# 10 Index calculus, smooth numbers, and factoring integers

Having explored generic algorithms for the discrete logarithm problem in some detail, we now consider a non-generic algorithm based on *index calculus*.[1] This algorithm depends critically on the distribution of *smooth numbers* (integers with small prime factors), which naturally leads to a discussion of two algorithms for factoring integers that also depend on smooth numbers: the Pollard $p-1$ method and the elliptic curve method (ECM).

## 10.1 Index calculus

Index calculus is a method for computing discrete logarithms in the multiplicative group of a finite field. This might not seem directly relevant to the elliptic curve discrete logarithm problem, but as we shall see when we discuss pairing-based cryptography, these two problems are not completely unrelated. Moreover, index calculus based methods can be applied to the discrete logarithm problem on elliptic curves over non-prime finite fields, as well as abelian varieties of higher dimension (even over prime fields); see [8, 9, 10].[2]

We will restrict our attention to the simplest case, a finite field of prime order $\mathbb{F}_p \simeq \mathbb{Z}/p\mathbb{Z}$, and let us fix the set of integers in $[0, N]$ with $N = p-1$ as a set of coset representatives for $\mathbb{Z}/p\mathbb{Z}$. Index calculus exploits the fact that we "lift" elements of $\mathbb{Z}/p\mathbb{Z}$ to their representatives in $[0, N] \cap \mathbb{Z}$.

$$\mathbb{Z} \overset{\longrightarrow}{\longleftarrow} \mathbb{Z}/p\mathbb{Z} \simeq \mathbb{F}_p$$

The map $\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$ is the canonical quotient map given by reduction modulo $p$, and it is a ring homomorphism. The "lifting" map from $\mathbb{Z}/p\mathbb{Z}$ to $\mathbb{Z}$ is a section of the quotient map, which is an injective map of sets but is not a ring homomorphism.[3] Nevertheless, if we lift elements from $\mathbb{Z}/p\mathbb{Z}$ to $\mathbb{Z}$, perform a sequence of ring operations in $\mathbb{Z}$, and then reduce modulo $p$, we will get the same result as if we had performed the entire sequence of ring operations in $\mathbb{Z}/p\mathbb{Z} \simeq \mathbb{F}_p$. A key feature of working in $\mathbb{Z}$ is that we can uniquely factor integers in $[1, N]$ into prime powers, something that makes no sense in the field $\mathbb{Z}/p\mathbb{Z}$ where every nonzero element is a unit and there are no nontrivial prime ideals.

Let us fix a generator $\alpha$ for $(\mathbb{Z}/p\mathbb{Z})^\times$, and let $\beta \in \langle\alpha\rangle$ be the element whose discrete logarithm we wish to compute. For any integer $e$, we may consider the prime factorization of the integer $\alpha^e\beta^{-1} \in [1, N] \subseteq \mathbb{Z}$; here we are implicitly lifting $\alpha^e\beta^{-1} \in \mathbb{Z}/p\mathbb{Z}$ to its unique coset representative in $[1, N]$, as we will continue to do without further comment. When $e = \log_\alpha \beta$ this prime factorization will be trivial, but in general we will have

$$\prod p_i^{e_i} = \alpha^e\beta^{-1},$$

where the $p_i$ vary over primes and the exponents $e_i$ are nonnegative integers. Multiplying both sides by $\beta$ and taking discrete logarithms with respect to $\alpha$ yields

$$\sum e_i \log_\alpha p_i + \log_\alpha \beta = e,$$

[1] If $\alpha$ is a generator for $\mathbb{F}_p^\times$ then the discrete logarithm of $\beta \in \mathbb{F}_p^\times$ with respect to $\alpha$ is also called the *index* of $\beta$ (with respect to $\alpha$), whence the term *index calculus*.

[2] The two are related: if $E$ is an elliptic curve over a finite field $\mathbb{F}_{q^n}$ for some prime-power $q$, there is an associated abelian variety of dimension $n$ over $\mathbb{F}_q$ known as the *Weil restriction* of $E$.

[3] Indeed, there are no homomorphisms from rings of positive characteristic to rings of characteristic zero (note that the zero ring has positive characteristic).

which determines $\log_\alpha \beta$ as a linear expression in the discrete logarithms $\log_\alpha p_i$, where $\log_\alpha p_i$ denotes the discrete logarithm of the image of $p_i$ under the quotient map $\mathbb{Z} \to \mathbb{Z}/p\mathbb{Z}$. This doesn't immediately help us, since we don't know the values of $\log_\alpha p_i$. However, if we repeat this procedure using many different values of $e$, we may obtain a system of linear equations that we can try to solve for $\log_\alpha \beta$.

