- It is possible to remove y-coordinate from point doubling formula, and if the difference $P_3' = P_1 - P_2$ is known then y-coordinate can also be neglected in the formula for $P_3 = P_1 + P_2$.

Note that in the Montgomery ladder $R_1 - R_0 = P$ in every iteration of the algorithm.

To emphasise that ECADDDBL does not make use of y-coordinate we denote the procedure by xECADDDBL.

Accordingly, the Montgomery ladder takes the form:

Input: • $d = (d_{n-1} d_{n-2} \dots d_1 d_0)_2$, where $d_{n-1} = 1$,
• point P

Output: $d \cdot P$

point doubling procedure, does not need y-coordinate

1. $Q[0] := P$, $Q[1] = xECDBL(P)$
2. for $i = n-2$ downto 0
3. $(Q[d_i \text{ xor } 1], Q[d_i]) := xECADDDBL(Q[d_i], Q[d_i \text{ xor } 1])$

this point will be doubled

the result of point doubling (see line 4 of the first version of the algorithm)

4. Return $Q[0]$.

The algorithm above will utilize the Standard Projective Coordinates. But instead of $P = (x : y : 1)$ we take its randomized

in the first line

representation without y-coordinate, that is $P=(rx:r)$ for $r \in K^*$. The difference $P_3' = Q[1] - Q[0]$ will be represented as $P_3' = (x:1)$, That is $X_3' = x$, $Z_3' = 1$.

Let us derive the formulas for point addition and point doubling without y-coordinate:

$$P_3 = (x_3, y_3) = P_1 + P_2$$

$$P_3' = (x_3', y_3') = P_1 - P_2 \quad , \text{ where } P_1 = (x_1, y_1)$$
$$P_2 = (x_2, y_2)$$

$$x_3 = \left(\frac{y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

$$x_3' = \left(\frac{-y_2 - y_1}{x_2 - x_1}\right)^2 - x_1 - x_2$$

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

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

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

$$= \frac{2x_2^3 + 2ax_2 + 2b + 2x_1^3 + 2ax_1 + 2b - 2x_2^3 + 2x_1x_2^2 + 2x_1^2x_2 - 2x_1^3}{(x_2 - x_1)^2}$$

$$= \frac{2x_1(x_1x_2 + a) + 2x_2(x_1x_2 + a) + 4b}{(x_2 - x_1)^2} = \frac{2(x_1 + x_2)(x_1x_2 + a) + 4b}{(x_2 - x_1)^2}$$

(7) $x_3 = \dfrac{2(x_1+x_2)(x_1x_2+a)+4b}{(x_2-x_1)^2} - x_3'$

$$P_4 = 2 \cdot P_1$$

$$P_4 = (x_4, y_4)$$

(8) $x_4 = \left(\dfrac{3x_1^2+a}{2y_1}\right)^2 - 2x_1 = \dfrac{(3x_1^2+a)^2 - 8x_1y_1^2}{4(x_1^3+ax_1+b)} =$

$$= \frac{9x_1^4 + 6x_1^2a + a^2 - 8x_1^4 - 8ax_1^2 - 8x_1b}{4(x_1^3+ax_1+b)} = \frac{x_1^4 - 2x_1^2a + a^2 - 8bx_1}{4(x_1^3+ax_1+b)}$$

$$= \frac{(x_1^2-a)^2 - 8bx_1}{4(x_1^3+ax_1+b)}$$

The result of the Montgomery Ladder computations is a projective point without y-coordinate. We need to recover it.

We know x_1, y_1, x_2, x_3 where $Q[0] = (x_2, y_2)$

$Q[1] = (x_3, y_3)$

$P = (x_1, y_1)$ and $Q[1] = Q[0] + P$

We need to find y_2.

