SPCS Cryptography Class Lecture 9

July 2, 2015

All the Sage codes we run in class can be found in this directory

`https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/`

Pollard's ρ method

Toy Pollard

To find a non-trivial factor of n, we

- Take $O(n^{1/4+\epsilon})$ random numbers.
- For any two such random numbers x, y, compute $gcd(x-y, n)$.
- If any of them is neither 1 nor n, we found a non-trivial factor of n.

How fast does this run? A big problem: There are $\asymp O(n^{1/4+\epsilon})^2 = O(n^{1/2+\epsilon})$ differences $x-y$ to compute - actually worse than the naive algorithm. Another problem is storage - we have to store too many numbers. Fortunately there is a way around this problem. Let's first describe Pollard's ρ-method and see an example, before coming back to see why we hope it works.

Pollard's ρ method

Example

We try to factorize $n = 55$.

- Consider the function $f(x) = x^2 + 2 \bmod n$. We pretend that

$$2, f(2), f(f(2)), f(f(f(2))), \cdots$$

  are random number mod 55, and we calculate just the consecutive difference, and look at the gcd of difference and n.
- In this case, the sequence $\pmod n$ is

$$2, 6, 38, 16, 36, \cdots$$

- $gcd(6-2, 55) = 1$, $gcd(38-6, 55) = 1$,
- $gcd(16-38, 55) = 11!$

So 5 is a factor of 55.

Pollard's ρ method

Example

We try to factorize $n = 45$.

- Consider the function $f(x) = x^2 + 2 \bmod n$. We pretend that

  $$2, f(2), f(f(2)), f(f(f(2))), \cdots$$

  are random number mod 45, and we calculate just the consecutive difference, and look at the gcd of difference and n.
- In this case, the sequence (mod n) is

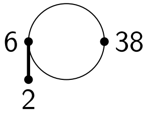
  $$2, 6, 38, 6, 38, \cdots$$

- $gcd(6-2, 45) = 1$ $gcd(38-6, 45) = 1$, $gcd(6-38, 45) = 1$

Any hope of getting a factor in this case?

Pollard's ρ method

A picture in this case:



How can we tell if the algorithm may fail quickly? i.e. Detecting looping.

Pollard's ρ method

An analogy: Alice is running on the track - how does she know if the track actually loops around quickly?

- She can just run until she loops around.
- or she can find Bob to help her, by running twice as fast and tell her that "Hey, it loops around."

This is the idea of Floyd's cycle-finding algorithm. With that, we state

Pollard's ρ method

Pollard's ρ method

- Choose a random polynomial function $f(x)$ that takes value in $\mathbb{Z}/n\mathbb{Z}$ and returns values in $\mathbb{Z}/n\mathbb{Z}$.
- Let $x = 2$ and $y = 2$.
- Replace x by $f(x)$ and y by $f(f(y))$.
- Compute $d = gcd(|x - y|, n)$.
- If $d = 1$, goes back to step 3. If $d = n$, then the algorithm fails, and we have to go back to step 1 and choose another random function or starting point for x, y. If $1 < d < n$, then congratulations! We found a factor of n.

Pollard's ρ method

Example

We factorize $n = 82123$, with $f(x) = x^2 + 1$ and starting point 2. We compute,

x	y	$gcd(x - y, n)$
5	26	1
26	47715	1
677	35794	1
47715	16651	1
25297	81447	1
35794	29766	1
9514	27554	41

So 41 is a factor. Now $\frac{82123}{41} = 2003$, which turns out to be a prime. So

$$82123 = 41 \times 2003$$

Pollard's ρ method

`https://cloud.sagemath.com/projects/113f54cf-25b5-4bf4-803f-c9bdef8b6eec/files/pollard_rho_method.sagews`

Running time?

- Heuristically this is $O(s(n)^{1/2})$, where n is the smallest prime factor of n.In particular, since $s(n) \leq n^{1/2}$, we see that the algorithm would run in $O(n^{1/4})$ steps, better than the naive algorithm $(O(n^{1/2})$ steps.
- In particular, if n has small prime factors - the algorithm can find them quickly.
- An important feature is that this takes $O(1)$-storage space, since you just compute the values as you go along - you don't need to store many numbers.
- Also, this is a *probabilistic* method. Whether you can find a factor depends on your luck - although the birthday problem suggests that you have a good chance of doing so, as long as n is composite.

Smooth numbers

The other algorithms we will discuss rely on a curious fact: there are lots of numbers all of whose prime factors are small. For example,

$$639146200 = 2^3 \cdot 5^2 \cdot 7^4 \cdot 11^3$$

Of course, "smallness" is relative, and so one really needs to quantify the meaning of smallness.

