Computational Number Theory and Algebra June 18, 2012

Lecture 17

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

In today's class, we will see an interesting application of the LLL algorithm in breaking the RSA cryptosystem. The method, due to Coppersmith [Cop97], shows the vulnerability of the RSA when the exponent of the public key is small and a significant fraction of the message is already known to all as a header. The key step is the use of lattice basis reduction in finding a root of a modular equation, when the root is much smaller than the modulus in absolute value. Today, we will discuss the following topics:

- Breaking Low Exponent RSA,
- Brief account of the complexity of existing integer factoring and discrete logarithm algorithms, and
- Pollard-Strassen integer factoring algorithm.

1 Breaking Low Exponent RSA

Following the notations introduced in Lecture 1, let m be the message string and (n, e) be the public key, where e is the exponent and n is the modulus. The encrypted message is $c = m^e \bmod n$. Suppose that the first ℓ bits of the message m is a secret string x, and the remaining bits of m form a header b that is known to all. In other words, $m = b \cdot 2^\ell + x$, where b is known and x in unknown. Since, $(b \cdot 2^\ell + x)^e = c \bmod n$, we get an equation $g(x) = x^e + \sum_{i=0}^{e-1} a_i x^i = 0 \bmod n$, where $a_0, \ldots, a_{e-1}$ are known and are all less than n. Decryption of c is equivalent to finding a root of g modulo n. We intend to solve this modular root finding problem using lattice basis reduction.

The idea is to transform the modular root finding problem into a root finding problem over integers. The latter problem can then be solved using the LLL algorithm. To achieve this, we are going to do something very similar to what has been done in Algorithm 2 (in Lecture 16) for polynomial factorization. Solve for a nonzero $f \in \mathbb{Z}[x]$ of degree less than $k > e$, such that $f = gh \bmod n$ for some $h \in \mathbb{Z}[x]$ and $\|f\|$ is 'small'. Treat k as a parameter. (We will see an example case where $k = 2e$.)

Consider the polynomials $x^{k-e-1}g, \ldots, xg, g$ and $nx^{e-1}, \ldots, n$, and $v_1, \ldots, v_k$ be the coefficient vectors in $\mathbb{Z}^k$ associated with these k polynomials. Notice that, the vectors $v_1, \ldots, v_k$ are linearly independent and the volume of the lattice $\mathcal{L} = \sum_{i=1}^{k} \mathbb{Z} v_i$ is $\mathsf{vol}(\mathcal{L}) = n^e$. By Minkowski's theorem (see Lecture 15), the length of the shortest vector in $\mathcal{L}$ is at most $\sqrt{k} n^{\frac{e}{k}}$. It is easy to observe that the coefficient vector v_f of any solution f is in the lattice $\mathcal{L}$. Using LLL algorithm, we can find a short vector $v \in \mathbb{Z}^k$ in $\mathcal{L}$, in time $\mathsf{poly}(k, \log n)$, such that $\|v\| \leq 2^{\frac{k-1}{2}} \cdot \sqrt{k} n^{\frac{e}{k}}$. Let f be the polynomial with v as its coefficient vector. Surely, $f = gh \bmod n$ and $\deg(f) < k$.

We know that the secret string x has absolute value at most 2^ℓ. Let $x = a < 2^\ell$ be a root of g modulo n. Therefore, a is also a root of f modulo n. The absolute value of $f(a)$ is bounded by,

$$|f(a)| \leq k \cdot \|v\| \cdot 2^{\ell(k-1)} \leq k \cdot 2^{\frac{k-1}{2}} \cdot \sqrt{k} n^{\frac{e}{k}} \cdot 2^{\ell(k-1)} = A \text{ (say)}.$$

Now, if it happens that $A < n$ then $f(a) = 0$ over $\mathbb{Z}$. This means, we can apply Algorithm 2 (in Lecture 16) to factor f and thereby find the secret string x.