In order to make this feasible, we need to restrict the primes $p_i$ to lie in a reasonably small set. We thus fix a *smoothness bound*, say $B$, and define the *factor base*

$$P_B = \{p : p \le B \text{ is prime}\} = \{p_1, p_2, \dots, p_b\},$$

where $b = \pi(B)$ is the number of primes up to $B$ (of which there are approximately $B/\log B$). Not all choices of $e$ will yield an integer $\alpha^e \beta^{-1} \in [1, N] \subseteq \mathbb{Z}$ that we can factor over our factor base $P_B$, in fact most will not. But some choices will work, and for those that do we obtain a linear equation of the form

$$e_1 \log_\alpha p_1 + e_2 \log_\alpha p_2 + \cdots + e_b \log_\alpha p_b + \log_\alpha \beta = e,$$

in which at most $\lfloor \lg N \rfloor$ of the $e_i$ are nonzero. We may not know any of the discrete logarithms that appear in this relation, but we can view

$$e_1 x_1 + e_2 x_2 + \cdots + e_b x_b + x_{b+1} = e$$

as a linear equation in $b+1$ variables $x_1, x_2, \dots, x_{b+1}$ over the ring $\mathbb{Z}/N\mathbb{Z}$. This equation has a solution, namely, $x_i = \log_\alpha p_i$, for $1 \le i \le b$, and $x_{b+1} = \log_\alpha \beta$. If we collect $b+1$ such equations by choosing random values of $e$ and discarding those for which $\alpha^e \beta^{-1}$ is not $B$-smooth, the resulting linear system may determine a unique value $x_{b+1}$, the discrete logarithm we wish to compute.

This system will typically be under-determined; indeed, many of the variables $x_i$ may not appear in any of our relations. But it is quite likely that the value of $x_{b+1}$, which is present in every equation, will be uniquely determined. We will not attempt to prove this (to give a rigorous proof one really needs more than $b+1$ equations, say, $b \log b$), but it is empirically true.[4] This suggests the following algorithm to compute $\log_\alpha \beta$.

**Algorithm 10.1** (Index calculus)**.** Given $\beta \in \langle \alpha \rangle = (\mathbb{Z}/p\mathbb{Z})^\times$, compute $\log_\alpha \beta$ as follows:

1. Pick a smoothness bound $B$, compute the factor base $P_B \coloneqq \{p_1, \dots, p_b\}$ with $b \coloneqq \pi(B)$, and let $N \coloneqq p - 1$.
2. Generate $b+1$ random relations $R_i = (e_{i,1}, e_{i,2}, \dots, e_{i,b}, 1, e_i)$ by picking $e \in [1, N]$ at random and attempting to factor $\alpha^e \beta^{-1} \in [1, N]$ over the factor base $P_B$. Each successful factorization yields a relation $R_i$ with $e_i = e$ and $\alpha^{e_i} \beta^{-1} = \prod p_j^{e_{i,j}}$.
3. Attempt to solve the system defined by the relations $R_1, \dots, R_{b+1}$ for $x_{b+1} \in \mathbb{Z}/N\mathbb{Z}$ using linear algebra (row reduce the corresponding matrix).
4. If $x_{b+1} = \log_\alpha \beta$ is uniquely determined, return this value, otherwise go to step 2.

It remains to determine the choice of $B$ in step 1, but let us first make the following remarks.

**Remark 10.2.** It is not actually necessary to start over from scratch when $x_{b+1}$ is not uniquely determined, typically adding just a few more relations will be enough.

[4] When considering potential attacks on a cryptographic system, one should always be willing to make any reasonable heuristic assumption that helps the attacker.

**Remark 10.3.** As noted above, the relations $R_1, \ldots, R_{b+1}$ will be very sparse (at most $\lfloor \lg N \rfloor + 1$ nonzero coefficients in each), which can speed up the linear algebra step significantly.

**Remark 10.4.** While solving the system $R_1, \ldots, R_{b+1}$ we are likely to encounter non-invertible elements of $\mathbb{Z}/N\mathbb{Z}$ (for example, 2 is never invertible, since $N = p - 1$ is even). Whenever this happens we can use a GCD computation to obtain a non-trivial factorization $N = N_1 N_2$ with $N_1$ and $N_2$ relatively prime. We then proceed to work in $\mathbb{Z}/N_1\mathbb{Z} \times \mathbb{Z}/N_2\mathbb{Z}$, using the CRT to recover the value of $x_{b+1}$ in $\mathbb{Z}/N\mathbb{Z}$ (recurse as necessary).

**Remark 10.5.** Solving the system of relations will determine not only $x_{b+1} = \log_\alpha \beta$, but also many $x_i = \log_\alpha p_i$ for $p_i \in P_B$, which do not depend on $\beta$. If we are computing discrete logarithms for many different $\beta$ with respect to the same base $\alpha$, after the first computation the number of relations we need is just one more than the number of $x_i = \log_\alpha p_i$ that have yet to be determined. If we are computing discrete logarithms for $\Omega(b)$ values of $\beta$, we expect to compute just $O(1)$ relations per discrete logarithm, on average.