As we remember from the first lectures:

$$x_3 = \lambda^2 - x_1 - x_2 \Rightarrow \lambda^2 = x_1 + x_2 + x_3$$

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

$$y_2^2 + y_1^2 - 2y_1y_2 = (x_2-x_1)^2(x_1+x_2+x_3)$$

$$2y_1y_2 = y_1^2 + y_2^2 - (x_2 - x_1)^2(x_1 + x_2 + x_3)$$

(9) $$y_2 = (2y_1)^{-1} \cdot \left(y_1^2 + y_2^2 - (x_2^2 + x_1^2 - 2x_1x_2)(x_1 + x_2 + x_3) \right) =$$

$$= (2y_1)^{-1}(y_1^2 + y_2^2 - x_2^2x_1 - x_2^3 - x_2^2x_3 - x_1^3 - x_1^2x_2 - x_1^2x_3 + 2x_1^2x_2$$

$$+ 2x_1x_2^2 + 2x_1x_2x_3) = (2y_1)^{-1}(x_1^3 + ax_1 + b - x_1^3 + x_1^2x_2 - x_1^2x_3$$

$$+ x_2^3 + ax_2 + b - x_2^3 + x_1x_2^2 - x_3x_2^2 + 2x_1x_2x_3) =$$

$$= (2y_1)^{-1}\left(a(x_1 + x_2) + 2b + (x_1 + x_2)x_1x_2 - x_3(x_1^2 + x_2^2 - 2x_1x_2) \right)$$

$$= (2y_1)^{-1}\left((a + x_1x_2)(x_1 + x_2) + 2b - x_3(x_1 - x_2)^2 \right)$$

Now we are going to introduce the xECADDDBL, and yRecovering procedures from Appendices A.4, A.5 of the paper

"Improved Elliptic Curve Multiplication Methods Resistant against Side Channel Attacks", Indocrypt 2002

To do so we need to move the formulas (7), (8), (9) to the projective coordinates.

(7): $$\frac{X_3}{Z_3} = \frac{2\left(\frac{x_1}{z_1} + \frac{x_2}{z_2}\right)\left(\frac{x_1}{z_1} \cdot \frac{x_2}{z_2} + a\right) + 4b}{\left(\frac{x_1}{z_1} - \frac{x_2}{z_2}\right)^2} - x =$$

$$= \frac{Z_1^2 Z_2^2}{Z_1^2 Z_2^2} \cdot \frac{2\left(\frac{X_1}{Z_1} + \frac{X_2}{Z_2}\right)\left(\frac{X_1}{Z_1}\cdot\frac{X_2}{Z_2} + a\right) + 4b}{\left(\frac{X_1}{Z_1} - \frac{X_2}{Z_2}\right)^2} - x =$$

$$= \frac{2(X_1 Z_2 + X_2 Z_1)(X_1 X_2 + a Z_1 Z_2) + 4b Z_1^2 Z_2^2}{(X_1 Z_2 - X_2 Z_1)^2} - x$$

$$= \frac{2(X_1 Z_2 + X_2 Z_1)(X_1 X_2 + a Z_1 Z_2) + 4b Z_1^2 Z_2^2 - x(X_1 Z_2 - X_2 Z_1)^2}{(X_1 Z_2 - X_2 Z_1)^2}$$

(8'):

$$\frac{X_4}{Z_4} = \frac{\left(\left(\frac{X_1}{Z_1}\right)^2 - a\right)^2 - 8b\frac{X_1}{Z_1}}{4\left[\left(\frac{X_1}{Z_1}\right)^3 + a\left(\frac{X_1}{Z_1}\right) + b\right]} \cdot \frac{Z_1^4 Z_2^4}{Z_1^4 Z_2^4} = \frac{[X_1^2 Z_2^2 - a Z_1^2 Z_2^2]^2 - 8b X_1 Z_1^3 Z_2^4}{4[X_1^3 Z_1 Z_2^4 + a X_1 Z_1^3 Z_2^4 + b Z_1^4 Z_2^4]}$$

$$= \frac{[X_1^2 Z_2^2 - a Z_1^2 Z_2^2]^2 - 8b X_1 Z_1^3 Z_2^4}{4[X_1 Z_1 (X_1^2 Z_2^4 + a Z_1^2 Z_2^4) + b Z_1^4 Z_2^4]}$$

this point will be doubled

xECADDDBL:

