We now consider our first practical application of elliptic curves: factoring integers. Before presenting the elliptic curve method (ECM) for factoring integers, we first present an older algorithm of Pollard that motivates the ECM approach.

12.1 Pollard $p-1$ method

In 1974, Pollard introduced a randomized (Monte Carlo) algorithm for factoring integers [6]. It makes use of a smoothness parameter B.

Algorithm 12.1 (Pollard $p-1$ factorization)**.**
Input: An integer N to be factored and a smoothness bound B.
Output: A proper divisor of N or `failure`.

1. Pick a random integer $a \in [1, N-1]$.
2. If $d = \gcd(a, N)$ is not 1 then return d.
3. Set $b = a$ and for each prime $\ell \le B$:
   a. Set $b = b^{\ell^e} \bmod N$, where $\ell^{e-1} < N \le \ell^e$.
   b. If $d = \gcd(b-1, N)$ is not 1 then return d if $d < N$ or `failure` if $d = N$.
4. Return `failure`

Rather than using a fixed bound B, we could simply let the algorithm keep running through primes ℓ until it either succeeds or fails in step 3b. But in practice one typically uses a very small smoothness bound B and switches to a different algorithm if the $p-1$ method fail. In any case, it is convenient to have B fixed for the purposes of analysis.

Theorem 12.2. *Let p and q be prime divisors of N, and let ℓ_p and ℓ_q be the largest prime divisors of $p-1$ and $q-1$, respectively. If $\ell_p \le B$ and $\ell_p < \ell_q$ then Algorithm 12.1 succeeds with probability at least $1 - \frac{1}{\ell_q}$.*

Proof. If $a \equiv 0 \bmod p$ then the algorithm succeeds in step 2, so we may assume $a \perp p$. When the algorithm reaches $\ell = \ell_p$ in step 3 we have $b = a^m$, where $m = \prod_{\ell \le \ell_p} \ell^e$ is a multiple of $p-1$. By Fermat's little theorem $b = a^m \equiv 1 \bmod p$ and therefore p divides $b-1$. But ℓ_q does not divide m, so with probability at least $1 - \frac{1}{\ell_q}$ we have $b \not\equiv 1 \bmod q$, in which case $1 < \gcd(b-1, N) < N$ in step 3b and the algorithm succeeds. □

For most values of N, the Algorithm 12.1 is guaranteed to succeed if it uses the smoothness bound $B = \sqrt{N}$. But it will still fail if N is a prime power, or if the largest prime dividing $p-1$ is the same for every prime factor p of N.

In the best case, the algorithm can succeed very quickly. As demonstrated in the Sage worksheet `https://hensel.mit.edu:8002/home/pub/8/`, if $N = p_1 p_2$ where p_1 and p_2 are 500-bit primes, if $p_1 - 1$ happens to be very smooth then Alogorithm 12.1 can factor N very quickly, whereas most other methods have no hope of factoring N in a reasonable amount of time. However, in the worst-case the running time is $O(\pi(B)\mathsf{M}(\log N)\log N)$

and we may need to use $B = \sqrt{N}$ The complexity is then $O(\sqrt{N}\mathsf{M}(\log N))$ time, which is the same complexity as trial division.

But rather than focusing on factoring a single number, let us consider a slightly different problem. Suppose we have a large set of composite integers (e.g. a set of RSA moduli), and our goal is to factor any one of them. How long would this take if we simply applied the $p-1$ method to each integer one-by-one, using a suitably chosen smoothness bound B?

For a given value of B, the expected time for the algorithm to achieve a success is

$$\frac{O(\pi(B)\mathsf{M}(\log N)\log N)}{\Pr[\texttt{success}]}. \tag{1}$$

Let p be a prime factor of N. The algorithm is very likely to succeed if $p-1$ is B-smooth, since it is very unlikely that all the other prime factors q of N have $q-1$ with the exact same largest prime factor as $p-1$. Let us heuristically assume that integers of the form $p-1$ are at least as likely to be smooth as a random integer of similar size (arguably they are slightly more likely, since, for example, they must be divisible by 2). By the Canfield-Pomerance-Erdős Theorem, the probability that a random integer less than N is B-smooth is $u^{-u+o(u)}$, where $u = \log N/\log B$. If we ignore the $o(u)$ error term and factors that are polynomial in $\log N$ (which will be bounded by $o(u)$ in any case), we may simplify (1) to

