SPCS Cryptography Class Lecture 8

July 1, 2015

Pohlig-Hellman

- Sometimes special features of p would help.
- One thing that often helps is the *factorization of* $p-1$.
- Why would it help?The discrete log problem is really a problem modulo $p-1$.
- If we know the prime factorization of $p-1$, then by Chinese remainder theorem, we only need to understand the exponent mod each prime power factor of $p-1$.

Pohlig-Hellman

Let us see an example:

Example

Take $p = 11251$, $g = 23$ (a primitive root), and $h = 9689$. We wish to find k such that

$$23^k \equiv 9689 \bmod 11251$$

Remember that k is defined modulo $p - 1$.

Solution

- *Observe that* $p - 1 = 2 \cdot 3^2 \cdot 5^4$, *so by Chinese Remainder Theorem, it suffices to understand* $k \bmod 2, k \bmod 3^2$ *and* $k \bmod 5^4$.
- *How to figure out* $k \bmod 2$*?*
- *We can use the square test!For any* $a \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{2}} \equiv \begin{cases} 1 \bmod p & (\textit{if } a \textit{ is a square in } \mathbb{F}_p^*) \\ -1 \bmod p & (\textit{if } a \textit{ is not a square in } \mathbb{F}_p^*) \end{cases}$$

Pohlig-Hellman

Solution

- *In our case,*
  - *if* k *is even, then* 9689 *is a square, so* $9689^{\frac{p-1}{2}} \equiv 1 \mod p$
  - *if* k *is odd, then* 9689 *is not a square, so* $9689^{\frac{p-1}{2}} \equiv -1 \mod p$
- *Thus it suffices to check* $9689^{\frac{p-1}{2}}$ *to see if it is a square.By repeated squaring, one can check that*

$$9689^{\frac{p-1}{2}} \equiv -1 \mod 11251,$$

  so k *is odd, i.e.* $k \equiv 1 \mod 2$*.*
- *What about* $k \mod 3^2$*?*
- *We will first work out the baby case* $k \mod 3$*.*

Pohlig-Hellman

Solution

(Refined) Cube test

Let p be a prime such that $3|p-1$, g be a primitive root mod p *and* $a = g^k \in \mathbb{F}_p^*$,

$$a^{\frac{p-1}{3}} \equiv \begin{cases} 1 \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^0 \bmod p & \textit{(if } k \equiv 0 \bmod 3\textit{)} \\ g^{\frac{p-1}{3}} \bmod p \equiv \left(g^{\frac{p-1}{3}}\right)^1 \bmod p & \textit{(if } k \equiv 1 \bmod 3\textit{)} \\ g^{\frac{2(p-1)}{3}} \bmod p = \left(g^{\frac{p-1}{3}}\right)^2 \bmod p & \textit{(if } k \equiv 2 \bmod 3\textit{)} \end{cases}$$

- *Thus,* $k \bmod 3$ *can be figured out by checking*
$$9689^{\frac{p-1}{3}} \equiv (g^{\frac{p-1}{3}})^k \bmod p.$$
Repeated squaring gives $k \equiv 1 \bmod 3$.
- *Note: we are doing a simpler discrete log problem!*

Solution

- *To find* $k \bmod 3^2$*, develop the* 3^2*th-power test.*
- *In other words,* $k \bmod 3^2$ *can be figured out by checking*

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p$$

  We can find that $k \equiv 4 \bmod 3^2$*.*
- *Similarly, one sees that* $k \equiv 511 \bmod 5^4$*.*
- *Now solve* k *by Chinese remainder theorem for the system*

$$\begin{cases} k \equiv 1 \bmod 2 \\ k \equiv 4 \bmod 3^2 \\ k \equiv 511 \bmod 5^4 \end{cases}$$

- *The smallest solution would be* $4261 \bmod 11250$*. Therefore, the discrete log in this case is 4261, i.e.*

$$23^{4261} \equiv 9689 \bmod 11251.$$

Pohlig-Hellman

What if $p-1$ is has a large prime power?We can refine the process for 3^2 as follows.

