Elliptic Curves - List 6

Task 1 (20 pts) The task is composed of the following steps:

(a) (3 pt) Generate domain parameters: field $\mathbb{F}_p$, where $p = 1 \bmod 12$, an elliptic curve $E(\mathbb{F}_p)$ and a basepoint $G \in E(\mathbb{F}_p)$. The basepoint G must be such that ord$G$ is odd. You may use `SageMath`.

(b) (7 pts) Implement the double-and-add method to calculate the results P of scalar multiplications

$$P = k \cdot G \tag{1}$$

where k are chosen with uniform probability distribution from the set

$$\{2, 3, \ldots, (\text{ord}G) - 1\}.$$

Intermediate values and P **must be** represented in Jacobian coordinates.

(c) (10 pts) Implement the method of recovering the least significant bits of scalar k discussed during the lecture (*Projective Coordinates Leak*). More specifically, focus your attak on the last bit of scalar k (the least significant one) and try to recover it. Run your attack for at least 1000 random scalars k and measure for how many scalars the attacks was successful (has indicated the correct value of the least significant bit of k).

Hints: To implement the attack you may use `SageMath`:

(a) to calculate e.g., a cubic root of some $a \in \mathbb{F}_p$ you may search for the roots of the polynomial $f(X) = X^3 - a$. In such a case assume that the polynomial $f(X)$ belongs to the ring $\mathbb{F}_p[X]$. For `SageMath` instructions see `http://doc.sagemath.org/html/en/constructions/polynomials.html#roots-of-polynomials`

(b) Apart from running `sage` scripts you can run python scripts with `SageMath`. According to `https://doc.sagemath.org/html/en/faq/faq-usage.html#how-do-i-import-sage-into-a-python-script` you can call

```
sage -python /path/to/my/script.py
```