$$N^{1/u}u^u. \tag{2}$$

To pick a value of u that minimizes (2), we take logarithms and set the derivative to 0:

$$\begin{aligned}\frac{d}{du}\left(\frac{1}{u}\log N + u\log u\right) &= 0\\ \frac{-\log N}{u^2} + \log u + 1 &= 0\\ u^2(\log u + 1) &= \log N\end{aligned}$$

If we set $u = \sqrt{2\log N/\log\log N}$ then the difference between the LHS and the RHS is $o(1)$, which is asymptotically negligible. Thus we should use the smoothness bound

$$B = N^{1/u} = \exp\left(\left(1/\sqrt{2}+o(1)\right)\sqrt{\log N\log\log N}\right) = L_N[1/2, 1/\sqrt{2}],$$

where the $o(1)$ term incorporates the $o(u)$ error term and the factors polynomial in $\log N$ that we have ignored. We also have $u^u = \exp(u\log u) = L_N[1/2, 1/\sqrt{2}]$, and the total expected running time is

$$N^{1/u}u^u = L_N[1/2, 1/\sqrt{2}]L_N[1/2, 1/\sqrt{2}] = L_N[1/2, \sqrt{2}].$$

Thus even though the $p-1$ method has an exponential worst-case running time, if we apply it to a sequence of random integers we can achieve a subexponential running time. But this isn't much help if we have a particular integer N that we want to factor.

12.2 The elliptic curve method for factoring integers (ECM)

Using elliptic curves we can effectively achieve the randomized scenario envisioned above even for a single fixed N. The Pollard $p-1$ method achieves a subexponential running time when the group $(\mathbb{Z}/N\mathbb{Z})^*$ varies with each iteration. With the elliptic curve factorization method (ECM), introduced by Hendrik Lenstra [4], a heuristically subexponential running

time is achieved by varying an elliptic curve $E/\mathbb{Q}$ that we reduce modulo a fixed integer N. Now the groups we are effectively working in are of the form $E(\mathbb{F}_p)$, where p is a prime divisor of N. Even though the primes p remain fixed, the number of points on the curve $E(\mathbb{F}_p)$ will vary as we vary the curve E.

The algorithm is essentially the same as Pollard's $p-1$ method. Rather than exponentiating a random element of $(\mathbb{Z}/N\mathbb{Z})^*$ to a large smooth power and hoping that it becomes the identity modulo some prime p dividing N, we instead multiply a random point on an elliptic curve by a large smooth scalar and hope that it becomes the identity modulo some prime p dividing N. The key advantage is that if this doesn't happen we can simply try a different curve, without changing N.

As in Pollard's algorithm, we don't know the value of p, we work modulo N and use gcd's to find a factor of N. If P is a point on $E/\mathbb{Q}$ and $Q = mP = (Q_x : Q_y : Q_z)$ is a multiple of P that reduces to 0 modulo a prime factor of p of N, then $p|\gcd(Q_z, N)$. Notice that even though we are working with points on an elliptic curve over Q, we only care about their reductions modulo primes dividing N, so we can keep the coordinates reduced modulo N throughout the algorithm.

In order to get a proper divisor of N we also need $\gcd(Q_z, N) \neq N$. This is very likely to be the case, so long as P is not a torsion point of $E(\mathbb{Q})$; if P is a torsion point it will have the same order modulo every prime divisor of N and we will always have $\gcd(Q_z, N) = N$ whenever the gcd is non-trivial. Given an elliptic curve $E/\mathbb{Q}$, it is generally hard to find non-torsion points in $E(\mathbb{Q})$, in fact there will often not be any.[1] Instead we pick integers $x_0, y_0, a \in [1, N-1]$ and let $b = y_0^2 - x_0^3 - ax_0$. This guarantees that $P = (x_0, y_0)$ is a rational point on the elliptic curve $E/\mathbb{Q}$ defined by $y^2 = x^3 + ax + b$, and we claim that with very high probability it is a point of infinite order. To see why, let us consider a prime $p \approx N$ (not necessarily a divisor of N). The reduction of the point P modulo p is then nearly uniformly distributed over the group of rational points on the reduction of E modulo p, of which there are $\Omega(p)$. But only $O(1)$ of these points can be the reduction of a torsion point (by Mazur's Theorem, the number of torsion points on an elliptic curve over $\mathbb{Q}$ is at most 16). Thus for large N the probability that P has infinite order is close to 1.