An integer whose prime factors are all bounded by $B$ is said to be *$B$-smooth*. A large value of $B$ will make it more likely that $\alpha^e \beta^{-1}$ is $B$-smooth, but it also makes it more difficult to determine whether this is in fact the case, since we need to determine all he prime factors of $\alpha^e \beta^{-1}$ up to $B$. We want to balance the cost of smoothness testing against the number of smoothness tests we expect to need in order to get $b + 1$ relations (note that $b$ depends on $B$). Let us suppose for the moment that the cost of the linear algebra step is negligible by comparison (which turns out to be the case, at least in terms of time complexity). If we choose $e \in [1, N]$ uniformly at random then $\alpha^e$, and therefore $\alpha^e \beta^{-1}$, will be uniformly distributed over $(\mathbb{Z}/p\mathbb{Z})^\times$, uniquely represented by the set of integers in $[1, N]$. To determine the optimal value of $B$, we need to know the probability that a random integer in $[1, N]$ is $B$-smooth.

## 10.2 The Canfield-Erdös-Pomerance Theorem

For positive real numbers $x$ and $y$, let $\psi(x, y)$ count the $y$-smooth integers in $[1, x]$. The probability that a random integer $m \in [1, x]$ is $y$-smooth is then approximately $\frac{1}{x}\psi(x, y)$. We want our smoothness bound $y$ to vary as a function of $x$, so it is standard to define

$$u := \frac{\log x}{\log y}$$

and replace $y$ by $x^{1/u}$.

**Theorem 10.6** (Canfield-Erdős-Pomerance)**.** *The asymptotic bound*

$$\frac{1}{x}\psi(x, x^{1/u}) = u^{-u+o(u)}$$

*holds uniformly as $u, x \to \infty$, provided that $u < (1 - \epsilon) \log x / \log \log x$ for some $\epsilon > 0$.*

For a proof on this result along with many other interesting facts about smooth numbers, we recommend the survey article by Granville [13].

## 10.3 Optimizing the smoothness bound

Let us assume that generating relations in step 2 dominates the overall complexity of Algorithm 10.1, and for the moment suppose that we simply use trial-division to attempt to factor $\alpha^e\beta^{-1}$ over $P_B$ (we will see a more efficient method for smoothness testing shortly). The expected running time of Algorithm 10.1 is then approximately

$$(b+1)\cdot u^u \cdot b \cdot \mathsf{M}(\log N), \tag{1}$$

where $u = \log N/\log B$. The four factors in (1) are:

- $b+1$: the number of relations $R_i$ that we need;
- $u^u$: the expected number of random exponents $e$ we need to try in order to obtain a $B$-smooth integer $m \coloneqq \alpha^e\beta^{-1} \in [1, N]$;
- $b$: the number of trial divisions to test whether $m$ is $B$-smooth (and factor it if it is);
- $\mathsf{M}(\log N)$: the time for each trial division.

We have $b = \pi(B) \sim B/\log B$, and if we ignore logarithmic factors we can replace both $b+1$ and $b$ by $B$ and drop the $\mathsf{M}(\log N)$ factor. We wish to choose $u$ to minimize the quantity

$$B^2u^u = N^{2/u}u^u, \tag{2}$$

where we have used $B^u = N$ to eliminate $B$. Taking logarithms, it suffices to minimize

$$f(u) = \log(N^{2/u}u^u) = \frac{2}{u}\log N + u\log u,$$

so we want to consider solutions to

$$f'(u) = -\frac{2}{u^2}\log N + \log u + 1 = 0.$$

Ignoring the asymptotically negligible constant 1, we would like to pick $u$ so that

$$u^2\log u \approx 2\log N.$$

For

$$u = 2\sqrt{\log N/\log\log N}, \tag{3}$$

we have

$$u^2\log u = \frac{4\log N}{\log\log N}\cdot\left(\log 2 + \frac{1}{2}(\log\log N - \log\log\log N)\right) = 2\log N + o(\log N),$$

as desired. The choice of $u$ in (3) implies that we should use the smoothness bound

$$\begin{aligned} B = N^{1/u} &= \exp\left(\frac{1}{u}\log N\right)\\ &= \exp\left(\frac{1}{2}\sqrt{\log N\log\log N}\right)\\ &= L_N[1/2, 1/2]. \end{aligned}$$

Here we have used the asymptotic notation

$$L_N[\alpha, c] \coloneqq \exp\big((c + o(1))(\log N)^{\alpha}(\log\log N)^{1-\alpha}\big),$$

which is commonly used to denote complexity bounds of this form. Note that

$$L_N[0, c] = \exp((c + o(1)\log\log N) = (\log N)^{c+o(1)}$$

is polynomial in $\log N$, whereas

$$L_N[1, c] = \exp((c + o(1))\log N) = N^{c+o(1)}$$

is exponential in $\log N$. For $0 < \alpha < 1$ the bound $L_N[\alpha, c]$ is *subexponential* (in $\log N$).

We also have $u^u = \exp(u \log u) = L_N[1/2, 1]$, thus the total expected running time is

$$B^2 u^u = L_N[1/2, 1/2]^2 \cdot L_N[1/2, 1] = L_N[1/2, 2].$$

The cost of the linear algebra step is certainly no worse than $\widetilde{O}(b^3)$, which is $\widetilde{O}(B^3)$, In our subexponential notation this is $L_N[1/2, 3/2]$, which is dominated by the bound above, so our assumption that the cost of generating relations dominates the running time is justified. In fact, if we take advantage of the sparseness of the system noted in Remark 10.3, the cost of the linear algebra step can be bounded by $\widetilde{O}(b^2)$. However, in large computations the linear algebra step is often a limiting factor in practice because it is memory intensive and not as easy to parallelize as relation finding.