Let us fix some parameters to get a feel of how this argument works in practice. Let $e = 3$ and $k = 6$, then all we need to find x is,

$$6\sqrt{6} \cdot 2^{\frac{5}{2}} \cdot \sqrt{n} \cdot 2^{5\ell} < n \;\Rightarrow\; \ell < \frac{1}{10} \cdot \log n - \frac{1}{5} \cdot \log(48\sqrt{3}).$$

Therefore, if roughly $\frac{9}{10}$-th of the message is known then the remaining secret $\frac{1}{10}$-th part of the message can be recovered in polynomial time. Since the approach works in time polynomial in k and $\log n$, it is efficient only if k is small. This means, from the relation $A < n$, that e has to be small. This shows that a low exponent e along with a small fraction of the secret part of the message make RSA vulnerable to lattice based attacks.

2 Brief history of integer factoring and discrete logarithm

2.1 Integer factoring

Factoring an integer N is closely related to primality testing. However, this is a long standing open problem and as yet no randomized polynomial time algorithm is known. The current best randomized algorithm, due to Lenstra and Pomerance [JP92], has an expected running time of $e^{(1+o(1))\sqrt{\ln N \ln\ln N}}$ bit operations. This bound is an improvement on the complexity of *Dixon's algorithm* [Dix81], which also has an expected subexponential running time of the form $e^{O(\sqrt{\ln N \ln\ln N})}$. We will discuss Dixon's algorithm in a later lecture. The best deterministic factoring algorithm, known as Pollard-Strassen method, runs in $\tilde{O}(N^{\frac{1}{4}})$ time.

In practice, two important (heuristic) techniques are used to factor integers more efficiently than the theoretical bounds mentioned above. They are the *Quadratic Sieve* [Pom84] and the *Number Field Sieve* [LJMP90]. A heuristic analysis of the Quadratic Sieve (QS) yields a time bound of $e^{(1+o(1))\sqrt{\ln N \ln\ln N}}$ operations. We will have a discussion on QS in a later class. The Number Field Sieve (NFS) has a heuristic time complexity of $e^{(c+o(1))((\ln N)^{\frac{1}{3}}(\ln\ln N)^{\frac{2}{3}})}$ bit operations, for a constant $c > 0$. The best known value for c is 1.902, due to Coppersmith [Cop93]. For an exposition to NFS and some other related factoring algorithms refer to the survey by Lenstra [Len00] and the article by Stevenhagen [Ste08] in [BS08]. Although, the NFS is asymptotically faster than the QS, it has been noticed that the latter performs better in practice for numbers up to 100 digits (see [Pom96]).

2.2 Discrete logarithm

The discrete logarithm problem is the following - Given a finite group $(G, \cdot)$ (in 'some form', say as a list of generators) and two elements a and b in G find an $x \in \mathbb{N}$ such that $a^x = b$, if such an x exists. We have already seen an application of the conjectured hardness of this problem in the Diffie-Hellman protocol (Lecture 1). Like the order finding problem, the hardness of computing discrete log depends on the choice of the underlying group. For instance, if the underlying group is $(\mathbb{Z}_n, +)$ then the value of x is precisely $ba^{-1} \bmod n$, assuming $\gcd(a, n) = 1$. On the other hand, a polynomial time algorithm for solving discrete log over $(\mathbb{Z}_n^\times, \cdot)$ for a composite n, yields a randomized polynomial time algorithm for factoring n (this is left as an exercise).

In the cryptanalysis of the Diffie-Hellman protocol, we are interested in the hardness of the discrete log problem for the group $\mathbb{F}_p^\times$. For certain choices of p, this problem turns out to be easy. Pohlig and Hellman [PH78] showed that if all the factors of $p-1$ have values bounded by $\log^c p$ for a constant c, then the problem can be solved in polynomial time. Therefore, such choices of p should be avoided for the security of the protocol.