We now give the algorithm.

Algorithm 12.3 (ECM)**.**
Input: An integer N to be factored, a smoothness bound B, and a prime bound M.
Output: A proper divisor of N or `failure`.

1. Pick random integers $a, x_0, y_0 \in [0, N-1]$ and set $b = y_0^2 - x_0^3 - ax_0$.
2. If $d = \gcd(4a^3 + 27b^2, N)$ is not 1 then return d if $d < N$ or `failure` if $d = N$.
3. Let $Q = P = (x_0 : y_0 : 1)$.
4. For all primes $\ell < B$:
   a. Set $Q = \ell^e Q \bmod N$, where $\ell^{e-1} \leq (\sqrt{M}+1)^2 < \ell^e$.
   b. If $d = \gcd(Q_z, N)$ is not 1 then return d if $d < N$ or `failure` if $d = N$.
5. Return `failure`.

[1] There are standard parameterizations that are guaranteed to produce a curve $E/\mathbb{Q}$ with a known point $P \in E(\mathbb{Q})$ of infinite order; see [1], for example. Here we show how to generate random E and P.

The scalar multiplication in step 4a is performed using projective coordinates, and while it is defined in terms of the group operation in $E(\mathbb{Q})$, we only keep track of the coordinates of Q modulo N; the projective coordinates are integers and there are no inversions involved, so all of the arithmetic is valid modulo N.

Theorem 12.4. *Assume $4a^3+27b^2$ is not divisible by N, and let P_1 and P_2 be the reductions of P modulo distinct primes p_1 and p_2 dividing N, with $p_1 \le M$. Suppose $|P_1|$ is ℓ_1-smooth and $|P_2|$ is not, for some prime $\ell_1 \le B$. Then Algorithm 12.3 succeeds.*

Proof. When the algorithm reaches step 2b with $\ell = \ell_1$ we must have $Q = mP$, where $m = \prod_{\ell \le \ell_1} \ell^e$ is a multiple of $|P_1|$, since $|P_1|$ is ℓ_1-smooth and $|P_1| \le (\sqrt{p_1}+1)^2 \le (\sqrt{M}+1)^2$. So $Q \equiv 0 \bmod p_1$, but $Q \not\equiv 0 \bmod p_2$, since $|P_2|$ is not ℓ_1-smooth. Therefore Q_z is divisible by p_1 but not p_2 and a proper factor $d = \gcd(Q_z, N)$ of N will be found in step 4b. □

If the algorithm fails, we can simply try again. Heuristically, so long as N is not a perfect power and has a prime factor $p \le M$, we will eventually succeed. The case where N is a prime power can be efficiently handled using the algorithm of Problem 1 from Problem Set 3. Provided N is not a prime power and has a prime factor $p < M$, Algorithm 12.3 is very likely to succeed whenever it picks a triple (x_0, y_0, a) that yields an elliptic curve whose reduction modulo p has B-smooth order. So the number of times we expect to run the algorithm before we succeed depends on the probability that $\#E(\mathbb{F}_p)$ is B-smooth.