**Remark 10.7.** As noted earlier, if we are computing many (say at least $L_N[1/2, \sqrt{2}/2]$) discrete logarithms with respect to the same base $\alpha$, we just need $O(1)$ relations per $\beta$, on average. In this case we should choose $B = N^{1/u}$ to minimize $Bu^u$ rather than $B^2u^u$. This yields an average expected running time of $L_N[1/2, \sqrt{2}]$ per discrete logarithm.

A simple version of Algorithm 10.1 using trial-division for smoothness testing is implemented in this Sage notebook.

## 10.4 Improvements

Using the elliptic curve factorization method (ECM) described in the next section, the cost of testing and factoring $B$-smooth integers can be made subexponential in $B$ and polynomial in $\log N$. This effectively changes $B^2u^u$ in (2) to $Bu^u$, and the optimal smoothness bound becomes $B = L_N[1/2, 1/\sqrt{2}]$, yielding a heuristic expected running time of

$$L_N[1/2, \sqrt{2}].$$

There is a batch smoothness testing algorithm due to Bernstein [3] that for a sufficiently large set of integers yields an average time per integer that is actually polynomial in $\log N$, but this does not change the complexity in a way that is visible in our $L_N[\alpha, c]$ notation.

Using more advanced techniques, analogous to those used in the *number field sieve* for factoring integers, one can achieve a heuristic expected running time of the form

$$L_N[1/3, c]$$

for computing discrete logarithms in $\mathbb{F}_p^\times$ (again using an index calculus approach); see [12].

In finite fields of small characteristic $\mathbb{F}_{p^n} \simeq \mathbb{F}_p[x]/(f(x))$, one uses the *function field sieve*, where now the factor base consists of low degree polynomials in $\mathbb{F}_p[x]$ that represent elements of $\mathbb{F}_{p^n}$ when reduced modulo $f(x)$. This also yields an $L_N[1/3, c]$ bound (with a smaller value of $c$). Under heuristic assumptions, such a bound holds for all finite fields [16].

But this is far from the end of the story. In 2013 Antoine Joux announced an index calculus approach for finite fields of the form $\mathbb{F}_{q^k}$ with $q \approx k$ that heuristically achieves an $L_N[1/4 + o(1), c]$ time complexity [14]. Shortly thereafter a recursive variant of Joux's approach was used to obtain a heuristically quasi-polynomial-time complexity of $k^{O(\log k)}$, which in terms of $N = q^k$ is bounded by $L_N[\epsilon, c]$ for every $\epsilon, c > 0$. At first glance the assumption $q \approx k$ might seem restrictive, but even for finite fields of the form $\mathbb{F}_{2^k}$ with $k$ prime it suffices to compute discrete logarithms in the extension field $\mathbb{F}_{2^{kr}}$ with $r = \lceil \lg k \rceil$, which for $q = 2^r \approx k$ has the desired form $\mathbb{F}_{q^k}$. Even though we are now working in a larger field, the $k^{O(\log k)}$ bound is still quasi-polynomial in the input size $k$, and as a function of $N = 2^k$ it is dominated by $L_N[\epsilon, c]$ for all $\epsilon, c > 0$, hence quasi-polynomial-time.

As of March 2021 the record for computing discrete logarithms in finite fields was set in the field $\mathbb{F}_{2^{30750}}$, using about 2900 core-years in 2019 [11]. The record for prime degree finite fields was set in 2014 in the field $\mathbb{F}_{2^{1279}}$, using less than 4 core years [15] (this record could surely be improved), and the record for "safe" prime fields $\mathbb{F}_p$ (where $(p-1)/2$ is prime), was set in 2019 for a 795-bit prime $p$ using about 3100 core years [6].

The recent dramatic improvements in computing discrete logarithms in finite fields of small characteristic has effectively eliminated interest in pairing-based elliptic curve cryptography over such fields. As discussed in Lecture 1, in pairing-based cryptography one needs to consider the difficulty of the discrete logarithm problem both in the group of rational points on an elliptic curve over a finite field $\mathbb{F}_q$ and in the multiplicative group of a low degree extension of $\mathbb{F}_q$. None of these results have had any impact on the prospects of pairing-based cryptography over prime fields.[5]

## 10.5 The Pollard $p-1$ method

In 1974, Pollard introduced a Monte Carlo algorithm for factoring integers [19] that works astonishingly well when the integer $p-1$ is extremely smooth (but in the worst case is no better than trial division). The algorithm takes as input an integer $N$ to be factored and a smoothness bound $B$.

**Algorithm 10.8** (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 $\ell^{e-1} < N \le \ell^e$. If $b = 1$ then return `failure`.
   b. If $d = \gcd(b-1, N)$ is not 1 then return $d$.
4. Return `failure`

[5] Quantum computers are a potential threat, but this is a separate issue; the attacks based on Joux's breakthrough all use classical models of computation.

