Factoring integers using elliptic curves

- classical problem
- useful - if you can do it well enough, you break RSA,
- the methods we will describe aren't actually good against reasonably large keysize RSA, but are the fastest way to find 'moderate size factors'.

What does 'fast' mean?

- # of logical operations, read, writes.

- in practice, we count additions & multiplications of integers (maybe mod some n).

Example: Computing powers mod n.

Given $a, k, n \in \mathbb{Z}_{\geq 1}$, & want to compute $a^k \bmod n$.

(i.e. find $b \in [0, n-1] \cap \mathbb{Z}$ s.t. $a^k - b \in n\mathbb{Z}$).

1st idea: $a_1 := a \bmod n$
$a_2 := a \cdot a_1 \bmod n$
$\vdots$
$a_k = a \cdot a_{k-1} \bmod n$.

} $2k$ 'operations', i.e. k additions, k mult mod n's

Better: write k in binary, i.e. write $k = \sum_{i \geq 0} \varepsilon_i 2^i$, $\varepsilon_i \in \{0, 1\}$

(Clearly $\varepsilon_i = 0$ if $i > \log_2 k$).

For each $0 \leq i \leq \log_2 k$, compute $a^{2^i} \bmod n$

- needs i operations.

So computing all these costs $< (\log_2 k)^2$ (can do better...)

Then computing k takes ~~6~~ can be done in another $\log_2 k$ multiplications & $\log_2 k$ additions, so our algorithm runs in time

$$\simeq (\log_2 k)^2 + 2\log_2 k.$$

Much faster! [Ex: do it in $4\log_2 k$ operations].

Eg Computing $\gcd(a,b)$ for $a,b \in \mathbb{Z}$. By looking at prime factors is very slow. But using Euclid's algorithm, can be done in

$$2\log_2 \max(2a, 2b)$$ 'operations', where each 'operation' is a 'division with remainder'.

Pf Homework? ☺

Pollard's factoring method

Now we have the general idea, we turn to factoring. First we describe an approach not using ECs, then will use ECs to refine.

- First naive algorithm for factoring will find a factor in $\leq \sqrt{n}$ steps (Algo: (does 2|n? no. Does 3|n? no, ...). Way too slow.

- Factoring fast $\cong$ finding a factor fast.

Why? - small factors v. easy to find.
~~- for interesting prob~~
- for RSA, $n = pq$ with p, q prime
- more generally, first get rid of small factors, then once you have one big factor, time to complete is negligible compared to whole algorithm.

Recall:

Thm [Little Fermat]: Let p prime & $a \in \mathbb{Z}$ s.t. $(a,p)=1$. Then

$$a^{p-1} \equiv 1 \bmod p.$$

Pf: $\left(\mathbb{Z}/p\mathbb{Z}\right)^*$ is a gp of order $p-1$, & exponent | order. □

Lemma: if $(a,p)=1$ then $p \mid \gcd(a^{p-1}-1, n)$.

Pollard's algorithm: (Π_k = (Product of first k primes)^k. All calcs should be done mod n in practice!)

Step 1: Set $k := 2$.

Step 2: Define $\ell = \Pi_k$.

Step 3: Pick $1 < a < n$ 'at random' (no repeats)

Step 4: compute $\gcd(a,n)$. If $\neq 1$, done (found a factor).

Step 5: Compute $D := \gcd(a^{\ell}-1, n)$. If $1 < D < n$, done.
- If $D = n$, go to step 2 (rare)
- If $D = 1$, $k \mathrel{+}= 1$, go to step 4.

Algorithms need pfs almost as much as theorems!

Prop: steps. Pf: ex. (□) More interesting:

Prop: Say n has a prime factor p s.t. $p-1 \mid (\Pi_k)^k$.

Then the algorithm probably stops after $\sim k$ steps/iterations.

Pf: - Mostly, $n \nmid a^k - 1$, so we can discount the 'go to step 2' loops.

