Elliptic curves, number theory and cryptography

5. handin – Isogenies, Pairings

Aurore Guillevic and Diego F. Aranha

Aarhus University

April 21, 2022

Due date: May 6, 4pm GMT+2 (16:00 Aarhus time)

The corresponding lecture materials are at
`https://www.hyperelliptic.org/tanja/teaching/isogeny-school21/`
and Lorenz Panny materials at
`https://yx7.cc/docs/misc/isog_bristol_notes.pdf`

Question 1 is the follow-up of Question 5 in hand-in 4.

Question 1 (SIKE on a toy-example)**.** Let $p = 431$ and note that $p+1 = 432 = 2^4 \cdot 3^3$. The curve $E_0 : y^2 = x^3 + x$ is a supersingular curve over $\mathbb{F}_p$ and has $p+1$ points. Consider the curve over $\mathbb{F}_{p^2}$ where it has $(p+1)^2$ points.

```
p = 431
Fp = GF(p)
A = Fp(0)
E0 = EllipticCurve(Fp, [1, 0])
E0.is_supersingular()
Fpz.<z> = Fp[]
Fp2.<i> = Fp.extension(z^2+1)
E0p2 = E0.base_extend(Fp2)
r2 = 2^4
r3 = 3^3
assert r2 * r3 == p+1
```

(a) Find a basis of the 2^4-torsion and a basis of the 3^3-torsion subgroups, *i.e.*, find points $P \in E(\mathbb{F}_p)$ and $Q \in E(\mathbb{F}_{p^2})$ of order 2^4 such that $\langle P \rangle \cap \langle Q \rangle = \mathcal{O}$ and points $R \in E(\mathbb{F}_p)$ and $S \in E(\mathbb{F}_{p^2})$ of order 3^3 such that $\langle R \rangle \cap \langle S \rangle = \mathcal{O}$.
Hint: You can check this as $[8]P \neq [8]Q$ and $[9]R \neq \pm[9]S$.
Hint: For the 3^3 torsion points, you can also use how the negative direction is defined for CSIDH to find the independent points.

(b) Alice and Bob.
Compute a generator $P_a \in E(\mathbb{F}_{p^2})$ for the kernel of Alice's isogeny, where $P_a = P + [a]Q$ and a is a random integer in $\{1, \ldots, 2^4 - 1\}$.
Compute a generator $P_b \in E(\mathbb{F}_{p^2})$ for the kernel of Bob's isogeny, where $P_b = R + [b]S$ and b is a random integer in $\{1, \ldots, 3^3 - 1\}$.
Check that P_a has order 2^4 and P_b has order 3^3 (without using `.order()`). If not, it means you have a problem with your basis, go back to Question (a).

(c) Isogenies of Alice and Bob.
With the function `phiA = E0p2.isogeny(Pa)`, compute Alice's isogeny ϕ_a and the isogenous curve E_a with `phiA.codomain()`.
Do the same for Bob with Bob's generator: compute Bob's isogeny ϕ_b and the isogenous curve E_b.

(d) Compute the image of P and Q under ϕ_b, then compute Alice's $\phi_b(P_a) = \phi_b(P) + [a]\phi_b(Q)$ in E_b.
Compute the image of R and S under ϕ_a, then compute Bob's $\phi_a(P_b) = \phi_a(R) + [b]\phi_a(S)$ in E_a.

(e) Compute the second part of the commutative diagram:
- compute an isogeny of kernel $\phi_b(P_a)$ from E_b, and the image curve E_{ba}.
- compute an isogeny of kernel $\phi_a(P_b)$ from E_a, and the image curve E_{ab}.

Check that the j-invariants of E_{ab} and E_{ba} are equal.

Question 2. The function `cocks_pinch(l, n, D)` is provided.

Use the Cocks-Pinch method to obtain a pairing-friendly curve of embedding degree $n = 6$ and $D = -3$ from a prime number ℓ of 256 bits.

Use the Cocks-Pinch method to obtain a pairing-friendly curve of embedding degree $n = 8$ and $D = -4$ from a prime number ℓ of 256 bits.

In both cases choose ℓ then run the method. Give ℓ, p, the curve trace t and the curve equation such that E has a subgroup of prime order ℓ and embedding degree n.

Hint: `random_prime(2**256)` returns a random-looking prime of 256 bits.

Observation: The questions below refer to the slides in Week 10 and SAGE code from Week 11.

Question 3. Using the file `tate_pairing_supersingular_curve.py` from Brightspace, implement Joux's tripartite key agreement in SAGE. The file defines pairing groups for a Type-1 setting using a supersingular curve and a distortion map.

Question 4. Convert the AKE protocol due to Sakai et al. that we studied in class to the Type-3 setting. Implement it in SAGE using the groups defined in the file `ate_pairing.py` from Brightspace. **Hint:** For hashing to the pairing groups, in this question you can hash to a scalar and multiply the scalar by a generator of the group. The next question will suggest something slightly more sophisticated.

Question 5. Convert the BLS short signature scheme that we studied in class to the Type-3 setting and implement it using file `ate_pairing.py` from Brightspace . What are the possible trade-offs in terms of public key and signature size? For simplicity, assume that signed messages are integers in the base field so you can encode them as points by encoding into the x-coordinate and incrementing it until a suitable y satisfying the elliptic curve is found. Some helpful code can be found below:

```
import hashlib

def encode_msg(M, E, A, B, cofactor):
    x = Fp(Integer(hashlib.sha256(M).hexdigest(), 16))
    rhs = x**3 + A*x + B
    while not(rhs.is_square()):
        x += 1
        rhs = x**3 + A*x + B
    y = sqrt(rhs)
    return cofactor * E(x, y)
```