ELLIPTIC CURVES IN COMPUTATIONAL NUMBER THEORY

HOMEWORK 1

ENS DE LYON

Instructions. This homework is due Friday February 21, 11:59pm to the latest. Answers and code should be sent to `benjamin.wesolowski@ens-lyon.fr`.

1. Factoring and modular square roots

An *oracle* is an idealized algorithm which solves a computational problem in a single step. This notion is useful to study the relative hardness of computational problems. Consider two computational problems P_1 and P_2. Suppose that you wish to solve problem P_1, and you have access to an oracle $\mathcal{O}$ solving problem P_2. We say that *there is a (probabilistic) polynomial time reduction from problem P_1 to problem P_2* (or that *P_1 reduces to P_2*) if there exists a (probabilistic) polynomial time algorithm with access to $\mathcal{O}$ solving problem P_1. In other words, if problem P_2 is easy, then problem P_1 is also easy. We say that two problems are equivalent if there are (probabilistic) polynomial time reductions in both directions.

The goal of this exercise is to prove that the problems of factoring integers and computing modular square roots are equivalent. For simplicity, we restrict to so-called RSA integers: an RSA integer is an integer of the form pq where p and q are distinct odd prime numbers.

- Factoring: given an RSA integer $N = pq$, find the prime factors p and q.
- SquareRootMod: given an RSA integer N, and an arbitrary integer x, find an integer y such that $y^2 = x \bmod N$, or assert that none exists.

Question 1.1. *Describe and analyse an algorithm which, on input $(M, N, a, b) \in \mathbf{Z}^4$ where $\gcd(M, N) = 1$, outputs an integer x such that $x \equiv a \bmod M$ and $x \equiv b \bmod N$.*

Question 1.2. *Prove that* SquareRootMod *reduces to* Factoring.

Question 1.3. *Let N be an RSA integer. Fix $x \in (\mathbf{Z}/N\mathbf{Z})^\times$. Let y be a uniformly random square root of x^2. What is the probability distribution of $\gcd(x - y, N)$?*

Question 1.4. *Prove that* Factoring *reduces to* SquareRootMod. Hint: beware that if $\mathcal{O}$ is an oracle for SquareRootMod, then $\mathcal{O}(N, x)$ outputs *some* square root of $x \bmod N$, but you cannot assume that it is uniformly distributed.

2. Evaluating an isogeny of degree ℓ^n from its kernel

The goal of this exercise is to prove the following theorem.

Theorem 2.1. *There is an algorithm which given a prime ℓ, an integer n, and a point $P \in E(\mathbf{F}_q)$ of order ℓ^n, can evaluate the quotient isogeny $E \to E/\langle P \rangle$ on any input point at the cost of $O(n \log(n))$ multiplications by ℓ and ℓ-isogeny evaluations.*

Throughout the exercise, we assume that we already know good algorithms for multiplication by ℓ on an elliptic curve (double-and-add) and for computing ℓ-isogenies when ℓ is a small prime (Vélu's formulas). When analysing algorithms, you may simply count the number of calls to these subroutines.

For any elliptic curve E and point $P \in E$, write $\varphi_P : E \to E/\langle P \rangle$ for the separable isogeny of kernel generated by P. For any $P \in E$ of order ℓ^n, let

$$g_n^i(P) = \varphi_{\ell^{n-i}P}(\ell^{n-1-i}P), \text{ and}$$
$$f_n(P) = (g_n^0(P), \ldots, g_n^{n-1}(P)).$$

Question 2.2. *Prove that each $g_n^i(P)$ has order ℓ.*

Question 2.3. *Given $P, Q \in E$ and the output of $f_n(P)$, describe an algorithm to evaluate $\varphi_P(Q)$. How many ℓ-isogenies or multiplications by ℓ does your algorithm require?*

Question 2.4. *Let $0 \leq m \leq n$, and $P \in E$ of order ℓ^n. Prove that*

$$f_n(P) = f_m(\ell^{n-m}P) \text{ cat } f_{n-m}(\varphi_{\ell^{n-m}P}(P)),$$

where cat *is the concatenation of two lists.*

Question 2.5. *Describe and analyse an algorithm to compute f_0 and f_1.*

Question 2.6. *Prove Theorem 2.1.* Hint: *recursively apply Question 2.4 with $m = \lfloor n/2 \rfloor$. Start by analyzing the case where n is a power of two.*

Question 2.7. *Implement your algorithm in Sage. You may use the following function to compute an ℓ-isogeny:*

```
# Compute the image of Q through the separable isogeny of kernel generated by P, when P
↪  has prime order
def velu(P,Q):
        assert(P.order().is_prime())
        E = P.curve()
        return E.isogeny(P)(Q)
```

Question 2.8. *Let $p = 2^{127} - 1$ and $E : y^2 = x^3 - x$ over $\mathbf{F}_{p^2}$. Find two generators P and Q of the group $E(\mathbf{F}_{p^2})$ with the code* `[P,Q] = E.gens()`. Verify that P and Q both have order 2^{127}. Use your implementation to compute $R = \varphi_P(Q)$.