The integer $\#E(\mathbb{F}_p)$ must lie in the Hasse interval $[p - 2\sqrt{p} + 1, p + 2\sqrt{p} + 1]$, which is unfortunately too narrow for us to apply any theorems on the density of B-smooth integers (we can't even prove that this interval contains a prime, and the density of primes is far greater). So to analyze the complexity of Algorithm 12.3 (and to optimize the choice of B), we resort to the heuristic assumption that, at least when $\#E(\mathbb{F}_p)$ lies in the narrower interval $[p+1-\sqrt{p}+1, p+1+\sqrt{p}]$, the probability the $\#E(\mathbb{F}_p)$ is B-smooth is comparable to the probability that a random integer in the interval $[p, 2p]$ is B-smooth.[2]

One can prove that the probability that $\#E(\mathbb{F}_p)$ lies in $[p-\sqrt{p}+1, p+\sqrt{p}+1]$ is at least $1/2$ (this is implied, asymptotically, by the Sato-Tate theorem), and further that probability that $\#E(\mathbb{F}_p)$ takes on any particular value in this interval is $\Omega(1/(\sqrt{p}\log p))$. These facts are both proved in Lenstra's paper [4], and we will be able to prove them ourselves once we have covered the theory of complex multiplication. This means that we can make our heuristic assumption independent of any facts about elliptic curves, we simply need to assume that a random integer in the interval $[p + 1 - \sqrt{p}, p + 1 + \sqrt{p}]$ has roughly the same probability of being B-smooth as a random integer in the interval $[p, 2p]$.

Under our heuristic assumption, the analysis of the algorithm follows the analysis of the Pollard $p-1$ method. This algorithm takes $O(\pi(B)(\log M)\mathsf{M}(\log N))$ time per elliptic curve, and if N has a prime factor $p \le M$, it will need to try an average of $O(u^u)$ curves before it finds a factor. As in §12.1, this implies that the optimal value of B is $L_M[1/2, 1/\sqrt{2}]$, and with this value of B the expected time to factor N is $L_M[1/2, \sqrt{2}]\mathsf{M}(\log N)$. In general, we may not know a bound M on the smallest prime factor p of N *a priori*, but if we simply start with a small choice of M and periodically double it, we can achieve a running time of

$$L_p[1/2, \sqrt{2}]\mathsf{M}(\log N).$$

The key point is that this running time depends almost entirely on p rather than N, a property that distinguishes ECM from all other factorization algorithms with heuristically

[2] Asymptotically, this is the same as the probability that a random integer in $[1, p]$ is B-smooth.

subexponential running times. There are factorization algorithms such as the quadratic sieve and the number field sieve that are heuristically faster when all of the prime factors of N are large, but in practice one first uses ECM to look for any relatively small prime factors before resorting to these heavyweight algorithms.

The fact that the complexity of ECM depends primarily on the size of the smallest prime divisor of N makes it a very good algorithm for smoothness testing, something that is needed as a sub-routine of the quadratic sieve and number field sieve, and by many other algorithms in computational number theory. Testing whether a given integer N is $L_N[1/2, c]$-smooth using ECM takes just

$$\begin{aligned} L_{L_N[1/2,c]}\left[1/2, \sqrt{2}\right] &\approx \exp\left(\sqrt{2\log(\exp(c\sqrt{\log N \log\log N})\log\log(\exp(c\sqrt{\log N \log\log N})}\right) \\ &= \exp\left(\sqrt{2c\sqrt{\log N \log\log N}(1/2\log\log N + o(\log\log N))}\right) \\ &= \exp\left(\sqrt{c}(\log N)^{1/4}(\log\log N)^{3/4}\right) \\ &= L_N\left[1/4 + o(1), \sqrt{c}\right] \end{aligned}$$

expected time, which is faster than any other method known.[3]

12.3 Efficient implementation

Algorithm 12.3 spends essentially all of its time performing elliptic curve scalar multiplications modulo N, so it is worth choosing the elliptic curve representation and the coordinate system to optimize this operation. Edwards curves, which we saw in Lecture 2, are an excellent choice; see [3] for a detailed discussion of how to efficiently implement ECM using Edwards curves. Another popular choice is Montgomery curves [5]. These were originally introduced specifically for the purpose of optimizing the elliptic curve factorization method but are now used in many other applications of elliptic curves, including primality proving and cryptography.