Input: $(X_1, Z_1), (X_2, Z_2), x, a, b$

Output: $(X_3, Z_3), (X_4, Z_4)$

$R_1 := X_1$

$R_2 := Z_1$

$R_3 := X_2$

$R_4 := Z_2$

we check correctness

$R_6 := R_1 R_4 = X_1 Z_2$

$$R_1 := R_1 \cdot R_3 = X_1 \cdot X_2$$

$$R_4 := R_2 \cdot R_4 = Z_1 \cdot Z_2$$

$$R_2 := R_3 \cdot R_2 = X_2 Z_1$$

$$R_3 := R_6 - R_2 = X_1 Z_2 - X_2 Z_1$$

$$\underline{R_3} := R_3^2 = (X_1 Z_2 - X_2 Z_1)^2$$

$$R_5 := x \cdot R_3 = x \cdot (X_1 Z_2 - X_2 Z_1)^2$$

$$R_7 := a R_4 = a Z_1 Z_2$$

$$R_1 := R_1 + R_7 = X_1 \cdot X_2 + a Z_1 Z_2$$

$$R_2 := R_2 + R_6 = X_2 \cdot Z_1 + X_1 \cdot Z_2$$

$$R_1 := R_1 \cdot R_2 = (X_1 \cdot X_2 + a Z_1 Z_2) \cdot (X_2 Z_1 + X_1 Z_2)$$

$$R_2 := R_4^2 = (Z_1 \cdot Z_2)^2$$

$$R_7 := b R_2 = b (Z_1 \cdot Z_2)^2$$

$$R_1 := R_1 + R_7 = (X_1 \cdot X_2 + a Z_1 Z_2)(X_2 Z_1 + X_1 Z_2) + b (Z_1 Z_2)^2$$

$$R_1 := R_1 + R_1 = 2 (X_1 \cdot X_2 + a Z_1 Z_2)(X_2 Z_1 + X_1 Z_2) + 2 b (Z_1 Z_2)^2$$

$$R_5 := R_1 - R_5 = 2 (X_1 \cdot X_2 + a Z_1 Z_2)(X_2 Z_1 + X_1 Z_2) + 2 b (Z_1 Z_2)^2 - x (X_1 Z_2 - X_2 Z_1)^2$$

$$R_5 := R_7 + R_5$$

$$\underline{R_5} := R_7 + R_5 = 2 (X_1 X_2 + a Z_1 Z_2)(X_2 Z_1 + X_1 Z_2) + 4 b (Z_1 Z_2)^2 - x (X_1 Z_2 - X_2 Z_1)^2$$

$$R_2 := a R_2 = a (Z_1 Z_2)^2$$

$$R_1 := R_6^2 = (X_1 Z_2)^2$$

$$R_1 := R_1 + R_2 = (X_1 Z_2)^2 + a (Z_1 Z_2)^2$$

$$R_2 := R_2 + R_2 = 2a (Z_1 Z_2)^2$$

$$R_2 := R_1 - R_2 = (X_1 Z_2)^2 - a(Z_1 Z_2)^2$$

$$R_2 := R_2^2 = \left((X_1 Z_2)^2 - a(Z_1 Z_2)^2\right)^2$$

$$R_1 := R_6 R_1 = X_1 Z_2 \cdot \left((X_1 Z_2)^2 + a(Z_1 Z_2)^2\right)$$

$$R_7 := R_4 R_7 = Z_1 Z_2 \cdot b(Z_1 Z_2)^2 = b(Z_1 Z_2)^3$$

$$R_1 := R_1 + R_7 = X_1 Z_2 \left((X_1 Z_2)^2 + a(Z_1 Z_2)^2\right) + b(Z_1 Z_2)^3$$