Definition

A number n is B-smooth if all the prime factors of n are at most B.

(Note: mistake in the book - B-smoothness in book is "Less than B". This is not the usual definition.)

Example

Is 12 2-smooth? 3-smooth?
Is 21 2-smooth? 3-smooth? 5-smooth?
639146200 above is 11-smooth.

Smooth numbers

Heuristically: around every (non-smooth) integer there are many smooth integers.

A similar concept is the B-power smoothness

Definition

A number n is B-power smooth if $n = \prod p_i^{e_i}$, and each prime power $p_i^{e_i} \leq B$.

Example

Is 12 2-powersmooth? 3-powersmooth? 4-powersmooth?
Is 21 5-powersmooth? 7-powersmooth?
Is 56 7-powersmooth?
639146200 is 2401-smooth.

$p - 1$ method

- The next method we discuss would depend on Fermat's little theorem. You may guess from the name that it has something to do factorization of $p - 1$.
- Recall that if p is a prime, and $(a, p) = 1$, then

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

  In other words, $p | a^{p-1} - 1$.
- Now, if $p - 1$ is B-power smooth, then $p - 1 | B!$ (why?).It is a general fact that if $m | n$, then

$$a^m - 1 | a^n - 1$$

  In our case, it means that $p | a^{p-1} - 1$ and $a^{p-1} - 1 | a^{B!} - 1$ as well. Thus, $p | a^{B!} - 1$.

$p-1$ method

- One would then know that

$$p \mid gcd(a^{B!}-1, n)$$

- If $gcd(a^{B!}-1, n) < n$, congratulations! We found a non-trival factor of n.
- Otherwise if $gcd(a^{B!}-1, n) = n$, we just replace a by something and try again. For simplicity we usually start with $a = 2$.

$p-1$ method

We want to find a non-trivial factor of n. Set $a = 2$ and B to be some number.

- Start with $j = 2$. Calculate $x = a^{j!} \bmod n$, then calculate $d = gcd(x-1, n)$.
- If $1 < d < n$, we found a factor of n.
- If not, repeat the last step with $j+1$ in place of j, unless j already hits our upper bound B.

$p - 1$ method

Example

Let $n = 5917$. We will take $a = 2$ and $B = 10$. Then,

j	x	d
2	4	1
3	64	1
4	2521	1
5	1648	61

So we found a factor 61. We calculate $\frac{5917}{61} = 97$, which turns out to be a prime. Thus $5917 = 61 \times 97$.

$p-1$ method

Summary:

- $p-1$ method is a special-purpose factorization method - it relies on one factor p of n satisfies $p-1$ being B-powersmooth. That means you have to be careful in choosing the primes for RSA as well.
- Run time: We will skip the analysis now - you can try that yourself. It would be something like $O(B \log B (\log n)^2)$.
- How large should B be?
  - If B is small - we probably can't find many factors.
  - If B is big, say up to $n^{1/2}$ size, then it would work but that will take too long - for example, taking B to be $n^{1/2}$ make it an $O(n^{1/2+\epsilon})$ algorithm, even worse than the naive one.
  - How large should we choose B? In practice people would often do a slightly different algorithm that won't require B-powersmoothness of $p-1$ (which is quite restrictive), and would take B to be of size around $[10^5, 10^6]$.
  - This allows computer to get a factor quickly if lucky. If not, we will get into more sophisticated test.

Baby baby Quadratic sieve

- The third method we will describe depends on one of the simplest identities in Mathematics - the difference of squares

$$x^2 - y^2 = (x - y)(x + y)$$

- If we want to factorize n, and if it happens that $n = x^2 - y^2$ for some x, y, then we factorized n!

Example

Factorize 21.

Baby baby Quadratic sieve

Example

We factorize $n = 25217$. By trial and error,

b	$n + b^2$	$\sqrt{n + b^2}$
0	25217	158.798614603529
1	25218	158.801763214393
2	25221	158.811208672436
3	25226	158.826949854236
4	25233	158.848984888163
5	25242	158.877311155495
6	25253	158.911925291968
7	25266	158.952823189775
8	25281	159.000000000000

So actually

$$25217 = 159^2 - 8^2 = 151 \cdot 167$$

Baby baby Quadratic sieve

We don't even need to do $x^2 - y^2 = n$ - changing right hand side by a little is fine.

Example

Factorize 20.

