Computational Number Theory and Algebra July 2, 2012

Lecture 21

Lecturers: Markus Bläser, Chandan Saha *Scribe: Chandan Saha*

Our final topic of discussion on applications of smooth numbers is the following:

- The Index Calculus method for computing discrete logarithm.

1 The Index Calculus method

The index calculus is a probabilistic method for computing discrete logarithms in groups that are endowed with a notion of 'smoothness'. An example of such a group is $\mathbb{F}_p^\times$, which can be naturally identified with the numbers less than p. An element of $\mathbb{F}_p^\times$ is 'smooth' if as an integer it is a smooth number. The basic idea of the index calculus method is simple, and apparently this too can be traced back to the work of Kraitchik [Kra26, Kra29]. It was later rediscovered and analyzed by several other mathematicians including Adleman [Adl79].

Recall that, the discrete logarithm problem over $\mathbb{F}_p^\times$ is the task of finding an integer $x \in [0, p-2]$ such that $a^x = b \bmod p$, given a generator a and an arbitrary element b of $\mathbb{F}_p^\times$. Going by the conventional notation, we write $x = \log_a b \bmod (p-1)$. As before, let y be the smoothness parameter, to be fixed later, and $p_1, \ldots, p_k$ be the primes less than y. The index calculus starts by picking a random integer $\alpha \in [0, p-2]$ and computing $a^\alpha \bmod p$. If $a^\alpha \bmod p$ is y-smooth, we factor it completely as $a^\alpha = p_1^{e_1} \ldots p_k^{e_k} \bmod p$. This gives us the following linear equation in the *indices*,

$$\alpha = e_1 \log_a p_1 + \ldots + e_k \log_a p_k \mod (p-1). \tag{1}$$

Here, $\log_a p_1, \ldots, \log_a p_k$ are the unknowns. If we collect $4k$ such linear equations then it is 'quite likely' that k of the $4k$ equations are linearly independent and we can solve for the $\log_a p_i$'s modulo $p-1$. Now pick another random integer $\beta \in [0, p-2]$ and compute $a^\beta b \bmod p$. If $a^\beta b \bmod p$ is also y-smooth then we get another linear relation of the form,

$$\beta + \log_a b = f_1 \log_a p_1 + \ldots + f_k \log_a p_k \mod (p-1), \tag{2}$$

by factoring as, $a^\beta b = p_1^{f_1} \ldots p_k^{f_k} \bmod p$. But this time $\log_a b$ is the only unknown and we can simply solve it from the above equation. This is the basic idea of the index calculus method. To formalize it, we need to address the following three questions:

- How likely is it that $a^\alpha \bmod p$ and $a^\beta b \bmod p$ are y-smooth ?
- What is the chance that k out of the $4k$ equations are 'linearly independent' modulo $p-1$?
- How do we solve linear equations modulo $p-1$?

We have already seen the answer to the first question. Since α and β are randomly chosen and a is a generator of $\mathbb{F}_p^\times$, both a^α and $a^\beta b$ modulo p are uniformly distributed among the integers in $[1, p-1]$. Therefore, the probability that each of them is y-smooth is $\frac{\psi(p-1,y)}{p-1}$. The second question is also not difficult to answer. Using a counting argument it can be shown that the determinant of the coefficient matrix formed by k of the $4k$ linear equations is invertible modulo $p-1$ with high probability. Also, solving a set of linear equations modulo $p-1$ (in fact any integer) is just like solving modulo a prime. In the Gaussian elimination phase if we fail to invert an element then using that element we can easily factor $p-1$ and continue solving modulo the factors. In the end we can compose the different modular solutions using Chinese Remaindering. All

these can be done in $\tilde{O}(k^3)$ time, hiding some polylog factors in p.

We are now ready to present the algorithm. We will show in the analysis that the optimum choice of y is $e^{(2^{-1/2}+o(1))\sqrt{\ln p \ln\ln p}}$.

Algorithm 1 Index calculus method

1. Find all the primes, $p_1, \ldots, p_k$, less than $y = e^{(2^{-1/2}+o(1))\sqrt{\ln p \ln\ln p}}$.
2. Set $i = 1$.
3. while $i \leq 4k$ do
4. Choose integer α_i randomly from $[0, p-2]$. Compute $\gamma_i = a^{\alpha_i} \mod p$.
5. If γ_i is y-smooth, let $v_i = (e_{i1}, \ldots, e_{ik})$ where $\gamma_i = p_1^{e_{i1}} \ldots p_k^{e_{ik}}$. Set $i = i + 1$.
6. If $i = 4k + 1$, check if $v_1, \ldots, v_{4k}$ span $\mathbb{Z}_{p-1}^k$. If not, goto step 2.
7. Solve for $\log_a p_i$, $1 \leq i \leq k$ modulo $p-1$ using equation 1 by Gaussian elimination on $v_1, \ldots, v_{4k}$, and Chinese remaindering to compose the solutions.
8. Keep choosing integer β randomly from $[0, p-2]$ till $a^\beta b \mod p$ is y-smooth.
9. Solve for $\log_a b$ using equation 2.

Time complexity - The total time spent by the algorithm is dominated by the time spent in the `while`-loop and the time spent in steps 7 and 8. In step 4, the probability that γ_i is a y-smooth number is about $\frac{\psi(p,y)}{p} \approx u^{-u}$, where $u = \frac{\ln p}{\ln y}$ (by Lemma 4 in lecture 18). Checking if γ_i is y-smooth in step 5 takes roughly $\tilde{O}(k)$ time, hiding some polylog p factors. Since, $v_1, \ldots, v_{4k}$ span $\mathbb{Z}_{p-1}^k$ with high probability, the expected time spent in the loop is $\tilde{O}(k^2 u^u)$. Time taken for Gaussian elimination in step 7 is about $\tilde{O}(k^3)$, and the expected time spent in step 8 is $\tilde{O}(ku^u)$. So, the expected total time spent by the algorithm is $\tilde{O}(k^3 + k^2 u^u)$. This expression is minimized when $u \approx \sqrt{\frac{2 \ln p}{\ln\ln p}}$, implying that $y = e^{(2^{-1/2}+o(1))\sqrt{\ln p \ln\ln p}}$. Therefore, the expected time taken by the algorithm is $e^{(3/\sqrt{2}+o(1))\sqrt{\ln p \ln\ln p}}$.

Pomerance [Pom87] showed that using an elliptic curve method for fast smoothness test, the complexity of the index calculus method can be brought down to $e^{(\sqrt{2}+o(1))\sqrt{\ln p \ln\ln p}}$. For further details on modifications of the index calculus method, refer to the survey by Schirokauer, Weber and Denny [SWD96].

Exercises:

1. Show that k of the $4k$ vectors $v_1, \ldots, v_{4k}$ collected in steps 4-6 of the above algorithm are linearly independent modulo $p-1$ with high probability.

References

[Adl79] Leonard Adleman. A subexponential algorithm for the discrete logarithm problem with applications to cryptography. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science*, pages 55–60, 1979.

[Kra26] M. Kraitchik. Théorie des Nombres, Tome II. 1926.

[Kra29] M. Kraitchik. Recherches sur la Théorie des Nombres, Tome II. 1929.

[Pom87] Carl Pomerance. Fast, rigorous factorization and discrete logarithm algorithms. In *Discrete Algorithms and Complexity*, pages 119–143. Academic Press, 1987.

[SWD96] Oliver Schirokauer, Damian Weber, and Thomas F. Denny. Discrete logarithms: The effectiveness of the index calculus method. In *ANTS*, pages 337–361, 1996.