ECC school: The Montgomery ladder in SageMath

Aurore Guillevic

Academia Sinica, Taipei, Taiwan
October 28, 2024

1. The Montgomery ladder in SageMath on curve25519

We will implement the Montgomery ladder for constant-time scalar multiplication on an elliptic curve. The general Montgomery curve, for $B \neq 0$ and $A \neq \pm 2$, is

$$E^M : By^2 = x^3 + Ax^2 + x \text{ over a finite field } \mathbb{F}_q \text{ of characteristic } p \geq 5 \ . \tag{1}$$

For testing in SageMath, we need a correspondance with a leading coefficient of y^2 to be 1. Let us neutralise the B coefficient of y^2 by dividing the curve equation by $B^3 \in \mathbb{F}_q$:

$$\begin{aligned}
\frac{E^M}{B^3} : \frac{By^2}{B^3} &= \frac{x^3}{B^3} + \frac{Ax^2}{B^3} + \frac{x}{B^3} \\
&\iff \left(\frac{y}{B}\right)^2 = \left(\frac{x}{B}\right)^3 + \frac{A}{B}\left(\frac{x}{B}\right)^2 + \frac{1}{B^2}\frac{x}{B} \\
&\iff y'^2 = x'^3 + \frac{A}{B}x'^2 + \frac{1}{B^2}x' \text{ with } x' = x/B,\ y' = y/B
\end{aligned}$$

therefore there is a $\mathbb{F}_q$-rational isomorphism between E^M and E'^M

$$E'^M : y'^2 = x'^3 + \frac{A}{B}x'^2 + \frac{1}{B^2}x'$$

$$\begin{aligned}
i : E^M &\to E'^M \\
(x, y) &\mapsto (x/B, y/B)
\end{aligned}$$

and the inverse is

$$\begin{aligned}
i^{-1} : E'^M &\to E^M \\
(x', y') &\mapsto (x \cdot B, y \cdot B)
\end{aligned}$$

We will use the representation E'^M for tests in SageMath as follows. Let E be the `curve25519` curve:

```
p = ZZ(2**255-19)
Fp = GF(p)
# Montgomery form is y^2 = x^3 + 486662*x^2 + x
A = Fp(486662)
B = Fp(1)
EM = EllipticCurve([0, A/B, 0, 1/B**2, 0])
```

The Montgomery form of elliptic curve is not competitive compared to the short Weierstrass form with the double-and-add algorithm. However Peter L. Montgomery observed that skipping the y-coordinate and using projective X, Z-coordinates, the scalar multiplication becomes competitive.

In this part, we will implement the group law in X, Z-coordinates, then the Montgomery ladder for scalar multiplication. The file `ladder_skeleton.py` contains the addition and doubling in affine and projective coordinates on E^M with test functions, to serve as an example. **Download the file `ladder_skeleton.py` and write the answers to the questions as SageMath functions in this file. Test with `ladder_tests.py`**

Question 1. Implement the x-only addition and doubling in x-only affine coordinates according to the formulas:

Let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two points on E^M. Let $P_1 + P_2 = (x_3, y_3)$ and $P_1 - P_2 = (x_4, y_4)$. Assume that $x_1 \neq x_2$, $x_1 \neq 0$ or $x_2 \neq 0$, and $x_4 \neq 0$. From Montgomery's formulas one has

$$x_3 x_4 (x_1 - x_2)^2 \quad = \quad (x_1 x_2 - 1)^2 \tag{2}$$

and for $P_1 = P_2$, with $x_1 \neq 0$,

$$4x_1x_3(x_1^2 + Ax_1 + 1) = (x_1^2 - 1)^2 \ . \tag{3}$$

Knowing $P_1, P_2, P_1 - P_2$, one can deduce $P_3 = P_1 + P_2$ for $P_1 \neq P_2$:

$$x_3 \quad = \quad \frac{(x_1x_2 - 1)^2}{x_4(x_1 - x_2)^2} \tag{4}$$

and for doubling P_1:

$$x_3 = \frac{(x_1^2 - 1)^2}{4x_1(x_1^2 + Ax_1 + 1)} \ . \tag{5}$$

The functions whose header are given below are sketched in the PYTHON file of the hand-in, complete these functions in the file:

```
def add_affine_x_only(x1, x2, x4):
def double_affine_x_only(x1, A):
```

Question 2. Test your functions of the previous question. The functions assume that the inputs are not $\mathcal{O}$ nor points of order 2, more precisely: $x_1 \neq x_2$, $x_1 \neq 0$ or $x_2 \neq 0$, and $x_4 \neq 0$.

Question 3. Implement the x-only addition and doubling in X, Z-projective coordinates, based on the affine coordinates. It means to avoid the divisions, you will have two coordinates (X, Z) such that the correspondance with the affine coordinates is $x = X/Z$ for non-zero Z, and if $P(X, Z = 0)$, then P corresponds to the point at infinity $\mathcal{O}$.

Remember that you can use the Elliptic Curve Formula Database at `http://www.hyperelliptic.org/EFD/` to check your answers.

Use these function names:

```
def add_proj_x_only(X1, Z1, X2, Z2, X4, Z4):
def double_proj_x_only(X1, Z1, A):
```

Question 4. Test your functions of the previous question.

Montgomery's binary scalar multiplication is given in Algorithm 1. Montgomery observed that at each step, the difference $R_0 - R_1$ is always equal to P.

Algorithm 1: Montgomery's binary scalar multiplication

Input: $m = \sum_{i=0}^{n-1} b_i 2^i$ with $b_{n-1} = 1$, and point $P \in E^M$ in affine coordinates

Output: $[m]P$

$(R_0, R_1) \leftarrow (P, [2]P)$
for $i = n - 2$ *down to* 0 **do**
  if $b_i = 0$ **then**
    $(R_0, R_1) \leftarrow ([2]R_0, R_0 + R_1)$
  else
    $(R_0, R_1) \leftarrow (R_0 + R_1, [2]R_1)$
return R_0

This gives the Montgomery ladder in Algorithm 2

Algorithm 2: Montgomery's ladder for scalar multiplication

Input: $m = \sum_{i=0}^{n-1} b_i 2^i$ with $b_{n-1} = 1$, and x_P affine x-coordinate of point $P \in E^M$

Output: x-coordinate of $[m]P$

$(x_0, x_1) \leftarrow (x_P, \texttt{double\_affine\_x\_only}(x_P, A))$
for $i = n - 2$ *down to* 0 **do**
  if $b_i = 0$ **then**
    $(x_0, x_1) \leftarrow (\texttt{double\_affine\_x\_only}(x_0), \texttt{add\_affine\_x\_only}(x_0, x_1, x_P)$
  else
    $(x_0, x_1) \leftarrow (\texttt{add\_affine\_x\_only}(x_0, x_1, x_P), \texttt{double\_affine\_x\_only}(x_1, A)$
return x_0

Question 5. Implement the Montgomery ladder, using the two affine functions of Question 1.

Question 6. Test your function of the previous question.

Question 7. Implement the Montgomery ladder, using the two projective functions of Question 3.

Question 8. Test your function of the previous question.