Solution

Of course, $20 = 6^2 - 4^2$. *But one can also see that*

$$4 \cdot 20 = 9^2 - 1^2$$

and so $(9-1)(9+1) \equiv 0 \mod 20$ - *there is a chance that* $9-1$ *or* $9+1$ *would have a non-trivial common factor with 20, unless we are unlucky.*

Dixon's method/Quadratic sieve

- So to factorize n, if we can find some x, y such that $x^2 \equiv y^2 \mod n$, hopefully that gives us some information, by computing $gcd(x-y, n)$ or $gcd(x+y, n)$.
- Of course we don't want $x \equiv \pm y \mod n$.

Dixon's method (more commonly known as quadratic sieve) is based on this simple idea - the method is about how you can find $x, y \mod n$.

- We will just take $a_1, \cdots, a_k \geq \sqrt{n}$, and consider $a_1^2, \cdots, a_k^2 \mod n$.
- If a_1^2 is congruent to some small ACTUAL square b_1^2 mod n, we win - compute $gcd(a_1 - b_1, n)$ and hope for the best.
- Note that we also win if some product of $a_1^2, \cdots, a_k^2$ is a square mod n. For example, if

$$a_1^2 \equiv b_1 \mod n, \text{ and } a_2^2 \equiv b_2 \mod n$$

such that $b_1 b_2$ is some square y^2 - then again $(a_1 a_2)^2 \equiv y^2 \mod n$, and we only need to calculate $gcd(a_1 a_2 - y, n)$ and pray to god.

Dixon's method/Quadratic sieve

Example

Factorize $n = 91$.

Solution

God tells me that

$$\begin{aligned} 14^2 &\equiv 14 \bmod 91 && \textit{and} && 14 = 2 \cdot 7 \\ 21^2 &\equiv 77 \bmod 91 && \textit{and} && 77 = 7 \cdot 11 \\ 29^2 &\equiv 22 \bmod 91 && \textit{and} && 22 = 2 \cdot 11 \end{aligned}$$

- *Left hand side are large squares, so that we actually need to mod 91 along the way.*
- *Right hand side have only small prime factors so that some product of them is a square. In this case,* $14 \cdot 77 \cdot 22$ *is a square.*

Dixon's method/Quadratic sieve

Solution

- *This gives us the relation*
$$(14 \cdot 21 \cdot 29)^2 \equiv (2 \cdot 7 \cdot 11)^2 \bmod 91$$

Then we calculate
$$gcd(14 \cdot 21 \cdot 29 - 2 \cdot 7 \cdot 11, 91) = gcd(8372, 91) = 91$$

$$gcd(14 \cdot 21 \cdot 29 + 2 \cdot 7 \cdot 11, 91) = gcd(8680, 91) = 7$$

So we found a factor 7.

Dixon's method

Let us generalize the idea just now,

Dixon's method

- Generate some numbers whose square mod n has only small prime factors, say, B-smooth after mod n.
- Choose appropriately some of them, so the product of right hand side is again a square. Call left hand side x^2, right hand side y^2. Then we have $x^2 \equiv y^2 \mod n$.
- Find $gcd(x - y, n)$ and pray to god.

Dixon's method/Quadratic sieve

Example

We factor $n = 914387$. Let's take $B = 11$, so that the computations

- There are 5 primes up to 11: 2,3,5,7,11. So $\pi(11) = 5$.
- We want to find some $a_i^2 \equiv b_i \mod n$, such that
  - a_i is not small - greater than $\sqrt{914387} \sim 956$, so that we actually need to mod 914387.
  - b_i is 11-smooth.
  - the product of b_i's would be a square itself.

God says,

$$\begin{aligned} 1869^2 &\equiv 750000 \mod 914387 &\quad \text{and} \quad & 750000 = 2^4 \cdot 3 \cdot 5^6 \\ 1909^2 &\equiv 901120 \mod 914387 &\quad \text{and} \quad & 901120 = 2^{14} \cdot 5 \cdot 11 \\ 3387^2 &\equiv 499125 \mod 914387 &\quad \text{and} \quad & 499125 = 3 \cdot 5^3 \cdot 11^3 \end{aligned}$$

These are actually the first three 11-smooth numbers larger than 956.

Dixon's method/Quadratic sieve

Example

- None of the numbers on the right are squares,but it happens that if we multiply them,

$$\begin{aligned}1869^2 \cdot 1909^2 \cdot 3387^2 &\equiv 750000 \cdot 901120 \cdot 499125 \bmod 914387\\ &\equiv (2^4 \cdot 3 \cdot 5^6) \cdot (2^{14} \cdot 5 \cdot 11) \cdot (3 \cdot 5^3 \cdot 11^3) \bmod 914\\ &= (2^9 \cdot 3 \cdot 5^5 \cdot 11^2)^2 \bmod 914387\\ &= 580800000^2 \bmod 914387\\ &\equiv 164255^2 \bmod 914387\end{aligned}$$

  We further note that $1869 \cdot 1909 \cdot 3387 \equiv 9835 \bmod 914387$, so we found our $x = 9835$, $y = 164255$!