- By FLT, if $p-1 \mid (\Pi_k)^k = \ell$ then $a^{\ell} \equiv 1 \bmod p$,

ie. $p \mid a^{e}-1$, so $p \mid D$. This happens by the k^{th} iteration. □

- This isn't guaranteed to work, because in theory you could waste a quite a bit of time on the 'go to step 2' loops. But in practice this does not happen.

- In reality, biggest problem with algorithm is that p may be very large, & $p-1$ may have a very large factor.

How large is 'very large'? (In 2015).

Before looking at EC methods, lets digress to talk about the kind of numbers you need to factor to break RSA:

- an 'RSA-number' is the product of two similar sized primes - & these are the hardest to factor.

- the largest RSA number to be factored so far (as part of RSA factoring challenge) is $\sim 10^{232} \simeq 2^{768}$. '768 bit key'

- In practice, most modern encryption uses 1024 or larger bit keys, too expensive to break for now.

- some still use 512 bit keys - safe in 1995, can now be factored in a few weeks on common hardware.

- note: technically to break RSA, don't have to factor n, but it suffices & actual problem doesn't seem very much easier.

Lenstra's elliptic curve factoring algorithm

We saw that Pollard works well if n has a factor p s.t. $p-1$ is product of 'small' primes to 'small' powers.
Conjecturally, Lenstra's doesn't suffer from this.

Pollard based on FLT, aka $\left(\mathbb{Z}/p\mathbb{Z}\right)^*$ is a gp of order $p-1$.

Lenstra replaces $\left(\mathbb{Z}/p\mathbb{Z}\right)^* = \mathbb{F}_m(\mathbb{F}_p)$ by $E(\mathbb{F}_p)$ for E an elliptic curve.

The algorithm is set up so that you win if n has a prime factor p s.t. for some E with small coeffs,

$E(\mathbb{F}_p)$ is a product of small primes to small powers.

With Pollard, if for every $p \mid n$ we have that (say) $p-1$ is a product of two similarly sized primes, the algorithm is v. slow (≈ doesn't work)

Whereas with Lenstra, you get to change E, giving yourself (conjecturally) a v. good chance of success.

(Ref: $y^2 = x^3 - x$ ... ≈ Grant - HW?)

(85)

The algorithm. Given $n > 2$, want to find a factor.

$K := 2$.

Step 1 Check $\gcd(n, 6) = 1$ & $n \neq m^r$ for some $r > 2$ (easy).

Step 2 choose random integers b, x_1, y_1 between 1 & n.

Step 3: let $c = y_1^2 - x_1^3 - b x_1$, & define $E: y^2 = x^3 + bx + c$ / $\mathbb{Q}$

let $Q = (x_1, y_1) \in E(\mathbb{Q})$.

Step 4 Compute $\gcd(4b^3 + 27c^2, n)$. If $\neq 1$, done.

(If $= 1$, then $E_{\mathbb{F}_p}$ is an elliptic curve over $\mathbb{F}_p$ $\forall p \mid n$).

If $= n$, go back to 3 & pick a new b.

Step 5 let $\ell = (\pi_K)^k$ (as in Pollard).

Step 6: Compute $\ell \cdot Q = \left(\frac{a_\ell}{d_\ell^2}, \frac{b_\ell}{d_\ell^3} \right) \in E(\mathbb{Q})$. The form of the coords come by looking at formulas for gp law, esp. duplication.

Step 7 $D := \gcd(d_\ell, n)$. If $1 < D < n$, done

- if $D = 1$, go to 5 & increase k or go to 2
- if $D = n$, go to 5 & increase k.

(86) 10

Why does it work? Say $\cdot\ p \mid n$

$\cdot$ for some E, have that $\#E(\mathbb{F}_p) \mid \ell$.

Then write $\ell Q = (d_k a_k : b_k : d_k^3)$ (integers a_k, b_k, d_k, $\sim$ coprime)

Then $\mathrm{red}_p(\ell Q) = (0:1:0)$, so $p \mid d_k^3 \ \forall k$, so $p \mid d_k$.

So $p \mid n$ & $p \mid d_k$ so $p \mid D$, so terminates (unless $D = n$, but again, unlikely congruence).