DM 1

Lei Bichang

February 16, 2025

1 Factoring and Modular Square Roots

Question 1.1. *Decription of an algorithm.* First, use the extended Euclidean division to find $u, v \in \mathbb{Z}$ s.t. $uM + vN = 1$. Then $x := vNa + uMb$ does the job:

$$x \equiv vNa \equiv a \bmod M, \quad x \equiv uMb \equiv b \bmod N.$$

Denote by ExtGCD a realization of the extended Euclidean division algorithm.

Algorithm 1: CongruenceModTwoIntegers

Input: $(M, N, a, b) \in \mathbb{Z}^4$, $\gcd(M, N) = 1$
Output: $x \in \mathbb{Z}$, $x \equiv a \bmod M$, $x \equiv b \bmod N$
$(u, v) \leftarrow \mathrm{ExtGCD}(M, N)$, s.t. $uM + vN = 1$;
return $vNa + uMb$

Analysis of the algorithm. This algorithm involves only a fixed number of steps more than ExtGCD. Hence its complexity is $O(\log(M)\log(N))$.

Question 1.2. Assume that we have an oracle $\mathcal{O}_F$ for the problem Factoring. Let $N = pq$ with p, q two distinct odd primes. Since $(\mathbb{Z}/N\mathbb{Z})^\times \simeq \mathbb{F}_p^\times \times \mathbb{F}_q^\times$ via $x \mapsto (x \bmod p, x \bmod q)$, an integer x is a square modulo N if and only if it is a square modulo p and q. In addition, if we have found integers a and b s.t. $x = a^2 \bmod p$ and $x = b^2 \bmod q$, then an integer y satisfying

$$y \equiv a \bmod p, \quad y \equiv b \bmod q$$

would satisfy $x \equiv y^2 \bmod p$ and mod q, and thus $y^2 \equiv x \bmod N$.

For a prime p and an integer x, denote by $\mathrm{Sq}(p, x)$ an algorithm that provides an integer that is a square root of x modulo p. By factoring the polynomial $X^2 - x \in \mathbb{F}_p[X]$, $\mathrm{Sq}(p, x)$ can be realized as a probabilistic polynomial time algorithm. This provides us with the following probabilistic polynomial time algorithm with access to the oracle $\mathcal{O}_F$.

Algorithm 2: SquareRootMod-$\mathcal{O}_F$

Input: N : RSA integer, $x \in \mathbb{Z}$
Output: $y \in \mathbb{Z}$ s.t. $x = y^2 \bmod N$ or False if x is not a square modulo N
$(p, q) \leftarrow \mathcal{O}_F(N)$;
if $x^{\frac{p-1}{2}} = 1$ *and* $x^{\frac{q-1}{2}} = 1$ **then**
  $a \leftarrow \mathrm{Sq}(p, x)$, $b \leftarrow \mathrm{Sq}(q, x)$;
  $y \leftarrow \mathrm{CongruenceModTwoIntegers}(p, q, a, b)$;
  return y
else
  return *False*

Question 1.3. Let $N = pq$ be the prime factorization of N.

Assume first that $2 \nmid N$. Let $y \in (\mathbb{Z}/N\mathbb{Z})^\times$ be a square root of x^2, then $y = \varepsilon x$ with

$$\varepsilon \in \mu_2(\mathbb{Z}/N\mathbb{Z}) = \{a \in \mathbb{Z}/N\mathbb{Z} \mid a^2 = 1\} \overset{f}{\simeq} \{\pm 1\} \times \{\pm 1\}$$

where f is the restriction of the isomorphism $(\mathbb{Z}/N\mathbb{Z})^\times \to \mathbb{F}_p^\times \times \mathbb{F}_q^\times$ obtained from Chinese remainder theorem. So we have

$$x - y \equiv x - \pm x = \begin{cases} 0 \bmod p, & \varepsilon \equiv 1 \bmod p, \\ x \bmod p, & \varepsilon \equiv -1 \bmod p, \end{cases}$$

i.e. $p \mid x - y \iff \varepsilon \equiv 1 \bmod p$. A similar result holds for q. Hence

$$\gcd(x-y, N) = \begin{cases} N, & f(\varepsilon) = \{1, 1\}, \\ p, & f(\varepsilon) = \{1, -1\}, \\ q, & f(\varepsilon) = \{-1, 1\}, \\ 1, & f(\varepsilon) = \{-1, -1\}. \end{cases}$$

If y is uniformly random, $\gcd(x - y, N)$ is also uniformly random in $\{1, p, q, N\}$.

Now assume that $2 \mid N$, say $q = 2$ and $N = 2p$. In this case $\mu_2(\mathbb{Z}/N\mathbb{Z}) \simeq \{\pm 1\}$, so if y is a uniformly random square root of x in $\mathbb{Z}/N\mathbb{Z}$, then $\gcd(x - y, N)$ is uniformly random in

$$\{\gcd(0, N),\ \gcd(2x, N)\} = \{N, 2\}.$$

Question 1.4. Let $\mathcal{O}_S$ be an oracle for SquareRootMod.