$$R_7 := R_6 \cdot R_7 = X_1 Z_2 \cdot b(Z_1 Z_2)^3 = b X_1 Z_1^3 Z_2^4$$

$$R_7 := R_7 + R_7$$

$$R_7 := R_7 + R_7$$

$$R_7 := R_7 + R_7 = 8b X_1 Z_1^3 Z_2^4$$

$$R_7 := R_2 - R_7 := \left((X_1 Z_2)^2 - a(Z_1 Z_2)^2\right)^2 - 8b X_1 Z_1^3 Z_2^4$$

$$R_6 := R_4 \cdot R_1 = Z_1 \cdot Z_2 \left[X_1 Z_2 \left((X_1 Z_2)^2 + a(Z_1 Z_2)^2\right) + b(Z_1 Z_2)^3\right] =$$

$$= X_1 Z_1 Z_2^2 \left((X_1 Z_2)^2 + a(Z_1 Z_2)^2\right) + b(Z_1 Z_2)^4 =$$

$$= X_1 Z_1 \left(X_1^2 Z_2^4 + a Z_1^2 Z_2^4\right) + b(Z_1 Z_2)^4$$

$$R_6 := R_6 + R_6$$

$$R_6 := R_6 + R_6 = 4\left[X_1 Z_1 \left(X_1^2 Z_2^4 + a Z_1^2 Z_2^4\right) + b(Z_1 Z_2)^4\right]$$

$$X_3 := R_5$$

$$Z_3 := R_3$$

$$X_4 := R_7$$

$$Z_4 := R_6$$

Now its turn to introduce y Recovering procedure.

$$(9'): \quad \frac{y_2}{z_2} = \frac{1}{2y_1}\cdot\left(\left(a + x_1\frac{x_2}{z_2}\right)\left(x_1 + \frac{x_2}{z_2}\right) + 2b - \frac{x_3}{z_3}\left(x_1 - \frac{x_2}{z_2}\right)^2\right)$$

$$= \frac{1}{2y_1 z_2^2 z_3}\left((az_2 + x_1x_2)(x_1z_2 + x_2)\cdot z_3 + 2bz_2^2z_3 - x_3(x_1z_2 - x_2)^2\right)$$

