \in \mathbb{Z}_{70}$

Public key $A, B(=a \cdot A)$

Bob

Public Key A, B

$M \in \mathbb{Z}_{70}$ [Random number k]

Signing

$R = k \cdot A = (x_R, y_R)$

$s = (m - a x_R) \cdot k^{-1}$

$(R, s) \longrightarrow (R, s)$

verifying

Calculate $x_R B + s R$

Verify if the value $= m \cdot A$

When everything is alright.

$$x_R B + s \cdot R = x_R \cdot aA + (m - a x_R) k^{-1} \cdot k \cdot A$$

$$= x_R \cdot aA + mA - a x_R A$$

$$= mA$$

If I am attacker... (I want to tell Bob that I am Alice)

- I can choose k freely and calculate $R = k \cdot A = (x_R, y_R)$
- But, I need a to calculate s.

  [If I do not have correct a, 2 terms cannot be cancelled]
- I can learn a from A, B → Discrete Logarithm problem.
- I can learn from R and s sent by Alice.

$$s = (m - a x_R) k^{-1}$$

s: known, m: known, x_R: known, a, k^{-1}: unknown.

- To know k, we have to solve discrete logarithm when the inputs are R and A. ($= kA$)