Pick $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ uniformly at random. First, consider $y := \mathcal{O}_S(x^2, N)$. If $y \not\equiv \pm x \bmod N$, $\gcd(x-y, N)$ is a non-trivial factor of N by the discussion above. If $y \equiv \pm x \bmod N$, we try another random x and repeat this procedure.

Algorithm 3: Factoring-$\mathcal{O}_S$

Input: N : RSA integer
Output: p, q : primes s.t. $N = pq$
repeat
  $x \leftarrow$ uniformly random in $(\mathbb{Z}/N\mathbb{Z})^\times$;
  $y \leftarrow \mathcal{O}_S(x^2, N)$;
  if $y \not\equiv \pm x$ **then**
    $p \leftarrow \gcd(x - y, N)$;
    return $(p, N/p)$
until;

Analysis of the algorithm. Choosing a uniformly random $x \in (\mathbb{Z}/N\mathbb{Z})^\times$ is equivalent to choosing a uniformly random square $z \in ((\mathbb{Z}/N\mathbb{Z})^\times)^2$ then choosing a uniformly random square root x of z. Therefore, the probability that one loop successfully factors N is the probability of the following event: for a uniformly random square z, a uniformly random square root of z does not equal $\pm\mathcal{O}_S(z, N)$.

Let $\Omega = ((\mathbb{Z}/N\mathbb{Z})^\times, \mathcal{B}, \mathbb{P})$ be the probability space where $\mathcal{B}$ is the powerset of $(\mathbb{Z}/N\mathbb{Z})^\times$ and $\mathbb{P}$ is the uniform distribution. For each $a \in (\mathbb{Z}/N\mathbb{Z})^\times$, fix a square root $\sqrt{a} \in (\mathbb{Z}/N\mathbb{Z})^\times$. Let X (resp. Z, E) be the inclusion maps from $(\mathbb{Z}/N\mathbb{Z})^\times$ (resp. $((\mathbb{Z}/N\mathbb{Z})^\times)^2$, $\mu_2(\mathbb{Z}/N\mathbb{Z})$) to $(\mathbb{Z}/N\mathbb{Z})^\times$. Each of them is a uniformly random variable on its domain[1] with value in Ω, and the chance for one loop to successfully factor N is

$$\mathbb{P}(X \neq \pm\mathcal{O}(X^2)) = \mathbb{P}\left(E\sqrt{Z} \neq \pm\mathcal{O}\left(\left(E\sqrt{Z}\right)^2\right)\right) = \mathbb{P}\left(E\sqrt{Z} \neq \pm\mathcal{O}\left(Z\right)\right) = \mathbb{P}\left(E \neq \frac{\mathcal{O}(Z)}{\sqrt{Z}}\right).$$

[1] As a probability subspace of Ω.

Since the random variables E and $f(Z) := \mathcal{O}(Z)/\sqrt{Z}$ are independent,

$$\begin{aligned}
\mathbb{P}\left(E \neq \pm f(Z)\right) &= 1 - \sum_{\varepsilon \in \mu_2(\mathbb{Z}/N\mathbb{Z})} \mathbb{P}(E = \varepsilon)\left(\mathbb{P}(f(Z) = \varepsilon) + \mathbb{P}(f(Z) = -\varepsilon)\right) \\
&= 1 - \frac{1}{4} \sum_{\varepsilon \in \mu_2(\mathbb{Z}/N\mathbb{Z})} \left(\mathbb{P}(f(Z) = \varepsilon) + \mathbb{P}(f(Z) = -\varepsilon)\right) \\
&= 1 - \frac{1}{4} \cdot 2 = \frac{1}{2}.
\end{aligned}$$

Therefore, each loop has an $\frac{1}{2}$ chance to factor N, ragardless of how $\mathcal{O}_S$ works. The expected number of loops required to factor N is 2. Since every loop costs polynomial time, we get a probabilistic polynomial time algorithm to factor N.

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

Question 2.2. Since ℓ is a prime, it suffices to show that $g_n^i(P) \neq 0$ and $\ell \cdot g_n^i(P) = 0$. Because $\varphi_{\ell^{n-i}P}$ is a group homomorphism from E with kernel generated by $\ell^{n-i}P$, we have

$$\ell g_n^i(P) = \ell \varphi_{\ell^{n-i}P}(\ell^{n-1-i}P) = \varphi_{\ell^{n-i}P}(\ell^{n-i}P) = 0.$$

If $g_n^i(P) = 0$, then $\ell^{n-1-i}P = a\ell^{n-i}P$ for some $a \in \mathbb{Z}$. Because $\ell^{n-i-1}P$ has order $i+1$, it cannot be a multiple of $\ell^{n-i}P$ which has order i. So $g_n^i(P) = \varphi_{l^{n-i}P}(\ell^{n-1-i}P) \neq 0$.