- We want to find $k \bmod 3^2$.First find $k \bmod 3$ as before - we got $k \equiv 1 \bmod 3$.
- Write $k = 1 + 3k'$, where k' is 0,1 or 2 mod 3.Note that

$$9689^{\frac{p-1}{3^2}} \equiv (g^{\frac{p-1}{3^2}})^k \bmod p \equiv (g^{\frac{p-1}{3^2}})(g^{\frac{p-1}{3}})^{k'} \bmod p.$$

- Discrete log problem with the SAME base $g^{\frac{p-1}{3}}$ - so we can just use previous calculations.
- One can check that $k' = 1$ in this case.Therefore $k \equiv 1 + 3k' \bmod 3^2 \equiv 4 \bmod 3^2$.

Pohlig-Hellman

We will skip the run time analysis for Pohlig-Hellman, but to summarize,

- Baby-step-giant-step is a general approach that works for ANY prime p, and is a $O(p^{1/2+\epsilon})$ algorithm. (Subexponential, but not polynomial)
- Pohlig-Hellman is an approach that depends on the factorization of $p-1$.
- In particular, if $p-1$ has many small prime factors, Pohlig-Hellman would be much better.
- For example, if $p-1=2^k$ for some k (Fermat prime), then the running time for Pohlig-Hellman would be $O(k)=O(\log p)$, i.e. polynomial time!This shows that one has to be careful in choosing the prime p.

RSA Cryptosystem

- RSA is the second public-key cryptosystem we discuss, after ElGamal. It was published in 1978 by MIT scientists Ron Rivest, Adi Shamir and Leonard Adleman.
- It relies on taking e-th root mod n for some large integer n.
- A toy example, recall cube test again - we said that if $p \equiv 2 \mod 3$, then everything is a cube, i.e. for any $a \mod p$, there exists a unique $x \mod p$ with

$$x^3 \equiv a \mod p$$

  How can you find x given a and p?
- Key point: $gcd(3, p-1) = 1$!
- We can ask the same question if we replace 3 by something larger - say 5, 7, 2015, 1024576,... Call that e. We also want $gcd(e, p-1) = 1$. Same method would work.
- What if we take arbitrary modulus?

RSA Cryptosystem

Recall basic facts about Euler's totient function $\phi(n)$.

- $\phi(n)$ is the number of positive integers up to n that are relatively prime to n.
- $\phi(4)$? $\phi(5)$? $\phi(6)$?
- $\phi(p)$? $\phi(p^2)$? $\phi(p^{999})$?
- $\phi(n)$ is multiplicative - meaning that if m, n are relatively prime, then $\phi(mn) = \phi(m)\phi(n)$.
- For example, $\phi(6)$ vs $\phi(2)\phi(3)$.
- $\phi(2^5 \cdot 3^4)$?
- Most important theorem for us is,

Theorem (Euler's totient function theorem)

For a relatively prime to n,

$$a^{\phi(n)} \equiv 1 \mod n$$

RSA Cryptosystem

- What if we take an arbitrary modulus? i.e. given some n AND $\phi(n)$, $a \bmod n$, and e relatively prime to $\phi(n)$, we want to find $x \bmod n$ such that

$$x^e \equiv a \bmod n$$

- Why such a restriction on e?
- What if we don't know $\phi(n)$?

RSA Cryptosystem

RSA Problem

Given

- n without its factorization.
- $e \bmod \phi(n)$, known to be relatively prime to $\phi(n)$.
- $a \bmod n$, known to be relatively prime to n. Also, a is known to be an e-th power mod n.

Then it is hard to solve for x so that

$$x^e \equiv a \bmod n$$

- What if it is easy to factor n?
- What if we can find some factors of n easily?

Textbook RSA

Suppose Alice wants to send a message to Bob,

- Bob chooses two large primes p, q, and compute their product $n = pq$. Bob also computes $\phi(n) = (p-1)(q-1)$, and choose an exponent $(e, \phi(n)) = 1$. He then calculates $d = e^{-1} \mod \phi(n)$.
- Bob publishes his *public key* (n, e), and keeps his *private key* d safe and sound.
- Alice wants to send Bob a message m. Suppose that $(m, n) = 1$ and $0 < m < n$. (In practice this is not an issue, since n has only 2 prime divisors) She computes the ciphertext $c = m^e \mod n$, and sends it to Bob.
- To decrypt the message, Bob computes $m = c^d \mod n$.

