Nov 11, 2024

ZK Hub, Bangkok

On proving scalar multiplications in SNARKs

Youssef El Housni
(Joint work with Thomas Piellard)

Linea

Youssef El Housni

- Cryptographer at Consensys
- Co-maintainer of gnark
- Co-developer of Linea

Outline

1. Motivation
2. Scalar multiplication
3. Scalar multiplication in SNARKs
   a. Fake GLV
   b. 4D fake GLV

Motivation

ECC

Elliptic curves cryptography (ECC) is used for **key agreement**, **digital signatures**, **pseudo-random generators** and **(zk) SNARKs**

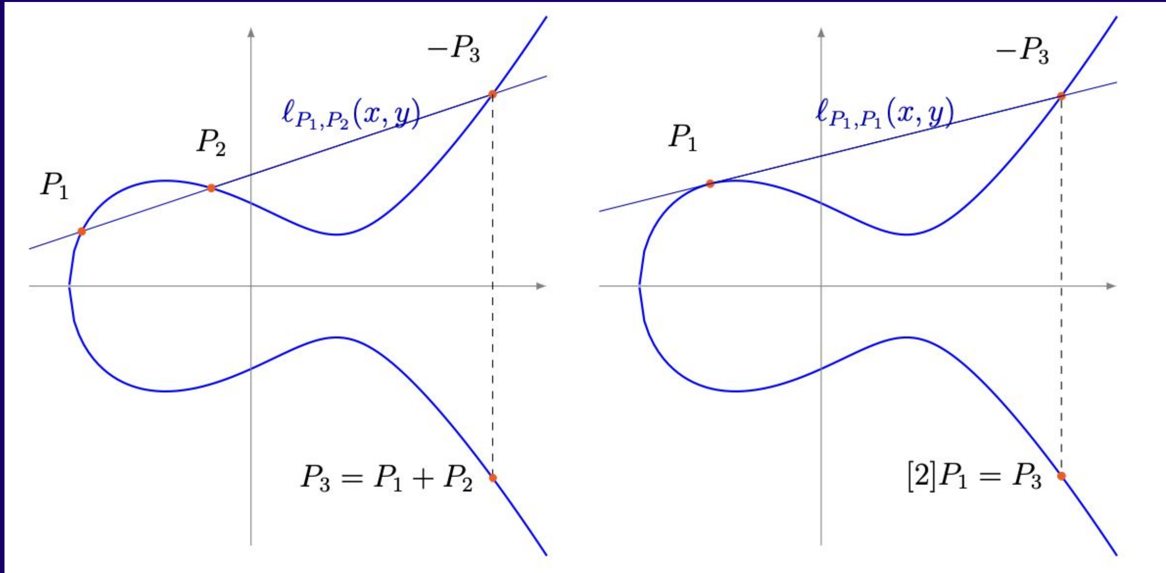
Proving ECC

- SNARK recursion
- zkEVM
- Account abstraction
- Verkle trie

Linea

ECC

E(Fp): $y^2=x^3+ax+b$ and r | #E



(a) "Chord": $P_1 + P_2$ (b) "Tangent": $2P$

Operations on E(Fp)[r]:

- Addition:
  P1 + P2 = P3
- Doubling:
  [2]P1 = P1+P1 = P3
- **Scalar multiplication:**
  [n]P = P + P + … + P
  (n times)

Linea

Proving ECC

SNARK recursion

(Linea)

BLS12-377

Proof of a proof:

BW6-761

- 1st proof verification requires scalar multiplication
- 2nd proof generation requires proving previous scalar multiplications

zkEVM

- ECDSA signatures on **secp256k1** curve
- **BN254*** precompile (ECMUL)
- Aggregation (SNARK recursion)

* soon **BLS12-381** too in Pectra

Account abstraction

- ECDSA signatures on **P-256** or **Ed25519**

Verkle trie

- (multi) Scalar multiplications on **Bandersnatch** curve

Linea

Standard scalar multiplication

left-to-right double-and-add

INPUT: s = (s_{t−1},..., s_1, s_0), P ∈ E(Fp).
OUTPUT: [s]P.

1. Q ← ∞.
2. For i from t−1 downto 0 do
   2.1 Q ← 2Q.
   2.2 If s_i = 1 then Q ← Q + P.
3. Return(Q).

- **secp256k1**
- **P-256**
- **Ed25519**
- **BN254**
- **BLS12-381**
- **BLS12-377**
- **BW6-761**
- **Bandersnatch**

GLV endomorphism

Example 1:
Curves of the form E: $y^2=x^3+b$ (a=0, D=3)

P(x,y) in E : **ϕ(P) = [λ]P** *for some fixed* ***λ***
ϕ(P) = (wx, y) *for some fixed* ***w***

- **secp256k1**
- **BN254**
- **BLS12-381**
- **BLS12-377**
- **BW6-761**

Example 2:
Curves with D=8

P(x,y) in E : **ϕ(P) = [λ]P** *for some fixed λ*
$\phi(P) = (u^2(x^2+wx+t) / (x+w), y(x^2+2wx+v) / (x+w)^2)$
for some fixed ***u, v, w, t***

- **Bandersnatch**

Linea

GLV scalar multiplication

How to compute [s]P?

- Write s as s1 + λ s2 mod r with s1, s2 < $\sqrt{r}$
- [s]P = [s1]P + [λ s2]P = **[s1]P + [s2]ϕ(P)**
- Use Strauss-Shamir trick to compute [s1]P + [s2]ϕ(P) simultaneously

INPUT: s and P $\in$ E(Fp).
OUTPUT: [s]P.