Question 2.3. Let P be of order ℓ^n. Note that for $0 \leq i \leq n-1$, we have $g_n^i(P) \in E/\left\langle \ell^{n-i}P \right\rangle$, and

$$\ker\left(E/\left\langle \ell^{n-i}P \right\rangle \to E/\left\langle \ell^{n-i-1}P \right\rangle\right) = \left\langle \varphi_{\ell^{n-i}P}(\ell^{n-i-1}P) \right\rangle = \left\langle g_n^i(P) \right\rangle.$$

Hence

$$\left(E/\left\langle \ell^{n-i}P \right\rangle\right)/\left\langle g_n^i(P) \right\rangle \simeq E/\left\langle \ell^{n-i-1}P \right\rangle.$$

Thereofre, we can evaluate $\varphi_P(Q)$ along the path

$$\begin{tikzcd}
E \arrow[d, "\text{Vélu}"] & \\
E/\left\langle g_n^0(P) \right\rangle \arrow[r, "\simeq"] & E/\left\langle \ell^{n-1}P \right\rangle \arrow[d, "\text{Vélu}"] \\
E/\left\langle \ell^{n-2}P \right\rangle \arrow[d, dotted] & \left(E/\left\langle \ell^{n-1}P \right\rangle\right)/\left\langle g_n^1(P) \right\rangle \arrow[l, "\simeq"'] \\
\left(E/\left\langle \ell^{n-2}P \right\rangle\right)/\left\langle g_n^{n-2}(P) \right\rangle \arrow[r, "\simeq"] & E/\left\langle \ell^2 P \right\rangle \arrow[d, "\text{Vélu}"] \\
E/\left\langle \ell P \right\rangle \arrow[d, "\text{Vélu}"] & \left(E/\left\langle \ell^{n-2}P \right\rangle\right)/\left\langle g_n^{n-2}(P) \right\rangle \arrow[l, "\simeq"'] \\
\left(E/\ell P\right)/g_n^{n-1}(P) := E/\left\langle P \right\rangle &
\end{tikzcd}$$

where horizontal arrows are isomrophisms and vertical arrows are ℓ-isogenies. The elliptic curves E and $E/\left\langle \ell^j(P) \right\rangle$'s are given by $f_n(P)$, and every other elliptic curve will be computed as the target of an ℓ-isogeny.

Since every curve is stored as a Weierstrass equation, the isomorphisms can be computed in $O(1)$-times $\bar{\mathbb{F}}_q$-operations.

Consequently, this algorithm requires ℓ-isogeny n-times and no multiplication-by-ℓ.

Question 2.4. We have

$$g_m^i(\ell^{n-m}P) = \varphi_{\ell^{m-i}\ell^{n-m}P}(\ell^{m-i-1}\ell^{n-m}P) = \varphi_{\ell^{n-i}P}(\ell^{n-i-1}P) = g_n^i(P).$$

By an argument similar to that in Question 2.3,

$$\left(E/\left\langle \ell^{n-m}P\right\rangle\right)/\left\langle \varphi_{\ell^{n-m}P}(\ell^{n-m-i}P)\right\rangle \simeq E/\left\langle \ell^{n-m-i}P\right\rangle,$$

so

$$\begin{aligned} g_{n-m}^i(\varphi_{\ell^{n-m}P}(P)) &= \varphi_{\ell^{n-m-i}\varphi_{\ell^{n-m}P}(P)}\left(\ell^{n-m-i-1}\varphi_{\ell^{n-m}P}(P)\right) \\ &= \varphi_{\varphi_{\ell^{n-m}P}(\ell^{n-m-i}P)}\left(\varphi_{\ell^{n-m}P}(\ell^{n-m-i-1}P)\right) \\ &= \varphi_{\ell^{n-m-i}P}(\ell^{n-m-i-1}P) = g_n^{m+i}(P). \end{aligned}$$

Hence

$$f_m(\ell^{n-m}P) = \left(g_m^0(\ell^{n-m}P), \ldots, g_m^{m-1}(\ell^{n-m}P)\right) = (g_n^0(P), \ldots, g_n^{m-1}(P)),$$

$$f_{n-m}(\varphi_{\ell^{n-m}P}(P)) = \left(g_{n-m}^0(\varphi_{\ell^{n-m}P}(P)), \ldots, g_{n-m}^{n-m-1}(\varphi_{\ell^{n-m}P}(P))\right) = (g_n^m(P), \ldots, g_n^{n-1}(P))$$

as desired.

Question 2.5. The list $f_0(P)$ is empty for every P of order $\ell^0 = 1$ (namely $P = O$). The complexity is $O(1)$.