- Now we hope for the best: Compute $gcd(914387, 9835 - 164225)$

$$gcd(914387, 9835 - 164255) = gcd(914387, 154420) = 1103.$$

Dixon's method/Quadratic sieve

Example

Hurray! We have factored $914387 = 1103 \cdot 829$. God does answer prayers.

How to implement this?

- Have to be careful about size of B. If too small, very few B-smooth numbers. If too big, very hard to do the next step - finding the "right" numbers to multiply and get a square.
- How to find $a_i^2 \equiv b_i \mod n$ which are B-smooth?
- Suppose we have a list of numbers $a_i^2 \equiv b_i \mod n$, where b_i are all B-smooth - how can we know which of them we should multiply to get a square?

Dixon's method/Quadratic sieve

The third question: how do we know what to multiply?

Example

Factorize 2043221. Suppose we pick $B = 11$, and we get

$$\begin{array}{lcr}
1439^2 \equiv 27500 \bmod 2043221 & \text{and} & 27500 = 2^2 \cdot 5^4 \cdot 11 \\
2878^2 \equiv 110000 \bmod 2043221 & \text{and} & 110000 = 2^4 \cdot 5^4 \cdot 11 \\
3197^2 \equiv 4704 \bmod 2043221 & \text{and} & 4704 = 2^5 \cdot 3 \cdot 7^2 \\
3199^2 \equiv 17496 \bmod 2043221 & \text{and} & 17496 = 2^3 \cdot 3^7 \\
3253^2 \equiv 365904 \bmod 2043221 & \text{and} & 365904 = 2^4 \cdot 3^3 \cdot 7 \cdot 11^2
\end{array}$$

Dixon's method/Quadratic sieve

The third question: how do we know what to multiply?

- Key question is: knowing how to solve a system of linear equations in $\mathbb{F}_2$!
- A basic idea of solving system of linear equations systematically is Gaussian elimination.
- It works over any *field* - an object where you can add, subtract, multiply, and divide by non-zero numbers.
- Examples: rational numbers $\mathbb{Q}$, real numbers $\mathbb{R}$, complex numbers $\mathbb{C}$, $\mathbb{F}_p$.

Dixon's method/Quadratic sieve

How to choose B?

- Note that in the third step, we need to generate many equations for the equation to be solvable.
- There are $\pi(B)$ equations - having $\pi(B)+1$ variables give us a great chance of finding one solution.
- This means we need $\pi(B)+1$ relations of the type $a_i^2 \equiv b_i \mod n$.
- $\pi(B)$ can't be too small comparing to n, otherwise searching takes forever.
- Point: we need many B-smooth numbers! (At least $\pi(B)+1$ of them)

Dixon's method/Quadratic sieve

How to choose B?

- This means we need to know the number of B-smooth numbers up to X, at least asymptotically. (Think about prime number theorem)
- B would also change as X is large.
- In our case, such a formula is due to Canfield-Erdos-Pomerance.
- It leads to the choice of B being $B = e^{\sqrt{\frac{\log X \log\log X}{2}}}$

Dixon's method/Quadratic sieve

How to find (many) $a_i^2 \equiv b_i \mod n$?

- Brute force: take $a > \sqrt{n}$, compute $a^2 \mod n$. Pray.
- Continued fraction: it's great that $a^2 \mod n$ is small. We can use continued fractions to find $\frac{a}{a'}$ as close to $\sqrt{kn}$ as possible for $k = 1, 2, \cdots$. Then $a^2 \approx b^2 kn$, thus $a^2 \mod n$ would be reasonably small, more likely to be B-smooth.
- Quadratic Sieve (Pomerance): look at the polynomial $x^2 - n$, sieve away multiples of prime powers up to B.

Running time of quadratic sieve:

$$e^{(1+o(1)) \log n \log \log n}$$

(Not polynomial time!)

Dixon's method/Quadratic sieve

What else can you do?

- Multi-Polynomial Quadratic Sieve: Optimizing in the quadratic sieve step, by using more polynomials.
- General number field sieve: Use another method to produce $x^2 \equiv y^2 \mod n$. Recall that we talked about "integers" of the form $a + b\sqrt{-5}$ in the first day. As we can talk about factoring etc as usual for these numbers, we may as well talk about squares. Main improvement of GNFS is that this way of searching for small squares on the right hand side takes subexponential time, beating the quadratic sieve.
- Elliptic Curve method.
- Shor's algorithm. (21 in 2012)