Rather than using a fixed bound $B$, we could simply let the algorithm keep running through primes $\ell$ 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 fails. In any case, it is convenient to have $B$ fixed for the purposes of analysis.

**Example 10.9.** Let $N = 899$ and suppose we pick $a = 2$ in step 1. Then $d = 1$ in step 2, and the table below illustrates the situation at the end of each iteration of step 3.

| $\ell$ | $e$ | $b$ | $d$ |
|---|---|---|---|
| 2 | 10 | 605 | 1 |
| 3 | 7 | 690 | 1 |
| 5 | 5 | 683 | 31 |

The algorithm finds the factor 31 of $N = 29 \cdot 31$ when $\ell = 5$ because $\#(\mathbb{Z}/31)^\times = 30 = 2 \cdot 3 \cdot 5$ is 5-smooth but $\#(\mathbb{Z}/29)^\times = 28 = 2^2 \cdot 7$ is not: if we put $m = 2^{10} \cdot 3^7 \cdot 5^5$ then $m$ is divisible by $\#(\mathbb{Z}/31\mathbb{Z})^\times$ but not by $\#(\mathbb{Z}/29\mathbb{Z})^\times$, and it follows that we always have $a^m \equiv 1 \bmod 31$, but for most choices of $a$ we will have $a^m \not\equiv 1 \bmod 29$, leading to $d = \gcd(a^m - 1, 29 \cdot 31) = 31$.

If we had instead used $N = 31 \cdot 41$ we would have found $d = N$ when $\ell = 5$ and failed because $\#(\mathbb{Z}/41\mathbb{Z})^\times = 40 = 2^3 \cdot 5$ has the same largest prime factor as $\#(\mathbb{Z}/31\mathbb{Z})^\times$.

**Theorem 10.10.** *Let $p$ and $q$ be prime divisors of $N$, and let $\ell_p$ and $\ell_q$ be the largest prime divisors of $p-1$ and $q-1$, respectively. If $\ell_p \le B$ and $\ell_p < \ell_q$ then Algorithm 10.8 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 $\ell_q$ 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 almost all values of $N$, Algorithm 10.8 will succeed with very high probability given the smoothness bound $B = \sqrt{N}$. But if $N$ is a prime power, or if the largest prime dividing $p-1$ is the same for every prime factor $p$ of $N$ it will still fail, no matter what value of $a$ is chosen. In the best case, the algorithm can succeed very quickly. As demonstrated in this Sage notebook, if $N = p_1 p_2$ where $p_1$ and $p_2$ are 512-bit primes, if $p_1 - 1$ happens to be very smooth then Algorithm 10.8 can factor $N$ within a few seconds; no other algorithm currently known can factor this integer $N$ 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 with $B = \sqrt{N}$ the complexity is $O(\sqrt{N}\,\mathsf{M}(\log N))$, the same as trial division (and as noted above, success is not guaranteed).

But rather than focusing on factoring a single integer $N$, let us consider a slightly different problem. Suppose we have a large set of composite integers (for example, a list of RSA moduli[6]), 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?

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

$$\frac{O(\pi(B)\,\mathsf{M}(\log N)\log N)}{\Pr[\texttt{success}]}. \tag{4}$$

[6] In fact, many RSA key generation algorithms incorporate specific measures to prevent the type of attack we consider here. In any case, current RSA keys are necessarily large enough (2048 bits) to be quite safe from the $L_N[1/2, \sqrt{2}]$ algorithm considered here.

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 exactly the 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.

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 (4) to

$$N^{1/u}u^u. \tag{5}$$

This is minimized (up to asymptotically negligible factors) for $u = \sqrt{2\log N/\log\log N}$, thus we should use the smoothness bound

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

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 therefore

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

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 achieve a (heuristically) subexponential running time. But this isn't much help if there is a particular integer $N$ that we want to factor.

## 10.6 The elliptic curve method for factoring integers (ECM)

Using elliptic curves we can effectively achieve the randomized scenario envisioned above while keeping $N$ fixed. The Pollard $p-1$ algorithm works in the group $(\mathbb{Z}/N\mathbb{Z})^\times$, but we can also think of it as performing simultaneous computations in the groups $(\mathbb{Z}/p\mathbb{Z})^\times$ for primes $p|N$; it succeeds when one of these groups has smooth order. If we instead take an elliptic curve $E/\mathbb{Q}$ defined by an integral equation $y^2 = x^3 + Ax + B$ that we can reduce modulo $N$, we have an opportunity to factor $N$ whenever $E(\mathbb{F}_p)$ has smooth order, for some prime $p|N$. The key difference is that we can vary the curve $E$ while keeping $N$ fixed; we get a new group $E(\mathbb{F}_p)$ each time we change $E$. This is the basis of the elliptic curve method (ECM), introduced by Hendrik Lenstra [17] in the mid 1980s.

The algorithm is essentially the same as Pollard's $p-1$ method. Rather than exponentiating a random element of $(\mathbb{Z}/N\mathbb{Z})^\times$ 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 $N$. If this doesn't happen we switch to a different curve and try again.