y Recovering

So in y Recovering $P(x_1, y_1) = (x, y)$

Input:
- $Q[0] = (x_2, z_2)$
- $Q[1] = (x_3, z_3)$
- x, y, a, b

Output: (x_2', y_2', z_2') - projective coordinates of $Q[0]$

$R_1 := x_2$

$R_2 := z_2$

$R_3 := x_3$

$R_4 := z_3$

we check the computations

$R_5 := x\,R_2 = x z_2 = x_1 z_2$

$R_6 := R_5 - R_1 = x z_2 - x_2 = x_1 z_2 - x_2$

$R_6 := R_6^2 = (x_1 z_2 - x_2)^2$

$R_6 := R_3 \cdot R_6 = x_3 \cdot (x_1 z_2 - x_2)^2$

$R_5 := R_5 + R_1 = x_1 z_2 + x_2$

$R_7 := x R_1 = x_1 x_2$

$R_1 := R_1 \cdot R_2 = X_2 \cdot Z_2$

$R_3 := a R_2 = a Z_2$

$R_2 := R_2^2 = Z_2^2$

$R_7 := R_3 + R_7 = a Z_2 + X_1 X_2$

$R_7 := R_5 \cdot R_7 = (X_1 Z_2 + X_2)(a Z_2 + X_1 X_2)$

$R_5 := y \cdot R_4 = y_1 Z_3$

$R_5 := R_5 + R_5 = 2 y_1 Z_3$

$\underline{\underline{R_3 := R_5 \cdot R_2 = 2 y_1 Z_3 \cdot Z_2^2}}$

$R_1 := R_5 \cdot R_1 = 2 y_1 Z_3 \cdot X_2 Z_2 = 2 y_1 Z_2 Z_3 \cdot X_2$

$\underline{\underline{R_2 := b \cdot R_2 = b Z_2^2}}$

$R_2 := R_2 + R_2 = 2 b Z_2^2$

$R_7 := R_7 + R_2 = (X_1 Z_2 + X_2)(a Z_2 + X_1 X_2) + 2 b Z_2^2$

$R_7 := R_4 \cdot R_7 = Z_3 (X_1 Z_2 + X_2)(a Z_2 + X_1 X_2) + 2 b Z_2^2 Z_3$

$\underline{\underline{R_7 := R_7 - R_6 = (a Z_2 + X_1 X_2)(X_1 Z_2 + X_2) \cdot Z_3 + 2 b Z_2^2 Z_3 - X_3 (X_1 Z_2 - X_2)^2}}$

$X_2' := R_1$

$Y_2' := R_7$

$Z_2' := R_3$

Let P be the input to the last version of the Montgomery Ladder algorithm (the version that utilizes x ECDBL, xECADDDBL procedures).

Assume that $P=(x,y)$ does not satisfy the elliptic curve E equation

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

but the equation

$$(10) \quad dy^2 = x^3 + ax + b$$

of the twist E^d, for some fixed d being QN in $\mathbb{F}_p^*$ (quadratic non-residue in $\mathbb{F}_p^*$).

It is easy to find such point P:

Define $f_{a,b}(x) = x^3 + ax + b$

1) If for a random $x \in \mathbb{F}_p$ $\omega = f_{a,b}(x)$ is such that $\omega = 0$ or ω is QR in $\mathbb{F}_p^*$ (quadratic residue in $\mathbb{F}_p^*$) then we set

$$y = \sqrt{\omega} \quad \text{or} \quad y = -\sqrt{\omega} = p - \sqrt{\omega}.$$

See that $y \in \mathbb{F}_p$ and $(x,y) \in E$

2) If $\omega = f_{a,b}(x)$ is QN in $\mathbb{F}_p^*$ then it means that ω is

an odd power of a generator g of $\mathbb{F}_p^*$ (we know that for each finite field its multiplicative group is cyclic, so there exists $g \in \mathbb{F}_p^*$ such that $\langle g \rangle = \mathbb{F}_p^*$; let us fix such g), that is

$$w = g^l, \quad l \text{ is odd}$$

The same applies to d defining E^d (for any fixed d such that d is QN in $\mathbb{F}_p^*$), that is

$$d = g^k, \quad k \text{ is odd}$$

Then $w \cdot d^{-1} = g^{l-k}$ is an even power of the generator g, so it is a QR in $\mathbb{F}_p^*$.

We set $y = \sqrt{w \cdot d^{-1}}$ or $y = -\sqrt{w \cdot d^{-1}} = p - \sqrt{w \cdot d^{-1}}$.

See that $y \in \mathbb{F}_p^*$ and (x, y) satisfies (10), the equation for E^d.

What is the probability of picking a random x from $\mathbb{F}_p^*$ such that there exists $y \in \mathbb{F}_p^*$ such that $(x, y) \in E^d$?

Let assume that $f(x)$ behaves pseudorandomly, that is for random $x \in \mathbb{F}_p^*$ we get a random $w = f(x)$ such that $w \in \mathbb{F}_p^*$. Note that the polynomial $f(x) = x^3 + ax + b$ has t-roots: $x_1, \ldots, x_t$, $0 \le t \le 3$, so for large p it is

very improbable to hit one of $t \leq 3$ elements $x_1, \ldots, x_t$ when we take random $x \in \mathbb{F}_p^*$).

Recall that $\frac{p-1}{2}$ elements of $\mathbb{F}_p^*$ are QR (are even powers of g) and $\frac{p-1}{2}$ elements of $\mathbb{F}_p^*$ are QN (are odd powers of g).

Hence for a random $w = f(x) \in \mathbb{F}_p^*$ we estimate the probability that w is a QN by

$$\approx \frac{\frac{p-1}{2}}{p-1} = \frac{1}{2}$$

That is, for large p and a random $x \in \mathbb{F}_p^*$ we get, with probability $\approx \frac{1}{2}$, a point $P = (x, y) \in E^d$ such that $y \in \mathbb{F}_p^*$.