1. Find s1 and s2 s.t. s = s1 + λ * s2 mod r
   1.1 let s1 = (s1_{t−1},..., s1_1, s1_0)
   1.2 and s2 = = (s2_{t−1},..., s2_1, s2_0)
2. P1 $\leftarrow$ P, P2 $\leftarrow$ ϕ(P), P3 $\leftarrow$ P1+P2 and Q $\leftarrow$ P3.
3. For i from t−1 downto 0 do
   3.1 Q $\leftarrow$ 2Q.
   3.2 If s1_i = 0 and s2_i = 0 then Q $\leftarrow$ Q.
   3.3 If s1_i = 1 and s2_i = 0 then Q $\leftarrow$ Q + P1.
   3.4 If s1_i = 0 and s2_i = 1 then Q $\leftarrow$ Q + P2.
   3.5 If s1_i = 1 and s2_i = 1 then Q $\leftarrow$ Q + P3.
4. Return(Q).

Linea

Scalar multiplication in SNARKs

right-to-left double-and-add

INPUT: s = (s_{t−1},..., s_1, s_0), P ∈ E(Fp).
OUTPUT: [s]P.

1. Q ← P.
2. For i from 1 to t−1 do
   2.1 If s_i = 1 then Q ← Q + P.
   2.2 P ← 2P.
3. if s_0 = 0 then Q ← Q - P
4. Return(Q).

GLV-like

INPUT: s and P ∈ E(Fp).
OUTPUT: [s]P.

1. Find s1 and s2 s.t. s = s1 + λ * s2 mod r
   1.1 let s1 = (s1_{t−1},..., s1_1, s1_0)
   1.2 and s2 = = (s2_{t−1},..., s2_1, s2_0)
2. Q ← [2](P+ϕ(P)).
3. For i from t−1 downto 0 do
   3.1 If s_{2i+1} = 1 then S ← [2s_{2i}-1]P.
   3.2 S ←ϕ([2s_{2i}-1]P).
4. Q ← [2]Q + S
4. Return(Q).

Optimized implementation in gnark/std/algebra/emulated/sw_emulated

Scalar multiplication in SNARKs

right-to-left double-and-add

- P-256
- Ed25519

GLV-like

- secp256k1
- BN254
- BLS12-381
- BLS12-377
- BW6-761
- Bandersnatch

Fake GLV

GLV: **[s]P** (s on n bits) → **[s1]P + [s2]ϕ(P)** (s1, s2 on n/2 bits)

- Instead of proving that [s]P = Q we prove that [s]P-Q = O
- Write s = u/v mod r with u, v < √r
- Prove that [v*s]P - [v]Q = v*O or **[u]P - [v]Q = O** (u, v on n/2 bits)

Solution: half-GCD algorithm (i.e. running GCD half-way)

https://hackmd.io/@yelhousni/fake-glv

Benchmarks: Fake GLV

Emulated scalar multiplication in a BN254-PLONK:

P-256	Old (Joye07)	New (fake GLV)
[s]P	738,031 scs 186,466 r1cs	385,412 scs 100,914 r1cs
ECDSA verification	1,135,876 scs 293,814 r1cs	742,541 scs 195,266 r1cs

Linea

Initial idea of Liam Eagen

4D fake GLV

Combining the **fake GLV** with the **endomorphism**

- Find r1, r2 s.t. r | norm(r1+λr2) , i.e. r = r1+λr2

  half-GCD in $\mathbb{Z}$ (precomputed)

- Find u1, u2, v1, v2 < c*r^{¼} s.t. s = (u1+λu2) / (v1+λv2) mod (r1+λr2)

  Half-GCD in K = $\mathbb{Q}[\lambda]/f(\lambda)$ where f(λ) = 0 mod r

- K needs to be an Euclidean domain
  - Example 1: K is the ring of Eisenstein integers $\mathbb{Z}[\omega]$
  - Example 2: K = $\mathbb{Q}[\sqrt{-2}] / \lambda^2+2$

Linea

Example 1: Eisenstein Integers

- commutative ring of algebraic integers in the algebraic number field $\mathbb{Q}(\omega)$ (the third cyclotomic field), i.e. $\mathbb{Z}[\omega]$.
- Of the form $z = a + b\omega$, where a and b are integers and ω is a primitive third root of unity i.e. $\omega^2+\omega+1 = 0$.
- Mul: $(x0+x1\omega)(y0+y1\omega) = (x0y0-x1y1) + (x0y1+x1y0-x1y1)\omega$
- Norm$(x0+x1\omega) = x0^2 + x1^2 - x0*x1$
- Quotient(x, y) = Re(x*conj(y))/Norm(y) + ω Im(x*conj(y))/Norm(y)
- c = log_(3/sqrt(3)))(r). For 128-bit security n/4+9 bits.

Linea

Benchmarks: 4D fake GLV

Emulated scalar multiplication in a BN254-PLONK:

scalar mul	old ordinary GLV (scs)	new 4D fake GLV (scs)
secp256k1	385,461	282,223
BN254	381,467	279,262
BW6-761	1,367,067	1,010,785
BLS12-381	539,973	390,294

Linea

Thank you

linea.build
gnark.io

youssef.elhousni@consensys.net
gnark@consensys.net

X: @YoussefElHousn3
TG: @ElMarroqui
GH: @yelhousni