As in Pollard's $p-1$ algorithm, we don't know the primes $p$ dividing $N$ *a priori*, so we work modulo $N$ and use GCD's to find a factor of $N$. If $P$ is a point on $E/\mathbb{Q}$ and $mP = (Q_x : Q_y : Q_z)$ is a multiple of $P$ that reduces to 0 modulo a prime $p$ dividing $N$, then $p$ divides $\gcd(Q_z, N)$. Notice that even though we are working with points on an elliptic curve over $\mathbb{Q}$, 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 may not be any.[7] 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 $y^2 = x^3 + ax + b$. The probability that $P$ is a torsion point is negligible.[8] We now give the algorithm, which takes not only an integer $N$ and a smoothness bound $B$, but also a bound $M$ on the largest prime factor of $N$ that we seek to find (as discussed below, this is useful for smoothness testing).

**Algorithm 10.11** (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 $\ell^{e-1} \le (\sqrt{M}+1)^2 < \ell^e$.
   b. If $d = \gcd(Q_z, N)$ is not 1 then return $d$ if $d < N$ or `failure` if $d = N$.
5. Return `failure`.

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 can be performed in $\mathbb{Z}/N\mathbb{Z}$.

**Theorem 10.12.** *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 $\ell_1$-smooth and $|P_2|$ is not, for some prime $\ell_1 \le B$. Then Algorithm 10.11 succeeds.*

*Proof.* When the algorithm reaches step 4b 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 $\ell_1$-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 $\ell_1$-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, provided $N$ is not a perfect power and has a prime factor $p \le M$, we will eventually succeed. Factoring perfect powers can be efficiently handled by the algorithm developed in Problem 1 of Problem Set 3. Provided $N$ is not a prime power and has a prime factor $p < M$, Algorithm 10.11 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 $[p+1-2\sqrt{p}, p+1+2\sqrt{p}]$, which is unfortunately too narrow for us to apply any theorems on the density of $B$-smooth integers

[7] 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 just generate random $E$ and $P$ at random.

[8] This follows (for example) from the Lutz–Nagell theorem [20, Theorem 8.7], which implies that if $y_0$ is nonzero then $y_0^2$ must divide $4a^3 + 27b^2 = 4a^3 + 27(x_0^3 + ax_0)^2$, which is extremely unlikely.

(we cannot even prove that this interval contains any primes, and smooth numbers are much rarer than primes). So to analyze the complexity of Algorithm 10.11 (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 $[p+1-\sqrt{p}, p+1+\sqrt{p}]$, 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.[9]

One can prove that the probability that $\#E(\mathbb{F}_p)$ lies in $[p+1-\sqrt{p}, p+1+\sqrt{p}]$ is at least $1/2$ (this is implied, asymptotically, by the Sato–Tate theorem), and further that the 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 [17], 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 §10.5, 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

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

where $p$ is the smallest prime factor of $N$.

A crucial 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 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 $N$ also makes it a very good algorithm for smoothness testing. Testing whether a given integer $N$ is $L_N[1/2, c]$-smooth using ECM takes just

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

expected time, which is faster than any other method known.[10]

## 10.7 Efficient implementation

Algorithm 10.11 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

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

[10] As noted earlier, for batch smoothness testing, Bernstein's algorithm [3] is faster.

system to optimize this operation. Edwards curves, which we saw in Lecture 2, are an excellent choice; see [4] for a detailed discussion of how to efficiently implement ECM using Edwards curves. Another popular choice is Montgomery curves [18]; as explained in [5], there is a close relationship between Montgomery curves and Edwards curves. 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.

## 10.8 Montgomery Curves

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

$$By^2 = x^3 + Ax^2 + x, \tag{6}$$

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 (6) 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 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 (6) 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

$$\frac{\partial}{\partial x}\colon 3x^2 + 2Axz + z^2 = 0, \qquad \frac{\partial}{\partial y}\colon 2Byz = 0, \qquad \frac{\partial}{\partial z}\colon By^2 - (Ax^2 + 2xz) = 0.$$

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 (6), so this cannot be the case.

## 10.9 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 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 $\overline{P_1P_2}$.

$$m = \begin{cases} \dfrac{y_1 - y_2}{x_1 - x_2} & \text{if } P_1 \neq P_2, \\ \dfrac{3x_1^2 + 2Ax_1 + 1}{2By_1} & \text{if } P_1 = P_2. \end{cases} \tag{7}$$

Now we want to intersect the line $y - y_1 = m(x - x_1)$ with the curve equation (6). Substituting $m(x - x_1) + y_1$ in for $y$, we get

$$B\left(m(x - x_1) + y_1\right)^2 = x^3 + Ax^2 + x. \tag{8}$$

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 (8) must be equal to $x_1 + x_2 + x_3$. We get a $Bm^2x^2$ term on the left side of (8) 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

$$\begin{aligned} x_3 &= Bm^2 - (A + x_1 + x_2) \\ y_3 &= m(x_1 - x_3) - y_1. \end{aligned} \tag{9}$$

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 do 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 (7) in the case $P_1 = P_2 = (x_1, y_1)$ into (9) and remembering the curve equation $By^2 = x^3 + Ax^2 + x$,