12.4 Montgomery Curves

A *Montgomery curve* is an elliptic curve defined by an equation of the form

$$By^2 = x^3 + Ax^2 + x, \tag{3}$$

where $B \neq 0$ and $A \neq \pm 2$. To convert this to Weierstrass form, let $u = Bx$ and $w = B^2y$. Substituting $x = u/B$ and $y = w/B^2$ in (3) and multiplying by B^3 yields

$$w^2 = u^3 + ABu^2 + B^2u,$$

which is in the form of a general Weierstrass equation. To obtain a short Weierstrass equation, we assume our base field has characteristic different from 3 and complete the

[3] There are faster algorithms for finding smooth numbers within a sufficiently large set of integers; see [2].

cube by letting $v = u + \frac{AB}{3}$. We then obtain

$$\begin{aligned}
w^2 &= u^3 + ABu^2 + B^2u \\
w^2 &= \left(v - \frac{AB}{3}\right)^3 + AB\left(v - \frac{AB}{3}\right)^2 + B^2\left(v - \frac{AB}{3}\right) \\
w^2 &= v^3 - ABv^2 + \frac{A^2B^2}{3}v - \frac{A^3B^3}{27} + ABv^2 - \frac{2A^2B^2}{3}v + \frac{A^3B^3}{9} + B^2v - \frac{AB^3}{3} \\
w^2 &= v^3 + \left(B^2 - \frac{A^2B^2}{3}\right)v + \left(\frac{2A^3B^3}{27} - \frac{AB^3}{3}\right).
\end{aligned}$$

In order to check that (3) actually defines an elliptic curve, we should verify that it is nonsingular. We could do these using the coefficients of the curve in short Weierstrass form, but it is easier to do this directly. We need to determine whether there are any points $(x : y : z)$ on the projective curve $By^2z = x^3 + Ax^2z + xz^2$ at which all three partial derivatives vanish. For any such point we must have

$$\begin{aligned}
\frac{\partial}{\partial x}: &\quad 3x^2 + 2Axz + z^2 = 0 \\
\frac{\partial}{\partial y}: &\quad 2Byz = 0 \\
\frac{\partial}{\partial z}: &\quad By^2 = Ax^2 + 2xz = 0.
\end{aligned}$$

We assume we are working in a field of characteristic not equal to 2 or 3. Suppose that $y \neq 0$. Then the equation for $\frac{\partial}{\partial y}$ gives $z = 0$, and from $\frac{\partial}{\partial x}$, we get $x = 0$. But this is a contradiction, since the equation for $\frac{\partial}{\partial z}$ is not satisfied. On the other hand, if $y = 0$, then $z = -\frac{A}{2}x \neq 0$. We have $3x^2 - A^2x^2 + \frac{A^2}{4}x^2 = 0$, and therefore $3 - \frac{3}{4}A^2 = 0$, since $x \neq 0$. Thus $A^2 = 4$, but we require $A \neq \pm 2$ in (3), so this cannot be the case.

12.5 Montgomery curve group law

The transformation of a Montgomery curve to Weierstrass form is a linear transformation that preserves the symmetry about the y-axis, so the geometric view of the group law remains the same: three points on a line sum to zero, which is is the point at infinity. To add points P_1 and P_2 we construct the line $\overline{P_1P_2}$ (using a tangent when $P_1 = P_2$), find the third intersection point with the curve, and then reflect over the y-axis to obtain $P_3 = P_1 + P_2$. In this section we compute explicit algebraic formulas for this operation, just as we did for curves in Weierstrass form earlier in the course.

The cases involving inverses and the point at infinity are easy (we have $P - P = 0$ and $P + 0 = 0 + P = P$), so let $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ be two (possibly equal but not opposite) affine points on the curve whose sum $P_3 = (x_3, y_3)$ we wish to compute. We first compute the slope m of the line $\overline{P_1P_2}$.

$$m = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{if } P_1 \neq P_2, \\ \dfrac{3x_1^2 + 2Ax_1 + 1}{2By_1} & \text{if } P_1 = P_2. \end{cases} \tag{4}$$

