Lecture 19

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

In today's class, we will discuss a randomized subexponential time algorithm for integer factoring, namely:

- Dixon's random square method.

1 Dixon's random square method

Let N be an odd composite with ℓ distinct prime factors, where $\ell > 1$ i.e. N is not a perfect power. Choose an integer α randomly in the range $[1, N-1]$ and compute $\gamma = \alpha^2 \bmod N$. We will show later (in Lemma 1) that with reasonable probability γ is y-smooth for a suitable choice of y. Further, if we are lucky then the prime factorization of $\gamma = p_1^{e_1} \dots p_k^{e_k}$ could be such that each e_i is even. Here, $p_1, \dots, p_k$ are the primes less than y. This means, $\beta = p_1^{e_1/2} \dots p_k^{e_k/2}$ is also a root of γ. Notice that, the value of β is uniquely determined by the factorization of γ and is *independent* of which random root α (of γ) is chosen initially. Since there are 2^ℓ distinct roots of γ in $\mathbb{Z}_N$ (assuming $\gcd(\gamma, N) = 1$) (why?), the probability that $\alpha = \pm\beta \bmod N$ is only $\frac{1}{2^{\ell-1}}$. Therefore, if all goes well, $\gcd(\alpha + \beta, N)$ is nontrivial. Unfortunately, the chance that all the e_i's are even is not quite sufficient to ensure good success rate of the algorithm. But there is a neat way around!

Instead of choosing a single α, suppose we choose $k+1$ random numbers, $\alpha_1, \dots, \alpha_{k+1}$, in the range $[1, N-1]$ such that $\gamma_j = \alpha_j^2 \bmod N$ is y-smooth for every j, $1 \le j \le k+1$. Consider the vectors $v_j = (e_{j1} \bmod 2, \dots, e_{jk} \bmod 2)$ corresponding to the prime factorizations of $\gamma_j = p_1^{e_{j1}} \dots p_k^{e_{jk}}$ for $1 \le j \le k+1$. Since there are $k+1$ vectors in a k dimensional space over $\mathbb{F}_2$, there exists a collection of vectors, $v_1, \dots v_m$ (say) such that their sum (over $\mathbb{F}_2$) is zero. What this means is that the prime factorization of $\gamma = \gamma_1 \dots \gamma_m = p_1^{e_1} \dots p_k^{e_k}$ has every e_i even. Once again, it can be easily argued that $\beta = p_1^{e_1/2} \dots p_k^{e_k/2} \bmod N$ and $\alpha = \alpha_1 \dots \alpha_m \bmod N$ are 'distinct' roots of $\gamma \bmod N$ with high probability.

We need to fill in one missing piece to formalize the above approach. It is to show that, for a random α, $\gamma = \alpha^2 \bmod N$ is y-smooth with high probability. The following lemma proves this fact.

Lemma 1 *Let $p_1, \dots, p_k$ be the primes less than y in increasing order, and $S(N, y)$ be the following set,*

$$S(N, y) = \{\alpha : 1 \le \alpha \le N-1 \text{ and } \gamma = \alpha^2 \bmod N \text{ is } y\text{-smooth}\}.$$

If $\gcd(p_i, N) = 1$ for all $1 \le i \le k$ and $r \in \mathbb{Z}^+$ is such that $p_k^{2r} \le N$ then $|S(N, y)| \ge \frac{k^{2r}}{(2r)!}$.

Proof Let $N = q_1^{f_1} \dots q_\ell^{f_\ell}$ be the prime factorization of N. Since N has ℓ prime factors, every element $\alpha \in \mathbb{Z}_N$ is naturally associated with an element v_α in $\{1, -1\}^\ell$ that represents its *quadratic character*. That is, the i^{th} index of v_α is 1 if α is a square modulo $q_i^{f_i}$ and -1 otherwise. Moreover, by Chinese Remaindering Theorem, if two elements in $\mathbb{Z}_N$ have the same quadratic character then their product is a square modulo N (why?). Define a set T as,

$$T = \{\alpha : \alpha = p_1^{e_1} \dots p_k^{e_k} \text{ where } \sum_{i=1}^{k} e_i = r\} \tag{1}$$

Surely, every element of T is less than or equal to $\sqrt{N}$ as $p_k^{2r} \le N$. The 2^ℓ possible values of the quadratic character define a partition of the set T. Call the partition corresponding to the quadratic character $g \in \{1, -1\}^\ell$ as T_g, so that $T_g = \{\alpha \in T : v_\alpha = g\}$. Every pair of elements within the same partition can be multiplied to form a square modulo N that is also a y-smooth number less than N. But there could be

repetitions as the same square can result from multiplying different pairs of elements. Using the fact that $\sum_{i=1}^{k} e_i = r$ (in equation 1), a simple counting argument shows that there can be at most $\binom{2r}{r}$ repetitions for each square thus obtained. Therefore,

$$|S(N,y)| \geq \frac{2^\ell}{\binom{2r}{r}} \sum_{g \in \{1,-1\}^\ell} |T_g|^2.$$

The factor 2^ℓ in the above expression is because every square in $\mathbb{Z}_N^\times$ has exactly 2^ℓ roots. Simplifying using Cauchy-Schwartz inequality, $|S(N,y)| \geq \frac{1}{\binom{2r}{r}} (\sum_{g \in \{1,-1\}^\ell} |T_g|)^2 = \frac{1}{\binom{2r}{r}} |T|^2$. Once again, using the fact that $\sum_{i=1}^{k} e_i = r$, size of the set T is $\binom{k+r-1}{r} \geq \frac{k^r}{r!}$ and hence, $|S(N,y)| \geq \frac{k^{2r}}{(2r)!}$. ■

We are almost done. All we need to do now is fix a value for y. This we will do in the analysis of the following algorithm and show that the optimum performance is obtained when $y = e^{O(\sqrt{\ln N \ln\ln N})}$.