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

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

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

Thus we may write

$$\begin{aligned} x_3 &= (x_1+z_1)^2(x_1-z_1)^2 \\ 4x_1z_1 &= (x_1+z_1)^2-(x_1-z_1)^2 \\ z_3 &= 4x_1z_1\big((x_1-z_1)^2+C(4x_1z_1)\big). \end{aligned} \tag{10}$$

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:

$$\begin{aligned} x_3 &= B\frac{(y_1-y_2)^2}{(x_1-x_2)^2}-(A+x_1+x_2) \\ x_3(x_1-x_2)^2 &= B(y_1-y_2)^2-(A+x_1+x_2)(x_1-x_2)^2 \\ &= By_1^2+By_2^2-2By_1y_2-(A+x_1+x_2)(x_1-x_2)^2 \\ &= -2By_1y_2+2x_1x_2(A+x_1+x_2)+x_1+x_2 \\ &= -2By_1y_2+x_2(x_1^2+Ax_1+1)+x_1(x_2^2+Ax_2+1) \\ &= -2By_1y_2+\frac{x_2}{x_1}By_1^2+\frac{x_1}{x_2}By_2^2 \\ &= B\frac{(x_2y_1-x_1y_2)^2}{x_1x_2} \end{aligned} \tag{11}$$

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

$$x_4(x_1-x_2)^2=B\frac{(x_2y_1+x_1y_2)^2}{x_1x_2}. \tag{12}$$

Multiplying equations (11) and (12) yields

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

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

$$x_3x_4(x_1-x_2)^2=(x_1x_2-1)^2, \tag{13}$$

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

We now switch to projective coordinates:

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

which yields

$$x_3 = z_4\left[(x_1 - z_1)(x_2 + z_2) + (x_1 + z_1)(x_2 - z_2)\right]^2 \tag{14}$$
$$z_3 = x_4\left[(x_1 - z_1)(x_2 + z_2) - (x_1 + z_1)(x_2 - z_2)\right]^2$$

These formulas require just 6 multiplications, but they assume that we already know the $x$-coordinate $x_4/z_4$ of $P_1 - P_2$. But if we structure the double-and-add algorithm for scalar multiplication appropriately, we can use the formulas in (10) and (14) to efficiently compute the $x$-coordinate of the scalar multiple $mP$ using what is known as a *Montgomery ladder*. We assume points are represented simply as projective pairs $(x : z)$ that omit the $y$-coordinate.

**Algorithm 10.13** (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. 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 (14), which saves one multiplication. This yields a total cost of $\big(10+o(1)\big)\log_2 m$ field multiplications for Algorithm 10.13, or only $\big(9+o(1)\big)\log_2 m$ if the constant $C$ is small enough to make the multiplications by $C$ negligible. This is faster than using Edwards' curves (at least in a side-channel resistant configuration where one is not using optimized doubling formulas).

An implementation of Algorithms 10.11 and 10.13 can be found in this Sage notebook.

## 10.10 Torsion on a Montgomery Curve

Every Montgomery curve 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 10.14.** *The Montgomery curve $E/k$ defined by $By^2 = x^3 + Ax^2 + x$ has either three rational points of order 2 or a rational point of order 4 (possibly 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 (10), 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, 4, 18] for examples (the $\mathbb{Z}/12\mathbb{Z}$ case is illustrated in the example implementation).

The converse of Theorem 10.14 does not hold; there are elliptic curves with three rational points of order 2 that cannot be put in Montgomery form. However, every elliptic curve with a rational point of order 4 can be put in Montgomery form.

**Theorem 10.15.** *Let $E\colon y^2 = x^3 + ax + b$ be an elliptic curve over a field $k$. Suppose $E(k)$ contains a point $P$ of order 4, and let $2P = (x_0, 0)$. Then $3x_0^2 + a$ is a square in $k$ and $E$ can be put in Montgomery form $E'\colon By^2 = x^3 + Ax^2 + x$ by setting $B = 1/\sqrt{3x_0^2 + a}$ and $A = 3x_0B$; the map $(x, y) \mapsto (B(x - x_0), By)$ defines an isomorphism from $E$ to $E'$.*

*Proof.* Let $P = (u, v)$. From the elliptic curve doubling formula, we have

$$\begin{aligned} x_0 &= \left(\frac{3u^2 + a}{2v}\right)^2 - 2u \\ &= \frac{(9u^4 + 6au^2 + a^2) - 8u(u^3 + au + b)}{4(u^3 + au + b)} \\ &= \frac{u^4 - 2au^2 - 8bu + a^2}{4(u^3 + au + b)}. \end{aligned}$$

Therefore $u$ satisfies

$$u^4 - 4x_0u^3 - 2au^2 - (4ax_0 + 8b)u - 4bx_0 + a^2 = 0.$$

We have $0^2 = x_0^3 + ax_0 + b$, so we can replace $b$ by $-x_0^3 - ax_0$, yielding

$$u^4 - 4x_0u^3 - 2au^2 + (8x_0^3 + 4ax_0)u + 4x_0^4 + 4ax_0^2 + a^2 = 0.$$

The LHS is a perfect square. If we put $u = z + x_0$ we can write this as

$$(z^2 - (3x_0^2 + a))^2 = 0.$$

Now $z = u - x_0 \in k$, so $z^2 - (3x_0^2 + a)$ must have a root in $k$. Thus $3x_0^2 + a$ is a square, as claimed, and it is nonzero because $x_0$ is not a repeated root of $x_0^3 + ax_0 + b$. Now let $B = 1/\sqrt{3x_0^2 + a}$ and $A = 3x_0B$ be as in the theorem and let $E'\colon By^2 = x^3 + Ax^2 + x$.