The current best randomized algorithm for the discrete log problem over $\mathbb{F}_p^\times$ is due to Pomerance [Pom87]. It has an expected running time of $e^{(\sqrt{2}+o(1))\sqrt{\ln p \ln\ln p}}$ bit operations. The technique used in [Pom87] is more generally known as the *index calculus method* and was used earlier by Adleman [Adl79] to obtain a subexponential discrete log algorithm having similar complexity. In a later lecture, we will discuss the index calculus method. Using an adaptation of the Number Field Sieve, Gordon [Gor93] proposed a heuristic algorithm showing a running time of $e^{(c+o(1))(\ln p)^{\frac{1}{3}}(\ln\ln p)^{\frac{2}{3}}}$ operations, where $c > 0$ is a constant. The best known value of c is 1.92. So far the best deterministic algorithms have $\tilde{O}(\sqrt{p})$ complexity (see [Pol78]). Despite the resemblance between the time complexity of integer factorization and discrete logarithm over

$\mathbb{F}_p^\times$, so far no reduction is known from one problem to the other. In fact, it was asked by Bach [Bac84] whether factoring $p-1$ reduces to finding discrete logarithm in $\mathbb{F}_p^\times$.

There are certain groups for which computing discrete logarithm is even harder, in the sense that no randomized subexponential time algorithm is known. Such an example is the group of points on an elliptic curve. In this case the precise question is the following - Given an elliptic curve $E_{p,a,b} = \{(x,y) \mid x,y \in \mathbb{F}_p \text{ and } y^2 = x^3 + ax + b \pmod p\} \cup \{0\}$, with $\gcd(p, 4a^3 + 27b^2) = 1$ and two points P and Q on $E_{p,a,b}$, find an n such that $P = nQ$. Here, $nQ \triangleq Q + Q + \ldots$ n times, where $+$ is the operation that endows $E_{p,a,b}$ with a group structure. The hardness of this problem has also been exploited in designing public key cryptosystems and digital signature schemes (refer to [Kob87]). For further details on discrete logarithm refer to the surveys by Odlyzko [Odl00] and McCurley [McC90], or the articles by Pomerance [Pom08] and Schirokauer [Sch08] in [BS08].

3 Pollard-Strassen integer factoring

Let N be a given composite number. Then there is a factor of N whose value is less than $M = \lceil N^{\frac{1}{2}} \rceil$. Let $b = \lceil M^{\frac{1}{2}} \rceil$ and consider the polynomial $f(x) = (x+1)(x+2)\ldots(x+b)$. It is clear that at least one of the numbers $a_i = f(ib) \mod N$, for $1 \le i < b$, shares a non-trivial gcd with N. Let this number be a_k. This implies that one of the numbers $kb+1, kb+2, \ldots, kb+b$ divides N, and therefore it yields a proper factor of N. This is the idea behind Pollard-Strassen's integer factoring algorithm.

Algorithm 1 Pollard-Strassen integer factoring

1. Let $M = \lceil \sqrt{N} \rceil$ and $b = \lceil \sqrt{M} \rceil$.
2. Compute $f = \prod_{j=1}^{b} (x+j) \mod N$, recursively, using polynomial multiplication in $\mathbb{Z}_N$.
3. Use multipoint evaluations in $\mathbb{Z}_N$ to compute $a_i = f(ib) \mod N$ for all $0 \le i < b$.
4. Let $k = \min\{i \mid \gcd(a_i, N) \neq 1\}$. If no such k exists, declare 'N is prime'.
5. Otherwise, go over $kb+1, \ldots, kb+b$ and output the minimum $kb+j$ that divides N.

Time and space complexity - It follows from the above discussion that the algorithm is correct. We leave it as an exercise to show that the time complexity of the algorithm is $\tilde{O}(N^{\frac{1}{4}})$ bit operations, where the $\tilde{O}$ notation hides some $\mathsf{poly}(\log N)$ factor. Since we are using multipoint evaluation in step 3, we need to store the numbers $\{a_i\}_{1 \le i < b}$, which makes the space complexity of the algorithm, $\tilde{O}(N^{1/4})$ bits.