Now we want to intersect the line $y - y_1 = m(x - x_1)$ with the curve equation (3). Substituting $m(x - x_1) + y_1$ in for y, we get

$$B\left(m(x - x_1) + y_1\right)^2 = x^3 + Ax^2 + x. \tag{5}$$

We know x_1, x_2, and x_3 are the three roots of this cubic equation, since P_1, P_2, and $-P_3$ all lie on the curve and the line $\overline{P_1P_2}$. Thus the coefficient of x^2 in (5) must be equal to $x_1 + x_2 + x_3$. We get a Bm^2x^2 term on the left side of (5) and an Ax^2 term on the right, so we have $x_1 + x_2 + x_3 = Bm^2 - A$. Solving for x_3 and using the equation for $\overline{P_1P_2}$ to compute $-y_3$, we obtain

$$\begin{aligned} x_3 &= Bm^2 - (A + x_1 + x_2) \\ y_3 &= m(x_1 - x_3) - y_1. \end{aligned} \tag{6}$$

These formulas closely resemble the formulas for a curve in short Weierstrass form, but with an extra B and A in the equation for x_3. However, they have the key property that they allow us to completely eliminate the y-coordinate from consideration. This is useful because the y-coordinate is not needed in many applications; we not need to know the y-coordinate of a point P in order to determine whether $mP = 0$ for a given integer m. This makes the y-coordinate superfluous in applications such as ECM and ECPP.

Let us consider the doubling case first. Plugging in the expression for m given by (4) in the case $P_1 = P_2 = (x_1, y_1)$ into (6) and remembering the curve equation $By^2 = x^3 + Ax^2 + x$, we compute

$$\begin{aligned} x_3 &= B\frac{(3x_1^2 + 2Ax_1 + 1)^2}{4B^2y_1^2} - (A + 2x_1) \\ &= \frac{(3x_1^2 + 2Ax_1 + 1)^2 - 4(A + 2x_1)(x_1^3 + Ax_1^2 + x_1)}{4(x_1^3 + Ax_1^2 + x_1)} \\ &= \frac{(x_1^2 - 1)^2}{4x_1(x_1^2 + Ax_1 + 1)}, \end{aligned}$$

thus we can derive x_3 from x_1 without needing to know y_1. Switching to projective coordinates, we have

$$\begin{aligned} \frac{x_3}{z_3} &= \frac{\left(\left(\frac{x_1}{z_1}\right)^2 - 1\right)^2}{4\frac{x_1}{z_1}\left(\left(\frac{x_1}{z_1}\right)^2 + A\frac{x_1}{z_1} + 1\right)} \\ &= \frac{(x_1^2 - z_1^2)^2}{4x_1z_1(x_1^2 + Ax_1z_1 + z_1^2)} \\ &= \frac{(x_1^2 - z_1^2)^2}{4x_1z_1\big((x_1 - z_1)^2 + (A + 2)x_1z_1\big)}. \end{aligned}$$

Thus we may write

$$\begin{aligned} x_3 &= (x_1 + z_1)^2(x_1 - z_1)^2 \\ 4x_1z_1 &= (x_1 + z_1)^2 - (x_1 - z_1)^2 \\ z_3 &= 4x_1z_1\big((x_1 - z_1)^2 + C(4x_1z_1)\big). \end{aligned} \tag{7}$$

where $C = (A + 2)/4$. Notice that these formulas do not involve y_1 and they only require 5 multiplications: 3 to compute x_3, none to compute $4x_1z_1$, and 2 more to compute z_3. One of these is a multiplication by the constant C, which may take negligible time if we can arrange for C to be small.

Now let us do the same thing for addition:

$$\begin{aligned}
x_3 &= B\frac{(y_1-y_2)^2}{(x_1-x_2)^2} - (A+x_1+x_2) \\
x_3(x_1-x_2)^2 &= B(y_1-y_2)^2 - (A+x_1+x_2)(x_1-x_2)^2 \\
&= By_1^2 + By_2^2 - 2By_1y_2 - (A+x_1+x_2)(x_1-x_2)^2 \\
&= -2By_1y_2 + 2x_1x_2(A+x_1+x_2) + x_1 + x_2 \\
&= -2By_1Y_2 + x_2(x_1^2+Ax_1+1) + x_1(x_2^2+Ax_2+1) \\
&= -2By_1y_2 + \frac{x_2}{x_1}By_1^2 + \frac{x_1}{x_2}By_2^2 \\
&= B\frac{(x_2y_1-x_1y_2)^2}{x_1x_2} \qquad (8)
\end{aligned}$$