The list $f_1(P)$ contains only one element $g_1^0(P) = \varphi_{\ell P}(P) = \varphi_O(P) = P$ is also trivial for P of order ℓ. The complexity is $O(1)$.

Question 2.6. For $P \in E[\ell]\backslash\{O\}$ and $Q \in E$, let $\mathrm{Velu}(-,-)$ be an algorithm such that $\mathrm{Velu}(P,Q) = \varphi_P(Q)$.

For $P \in E[\ell^n] \setminus E[\ell^{n-1}]$ and $Q \in E$, let $\mathrm{Compute}(-,-,-,-)$[2] be the algorithm described in Question 2.3., so that $\mathrm{Compute}(n, P, f_n(P), Q) = \varphi_P(Q)$. In particular, if $n = 1$, then $\mathrm{Compute}(1, P, f_1(P), Q) = \mathrm{Velu}(P, Q)$.

To compute $\varphi_P(Q)$, we use the following algorithm $\mathrm{Aux}(-,-)$ to find $f_n(P)$ so that we can use Compute.

Algorithm 4: Aux

```
Input: n ∈ Z_{≥0}, P ∈ E(F_q) of order ℓ^n
Output: f_n(P)
if n = 0 then
    return ∅
else
    if n = 1 then
        return (P)
    else
        m ← ⌊n/2⌋;
        R ← ℓ^{n−m} P;
        f ← Aux(m, R);
        φ_R(P) ← Compute(m, R, f, P);
        return f ⊔ Aux(n − m, φ_R(P))
```

---

[2] This realization requires redundant arguments to make it more clear for me.

Analysis of the algorithm. Let $T_a(n)$ (resp. $T_c(n)$) be the numbers of multiplications by ℓ and ℓ-isogeny evaluations required for $\mathrm{Aux}(n, -)$ (resp. $\mathrm{Compute}(n, -, f_n(-), -)$ when the list $f_n(-)$ is given). By Question 2.3 and Question 2.5, $T_a(0) = T_a(1) = 0$, $T_c(n) = n$. By definition of the algorithm Aux,

$$\begin{aligned} T_a(n) &= T_a(\lfloor n/2 \rfloor) + T_a(n - \lfloor n/2 \rfloor) + T_c(\lfloor n/2 \rfloor) + (n - \lfloor n/2 \rfloor) \\ &= T_a(\lfloor n/2 \rfloor) + T_a(n - \lfloor n/2 \rfloor) + n \end{aligned}$$

Lemma 2.1. The function T_a on $\mathbb{Z}_{\geq 0}$ is non-decreasing.

Proof. We show that $T_a(n+1) \geq T_a(n)$ by induction. Clearly, that $T_a(2) = 2 > T(1) = T(0)$. Assume that $T_a(m+1) \geq T_a(m)$ for all $m < n$, and set $r := \lfloor n/2 \rfloor$. If n is even, i.e. $n = 2r$,

$$T_a(n+1) - T_a(n) = T_a(r) + T_a(r+1) - 2T_a(r) + 1 = T_a(r+1) - T_a(r) + 1 > 0$$

by induction hypothesis. If n is odd, i.e. $n = 2r + 1$,

$$T_a(n+1) - T_a(n) = 2T_a(r+1) - (T_a(r) + T_a(r+1)) + 1 = T_a(r+1) - T_a(r) + 1 > 0$$

by induction hypothesis. Hence T is non-decreasing. □

Now for $n = 2^r$ with $r \in \mathbb{Z}_{\geq 0}$, we have

$$T_a(2^r) = 2T_a(2^{r-1}) + 2^r,$$

and it is plain to show that

$$T_a(2^r) = 2^r(r + T(0)) = r \cdot 2^r.$$

For general n, let $r := \lfloor \log_2 n \rfloor$, so that $2^r \leq n < 2^{r+1}$, then

$$r \cdot 2^r \leq T(n) \leq (r+1)2^{r+1}$$

by the above lemma, which implies that $T(n) = O(n \log n)$.

Therefore, computing $\varphi_P(Q)$ takes

$$T_a(n) + T_c(n) = T_a(n) + n = O(n \log n)$$

multiplications by ℓ and ℓ-isogeny evaluations.

Question 2.7. Please see the other file. Unlike what we described in Question 2.3, we do not need to compute the isomorphisms

$$\left(E/\left\langle \ell^{n-i}P \right\rangle\right) / \left\langle g_n^i(P) \right\rangle \simeq E/\left\langle \ell^{n-i-1}P \right\rangle$$

in the actual function Compute, probably because the list $f_n(P)$ is computed recursively when running the functions, and every isogeny is computed via Vélu's formula. But I chose to keep the isomorphism-computation part for more generality.

Question 2.8. Please see the other file.