Algorithm 1 Dixon's random square method

1. `Find all the primes,` $p_1, \ldots, p_k$ `, less than` $y = e^{(2^{-1/2}+o(1))\sqrt{\ln N \ln\ln N}}$.
2. `If` $p_i | N$ `for any` $1 \leq i \leq k$, `return` p_i.
3. `Set` $i = 1$.
4. `while` $i \leq k+1$ `do`
5. `Choose integer` α_i `randomly from` $[1, N-1]$. `If` $\gcd(\alpha_i, N) \neq 1$ `return it.`
6. `Else, compute` $\gamma_i = \alpha_i^2 \mod N$.
7. `If` γ_i `is` y`-smooth, let` $v_{\gamma_i} = (e_{i1} \mod 2, \ldots, e_{ik} \mod 2)$ `where` $\gamma_i = p_1^{e_{i1}} \ldots p_k^{e_{ik}}$. `Set` $i = i+1$.
8. `Find` $I \subset \{1, \ldots, k+1\}$ `such that` $\sum_{i \in I} v_{\gamma_i} = 0$ `over` $\mathbb{F}_2$.
9. `Let` $\gamma = \prod_{i \in I} \gamma_i = p_1^{e_1} \ldots p_k^{e_k}$, $\beta = p_1^{e_1/2} \ldots p_k^{e_k/2} \mod N$ `and` $\alpha = \prod_{i \in I} \alpha_i \mod N$.
10. `If` $\gcd(\alpha + \beta, N) \neq 1$ `return the factor. Else, goto step 3.`

Time complexity - Using the Sieve of Eratosthenes, we can find the first k primes in step 1 in $O(k \log^2 k \log\log k)$ time. The divisions in step 2 take $O(k \cdot \mathsf{M_I}(n))$ time, where $n = \ln N$. Each iteration of the `while`-loop in step 4 takes $O(\mathsf{M_I}(n) \log n)$ operations for the gcd computation and modular operations in steps 5 and 6. Whereas, trial divisions in step 7 can be done in $O((k+n)\mathsf{M_I}(n))$ time. In step 8, we can use Gaussian elimination to find the linearly dependent vectors in $O(k^3)$ time. Modular computations and gcd finding can be done in steps 9 and 10 using $O(\mathsf{M_I}(n) \log n)$ operations. Finally, we need to bound the number of iterations of the `while`-loop.

By Lemma 1, expected number of iterations to find a y-smooth square is $\frac{N \cdot (2r)!}{k^{2r}} \leq N \cdot (\frac{2r \ln y}{y})^{2r}$, taking $k \approx \frac{y}{\ln y}$. Now, if we choose y such that $y^{2r} = N$ then the expected number of executions of the loop is bounded by $(k+1) \cdot n^{2r}$. One final thing to notice is that the algorithm fails to return a proper factor at step 10 with only a constant probability (why?), meaning the expected number of times we need to jump to step 3 and re-run the algorithm is also a constant. Therefore, the expected running time of the algorithm is $\tilde{O}(k^3 + k^2 n^{2r+2}) = \tilde{O}(e^{\frac{3n}{2r}} + e^{\frac{2n}{2r}} n^{2r+2})$, as $y^{2r} = N$ (here, $\tilde{O}$ hides lower order terms in k and n). This expression is minimized if $2r \approx \sqrt{\frac{2n}{\ln n}}$. This means, $y = e^{(2^{-1/2}+o(1))\sqrt{\ln N \ln\ln N}}$ and the expected time taken by Dixon's random square method is $e^{(\sqrt{\frac{9}{2}}+o(1))\sqrt{\ln N \ln\ln N}}$.

The best known running time for Dixon's algorithm is $e^{(\sqrt{\frac{4}{3}}+o(1))\sqrt{\ln N \ln\ln N}}$, a result due to Pomerance [Pom82] and Vallée [Val89]. But this is surpassed by the work of Lenstra and Pomerance [JP92] which has an expected time complexity of $e^{(1+o(1))\sqrt{\ln N \ln\ln N}}$.

Despite the theoretical progress, the performance of these randomized algorithms are not very encouraging in practice. With the widespread use of the RSA cryptosystems in various commercial applications, there is just too much at stake. One cannot help but wonder if there are variants of these ideas that are more efficient in practice, which in turn may deem some of the RSA public keys as unsafe. The Quadratic Sieve method is one such practical improvement. We will discuss this in the next class.

References

[JP92] Hendrik W. Lenstra Jr. and Carl Pomerance. A Rigorous Time Bound for Factoring Integers. *Journal of the American Mathematical Society*, 5:483–516, 1992.

[Pom82] Carl Pomerance. Analysis and comparison of some integer factoring algorithms. In H.W. Lenstra Jr. and R. Tijdeman, editors, *Computational Methods in Number Theory*, volume 154 of *Math. Centrum Tract*, pages 89–139, 1982.

[Val89] B. Vallée. Provably fast integer factoring with quasi-uniform small quadratic residues. In *STOC '89: Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 98–106, 1989.