RSA Cryptosystem

`http://logos.cs.uic.edu/340%20notes/rsa.html` Why does this work?
The message Bob recovers is

$$c^d \equiv (m^e)^d \bmod n = m^{de} \bmod n$$

But $de \equiv 1 \bmod \phi(n)$, so by Euler's totient function theorem,

$$m^{de} \equiv m \bmod n$$

hence decrypting the message.

RSA Cryptosystem

Is RSA secure?

- For *generic* n, e, d, people think that this is as hard as the factorization problem of n. But no one knows.
- That means you have to be VERY careful in choosing n, e, d.
- Some examples of what is bad...

Small d is bad:

Theorem (Wiener 1990)

Let $n = pq$, where p, q are primes satisfying $q < p < 2q$. Let the private key $d < \frac{1}{3}n^{1/4}$. Given the public key (n, e), Eve can efficiently recover d!

Small e is actually encouraged.

- Because people want to encrypt things quickly!
- If padding is done properly (we will explain this soon), small exponents shouldn't matter.
- Many people like to use $e = 3, 5, 17, 257, 65537$.

n should be large (of course) - currently 2048 bit would be considered safe.

What can you do if you can factorize

- 1995: Breaking a (hypothetical) black market. http://baby.indstate.edu/msattler/_sci-tech/comp/privacy//pgp/misc/blacknet-key-attack.html A 384-bit key cracked.
- 1998: Hop onto metro for free.. and 10 months in jail. A 321-bit key cracked for France bank cards.
- Up to 2007: Make some money. https://en.wikipedia.org/wiki/RSA_numbers
- 2009: Flash your favorite OS on TI-83 calculator. https://en.wikipedia.org/wiki/Texas_Instruments_signing_key_controversy A 512-bit key cracked for signature on TI-83.

RSA Cryptosystem

So how do you factorize numbers? We will talk about a few methods:

- Pollard's ρ method
- $p - 1$ method
- Quadratic sieve.

The birthday problem

Question

How many people must be in a room for it to more likely than not that two people share a birthday? (Assume that all birthdays are equally likely, and we are ignoring Feb 29)

Solution

23. Why? Let's calculate the probability for none of k people to share the same birthday.

- *First person: he can choose to be born on any day -* $\frac{365}{365} = 1$.
- *Second person: he cannot be born on the day that first guy is born -* $\frac{365}{365} \times \frac{364}{365}$.
- *Third person: he cannot be born on the day that first, second guy is born -* $\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365}$.

The birthday problem

Solution

With k people, the probability that none of them is born on the same day is

$$\frac{365}{365} \times \frac{364}{365} \times \frac{363}{365} \times \cdots \times \frac{365-k+1}{365}$$

Turns out that the first k where it falls below $\frac{1}{2}$ is 23.

What if you come from Mars, where a year has n days?In general you need around $\sqrt{2n \log n}$ people to have same birthday.

Pollard's ρ method

Why is this related to factorization at all? Goal: Given n, find one non-trivial factor m.

- Suppose that Martians don't talk about birthdays, but they talk about remainder mod m.
- It takes around $\sqrt{2m \log m}$ random numbers to have a good chance that some of them are the same mod m.
- Suppose that $x \equiv y \mod m$. Then we can calculate $gcd(x - y, n)$, and hope that this is a non-trivial factor of n.
- How large is m? If n is composite, there should be a non-trivial factor at most $\sqrt{n}$. So we may assume that $m \leq \sqrt{n}$ - this means that we only need to generate $O(n^{1/4} \log n) = O(n^{1/4+\epsilon})$ numbers.

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 $f(x) = x^2 + 2 \mod n$. 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

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

- $gcd(6-2, 55) = 1$, $gcd(38-6, 55) = 1$,
- $gcd(16-38, 55) = 1$,
- $gcd(36-16, 55) = 5!$

Pollard's ρ method

Example

We try to factorize $n = 45$.

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

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

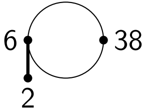
  are random number mod 45, and we calculate just the consecutive difference, and look at the gcd of difference and n.
- In this case, the sequence $\pmod{n}$ is

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

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

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