To check that $(x, y) \mapsto (B(x - x_0), By)$ defines an isomorphism from $E \to E'$, we plug $(B(x - x_0), By)$ into the equation for $E'$ and note that

$$\begin{aligned} B(By)^2 &= (B(x - x_0))^3 + A(B(x - x_0))^2 + B(x - x_0) \\ B^2y^2 &= B^2(x^3 - 3x_0x^2 + 3x_0^2x - x_0^3) + 3x_0B^2(x^2 - 2x_0x + x_0^2) + x - x_0 \\ y^2 &= x^3 - 3x_0^2x + 2x_0^3 + (x - x_0)(3x_0^2 + a) \\ y^2 &= x^3 + ax - x_0^3 - ax_0 \\ y^2 &= x^3 + ax + b. \end{aligned}$$

This also shows that $E'$ is not singular, since $E$ is not (so we must have $A^2 \neq 4$). □

## References


[1] A.O.L. Atkin and François Morain, *Finding suitable curves for the elliptic curve method of factorization*, Mathematics of Computation **60** (1993), 399–405.

[2] Razvan Barbulescu, Pierrick Gaudry, Antoine Joux, Emmanuel Thomé, *A heuristic quasi-polynomial algorithm for discrete logarithm in finite fields of small characteristic*, in Advances in Cryptology — EUROCRYPT 2014, LNCS **8441** (2014), Springer, 1–16.

[3] Daniel J. Bernstein, *How to find smooth parts of integers*, unpublished preprint, 2004.

[4] Daniel J. Bernstein, Peter Birkner, Tanja Lange, and C. Peters, *ECM using Edwards curves*, Mathematics of Computation **82** (2013), 1139–1179.

[5] Daniel J. Bernstein and Tanja Lange, *Montgomery curves and the Montgomery ladder*, Cryptology ePrint Archive, Report 2017/293, 2017.

[6] Fabrice Boudot, Pierrick Gaudry, Aurore Guillevic, Nadia Heninger, Emmanuel Thomé, and Paul Zimmermann, *795-bit factoring and discrete logarithms*, NMBRTHRY listserv posting, December 2, 2019.

[7] E. R. Canfield, Paul Erdős, and Carl Pomerance, *On a problem of Oppenheim concerning "factorisatio numerorum"*, Journal of Number Theory **17** (1983), 1–28.

[8] Andreas Enge, *Discrete logarithms in curves over finite fields*, Finite fields and applications, Contemporary Mathematics **461**, AMS, 2008, 119–139 (arXiv:0712.3916).

[9] Andreas Enge and Pierrick Gaudry, *A general framework for subexponential discrete logarithm algorithms*, Acta Arithmetica **102** (2002), 83–103.

[10] Pierrick Gaudry, *Index calculus for abelian varieties of small dimension and the elliptic curve discrete logarithm problem*, J. Symbolic Computation **44** (2009), 1690–1702.

[11] Robert Granger, Thorsten Kleinjung, Arjen Lenstra, Benjamin Wesolowski, Jens Zumbrägel, *Discrete logarithms in GF*($2^{30750}$), NMBRTHRY listserv posting, July 10, 2019.

[12] Daniel M. Gordon, *Discrete Logarithms in $GF(p)$ using the number field sieve*, SIAM J. Discrete Math **6** (1993), 124–138.

[13] Andrew Granville, *Smooth numbers, computational number theory and beyond*, in Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography (MSRI Workshop), Mathematical Sciences Research Institute Publications **44**, 2008, 267–324.

[14] Antoine Joux, *A new index calculus algorithm with complexity $L(1/4 + o(1))$ in very small characteristic*, in Selected Areas in Cryptography — SAC 2013, LNCS **8282** (2014), Springer, 355–379.

[15] Thorsten Kleinjung, *Discrete logarithms in $GF(2^{1279})$*, NMBRTHRY listserv posting, October 17, 2014.

[16] Antoine Joux, Reynald Lercier, Nigel Smart, and Frederik Vercauteren, *The number field sieve in the medium prime case*, Advances in Cryptology — CRYPTO 2006, LNCS **4117** (2006), Springer, 326–344.

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

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

[19] J. M. Pollard, *Theorems of factorization and primality testing*, Proceedings of the Cambridge Philosophical Society **76** (1974): 521–528

[20] Lawrence C. Washington, *Elliptic curves: Number theory and cryptography*, second edition, Chapman and Hall/CRC, 2008.

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