If such point P is an input to the above-mentioned Montgomery Ladder algorithm then the implementation does not notice this:

- there is no check that the input point P belongs to E,
- the procedures xECDBL, xECADDDBL do not make use of y-coordinate,
- the YRecovering procedure will yield us $(X:Y:Z)$ such that $(x, y) = (X/Z, Y/Z)$ is on E^d, but there is no check that the output point belongs to E.

But if the input to the Montgomery ladder is $P \in E^d$ then scalar multiplication $s \cdot P$ carried out by the Algorithm is multiplication on E^d.

For a random E the number of points of E^d is random as well. If the twist E^d is not checked against primality of its cardinality, then the size of the largest factor of $|E^d|$ is smaller than $|E^d|^{2/3}$ with probability at least $1/2$.

So due to the Pohlig-Hellman algorithm (discussing this algorithm is outside the scope of this lecture) and the Pollard-ρ method: for random E the complexity of the attack on s when $P \in E^d$ and $s \cdot P \in E^d$ are known to the attacker equals, with probability at least $\frac{1}{2}$:

$$O\left(\sqrt{|E^d|^{2/3}}\right) = O\left(|E^d|^{1/3}\right)$$

and the attack reconstructs s modulo powers of primes dividing $|E^d|$. From CRT we can reconstruct s modulo the product of these powers of primes.

For any elliptic curve we have the following

Theorem: (group structure of an elliptic curve)

Let E be an elliptic curve defined over $\mathbb{F}_q$.

Then $E(\mathbb{F}_q)$ is isomorphic to the cartesian product $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}$ for some $n_1, n_2 \in \mathbb{Z}$

such that $n_2 \mid \gcd(n_1, q-1)$.

Consequently, $|E(\mathbb{F}_q)| = n_1 \cdot n_2$. If $n_2 = 1$ then $E(\mathbb{F}_q)$ is cyclic.

So in our case $|E^d| = n_1 \cdot n_2$ and $n_2 \mid \gcd(n_1, p-1)$

Thus the attack (or a few iterations of it) reconstructs

$$s \bmod n_1$$

Denote by r the value $s \bmod n_1$. That is

$$(11) \quad s = Q \cdot n_1 + r \quad \text{for some } Q \in \mathbb{Z}$$

We know that $s < |E|$

We also know that:

$$|E| = p + 1 - c$$

$$|E^d| = p + 1 + c \quad \text{where } |c| \leq 2\sqrt{p}$$

It may happen that $|E| > |E^d|$ or that $|E^d| > |E|$

But it is easy to see that for sufficiently large p we certainly have:

$$2|E^d| \geq |E|$$

Indeed, the inequality above is equivalent to:

$$2(p+1+c) \geq p+1-c$$

which in turn is equivalent to:

$$p+1 \geq -3c$$

The last one is satisfied for $p+1 \geq 6\sqrt{p}$, because $2\sqrt{p} \geq -c$.
But $p+1 \geq 6\sqrt{p}$ is satisfied already for $p=37$.

Since $|E| > s$ it suffices to perform at most $2 \cdot n_2$ brute-force steps to reconstruct Q in (11).

Usually we take a standardized elliptic curve, not a random one. The paper "Fault Attack on Elliptic Curve with Montgomery Ladder Implementation" summarizes the curves proposed by NIST and SECG and shows that e.g., complexity of the above attack on secp224r1 curve is equal to 2^{59} group operations.
Interestingly, Brainpool curves are immune to this attack because the curves are chosen so that both $|E|$ and $|E^d|$ are prime.

A classical countermeasure against this attack is to check at the end of the computations of $s \cdot P$ whether the result belongs to E and to return an error if it does not.

The paper shows how, under some plausible assumptions, to deal with such a countermeasure by injecting faults. For details we refer the reader to the paper.