Remark - Both the time and space complexity of the above algorithm can be brought down to $\tilde{O}(s(N)^{1/2})$, where $s(N)$ is the smallest prime factor of the composite number N. This is done by a slight modification of Algorithm 1: Instead of setting $b = \lceil \sqrt{M} \rceil$, run the algorithm with $b = 1, 2, 2^2, 2^3, \ldots$ and so on. It is an exercise to observe that the running time of this modified version of Algorithm 1 is $\tilde{O}(s(N)^{1/2})$ bit operations, while the space complexity is $\tilde{O}(s(N)^{1/2})$ bits.

Exercises:

1. Show that a deterministic polynomial time algorithm for solving the discrete log problem over $(\mathbb{Z}_n^\times, \cdot)$ for a composite n, yields a randomized polynomial time algorithm for factoring n.
2. Show that the time and space complexity of (the modified version of) Algorithm 1 are indeed $\tilde{O}(s(N)^{1/2})$, where $s(N)$ is the smallest prime factor of N.

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.

[Bac84] Eric Bach. Discrete Logarithms and Factoring. Technical report, Berkeley, CA, USA, 1984.

[BS08] Joseph P. Buhler and Peter Stevenhagen, editors. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography*, volume 44 of *Mathematical Sciences Research Institute Publications*. Cambridge University Press, 2008.

[Cop93] Don Coppersmith. Modifications to the number field sieve. *Journal of Cryptology*, 6:169–180, 1993.

[Cop97] Don Coppersmith. Small Solutions to Polynomial Equations, and Low Exponent RSA Vulnerabilities. *J. Cryptology*, 10(4):233–260, 1997.

[Dix81] John D. Dixon. Asymptotically Fast Factorization of Integers. *Mathematics of Computation*, 36(153):255–260, 1981.

[Gor93] Daniel M. Gordon. Discrete Logarithms in *GF(p)* Using the Number Field Sieve. *SIAM J. Discrete Math.*, 6(1):124–138, 1993.

[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.

[Kob87] Neal Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[Len00] Arjen K. Lenstra. Integer factoring. *Des. Codes Cryptography*, 19(2/3):101–128, 2000.

[LJMP90] Arjen K. Lenstra, Hendrik W. Lenstra Jr., Mark S. Manasse, and John M. Pollard. The Number Field Sieve. In *STOC*, pages 564–572, 1990.

[McC90] Kevin S. McCurley. The Discrete Logarithm Problem. In *Proceedings of Symposia in Applied Mathematics*, volume 42, pages 49–74, 1990.

[Odl00] Andrew M. Odlyzko. Discrete Logarithms: The Past and the Future. *Des. Codes Cryptography*, 19(2/3):129–145, 2000.

[PH78] S. Pohlig and M. Hellman. An improved algorithm for computing logarithms over GF(p) and its cryptographic significance. *IEEE Transactions on Information Theory*, 24:106–110, 1978.

[Pol78] J. M. Pollard. Monte Carlo Methods for Index Computation mod *p*. *Mathematics of Computation*, 32(143):918–924, 1978.

[Pom84] Carl Pomerance. The Quadratic Sieve Factoring Algorithm. In *EUROCRYPT*, pages 169–182, 1984.

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

[Pom96] Carl Pomerance. A Tale of Two Sieves. *Notices of the American Mathematical Society*, 43:1473–1485, 1996.

[Pom08] Carl Pomerance. Elementary thoughts on discrete logarithms. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, MSRI Publications*, 44:385–396, 2008.

[Sch08] Oliver Schirokauer. The impact of the number field sieve on the discrete logarithm problem in finite fields. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, MSRI Publications*, 44:397–420, 2008.

[Ste08] Peter Stevenhagen. The Number Field Sieve. *Algorithmic Number Theory: Lattices, Number Fields, Curves and Cryptography, MSRI Publications*, 44:83–100, 2008.