This gives us an equation for x_3 in $P_3 = P_1 + P_2$, but it still involves the y-coordinates of P_1 and P_2. To address this, let us also compute the x-coordinate x_4 of $P_4 = P_1 - P_2$. The hard work is already done, we just need to negate y_2 in the equation for x_3. Thus

$$x_4(x_1-x_2)^2 = B\frac{(x_2y_1+x_1y_2)^2}{x_1x_2}. \qquad (9)$$

Multiplying equations (8) and (9) yields

$$\begin{aligned}
x_3x_4(x_1-x_2)^4 &= \frac{B^2(x_2^2y_1^2 - x_1^2y_2^2)^2}{x_1^2x_2^2} \\
&= \frac{(x_2^2By_1^2 - x_1^2By_2^2)^2}{x_1^2x_2^2)} \\
&= \frac{\left(x_2^2(x_1^3+Ax_1^2+x_1) - x_1^2(x_2^3+Ax_2^2+x_2)\right)^2}{x_1^2x_2^2} \\
&= \left(x_2(x_1^2+Ax_1+1) - x_1(x_2^2+Ax_2+1)\right)^2 \\
&= (x_2x_1^2 - x_1x_2^2 + x_2 - x_1)^2 \\
&= ((x_1-x_2)(x_1x_2-1))^2 .
\end{aligned}$$

Canceling a factor of $(x_1-x_2)^2$ from both sides gives

$$x_3x_4(x_1-x_2)^2 = (x_1x_2-1)^2, \qquad (10)$$

which does not involve y_1 or y_2 (but does require us to know x_4).

We now switch to projective coordinates:

$$\begin{aligned}
\frac{x_3}{z_3}\cdot\frac{x_4}{z_4}\left(\frac{x_1}{z_1}-\frac{x_2}{z_2}\right)^2 &= \left(\frac{x_1x_2}{z_1z_2}-1\right)^2 \\
\frac{x_3}{z_3} &= \frac{z_4}{x_4}\cdot\frac{(x_1x_2-z_1z_2)^2}{(x_1z_2-x_2z_1)^2},
\end{aligned}$$

which yields

$$\begin{aligned}
x_3 &= z_4\left[(x_1-z_1)(x_2+z_2)+(x_1+z_1)(x_2-z_2)\right]^2 \qquad (11) \\
z_3 &= x_4\left[(x_1-z_1)(x_2+z_2)-(x_1+z_1)(x_2-z_2)\right]^2
\end{aligned}$$

These formulas require just 6 multiplications, but they assume that we already know the x-coordinate x_4/z_4 of $P_1 - P_2$. This appears rather limiting, but if we structure the double-and-add algorithm for scalar multiplication appropriately, we can use the formulas in (7) and (11) to efficiently compute the x-coordinate of the scalar multiple mP using what is known as a *Montgomery ladder*. In the algorithm below we assume points are represented simply as projective pairs $(x : z)$ that omit the y-coordinate.

Algorithm 12.5 (Montgomery Ladder)**.**
Input: A point $P = (x_1 : z_1)$ on a Montgomery curve and a positive integer m.
Output: The point $mP = (x_m : z_m)$.

1. Let $m = \sum_{i=0}^{k} m_i 2^i$ be the binary representation of m.
2. Set $Q[0] = P$ and compute $Q[1] = 2P$ (note that $P = Q[1] - Q[0]$).
3. For $i = k - 1$ down to 0:
   a. $Q[1 - m_i] \leftarrow Q[1] + Q[0]$ (Using $P = Q[1] - Q[0]$)
   b. $Q[m_i] \leftarrow 2Q[0]$
4. Return $Q[0]$.

The Montgomery ladder is the usual double-and-add algorithm, augmented to ensure that $Q[1] - Q[0] = P$ is invariant throughout. A nice feature of the algorithm is that every iteration of the loop is essentially the same: a Montgomery addition followed by a Montgomery doubling. This makes the algorithm resistant to side-channel attacks based on processor timing. If we assume that the input point P is in affine form $(x_1 : 1)$, then $z_1 = z_4 = 1$ in the addition formulas in (11), which saves one multiplication. This yields a total cost of $\big(10 + o(1)\big) \log_2 m$ field multiplications for Algorithm 12.5, or only $\big(9 + o(1)\big) \log_2 m$ if the constant C is small enough to make the multiplications by C negligible.

An implementation of Algorithms 12.3 and 12.5 can be found in the sage worksheet

https://hensel.mit.edu:8002/home/pub/11/.

12.6 Torsion on a Montgomery Curve

Every Montgomery point has $(0, 0)$ as a rational point of order 2 (as with curves in short Weierstrass form, the points of order 2 are precisely those with y-coordinate 0). This tells us that not every elliptic curve can be put in Montgomery form, since not every elliptic curve has a rational point of order 2. In fact, more is true.

Theorem 12.6. *The Montgomery curve E/k defined by $By^2 = x^3 + Ax^2 + x$ has either 3 rational points of order 2 or a rational point of order 4 (or both).*

Proof. The cubic $x^3 + Ax^2 + x$ has either one or three rational roots, and these roots are distinct, since the curve is nonsingular. If it has three roots, then there are three rational points of the form $(x, 0)$, all of which have order 2.

If it has only one root, then $x^2 + Ax + 1$ has no roots, so $A^2 - 4 = (A + 2)(A - 2)$ is not a quadratic residue. Therefore one of $A + 2$ and $A - 2$ is a quadratic residue (and the other is not), so either $\frac{A+2}{B}$ or $\frac{A-2}{B}$ is a quadratic residue. We will use this fact to find a point of order 4 that doubles to the 2-torsion point $(0, 0)$, which is the unique point on the curve whose x-coordinate is 0.

To get $x_3 = 0$ in the doubling formulas (7), we must have $x_1 = \pm z_1$, equivalently, $x_1/z_1 = \pm 1$. Plugging this into the curve equation, we seek a solution to either $By^2 = A+2$ or $By^2 = A-2$. But we have already shown that either $\frac{A+2}{B}$ or $\frac{A-2}{B}$ is a quadratic residue, so one of these equations has a solution and there is a rational point of order 4. □

Thus, like Edwards curves, the torsion subgroup of a Montgomery curve always has order divisible by 4. For the purposes of the ECM algorithm this is actually a feature, since it slightly increases the likelihood that the group order will be smooth. In fact, most implementations use specific parameterizations to generate curves $E/\mathbb{Q}$ that are guaranteed to have even larger torsion subgroups, typically isomorphic to either $\mathbb{Z}/12\mathbb{Z}$ or $\mathbb{Z}/2\mathbb{Z}\oplus\mathbb{Z}/8\mathbb{Z}$; see [1, 3, 5] for examples.

12.7 Second stage

Finally, we should mention that almost all practical implementations of ECM utilize some form of *second stage* that is applied at the end of Algorithm 12.3. This topic is explored in the problem set.

References

[1] A. O. L. Atkin and F. Morain, *Finding suitable curves for the elliptic curve method of factorization*, Mathematics of Computation **60** (1992), 399–405.

[2] D. J. Bernstein, *How to find smooth parts of integers*, preprint http://cr.yp.to/factorization/smoothparts-20040510.pdf, 2004.

[3] D. J. Bernstein, P. Birkner, T. Lange, and C. Peters, *ECM using Edwards curves*, Mathematics of Computation **82** (2013), 1139–1179.

[4] H. Lenstra, *Factoring integers with elliptic curves*, Annals of Mathematics 126 (1987), 649–673

[5] P. L. Montgomery, *Speeding the Pollard and elliptic curve methods of factorization*, Mathematics of Computation **48** (1987), 243–264.

[6] J. M. Pollard, *Theorems of Factorization and Primality Testing*, Proceedings of the Cambridge Philosophical Society 76 (1974): 521–528

[7] P. Zimmermann and B. Dodson, *20 years of ECM*, Algorithmic Number Theory 7th International Symposium (ANTS VII), LNCS 4076 (2006), 525–542.