28 Субэкспоненциальные алгоритмы факторизации и логарифмирования

28.1 Метод Диксона

Для факторизации n Диксон предложил искать решение $(x, y) \in \mathbb{Z}_n \times \mathbb{Z}_n$ сравнения

$$x^2 \equiv y^2 \pmod{n}.$$

Если $x \not\equiv \pm y \pmod{n}$, то n делит $(x-y)(x+y)$, но не делит ни $(x-y)$, ни $(x+y)$. Таким образом, $d = (x-y, n)$ является собственным (нетривиальным) делителем n.

Теорема 28.1. Пусть n — нечетное составное, не являющееся степенью простого числа. Если x, y — случайные числа из $\mathbb{Z}_n^*$ такие, что

$$(x, n) = 1, \quad (y, n) = 1, \quad x^2 \equiv y^2 \pmod{n},$$

то $x \not\equiv \pm y \pmod{n}$ с вероятностью $\geqslant 1/2$.

Доказательство. Пусть $n = p_1^{e_1} \dots p_k^{e_k}$, p_i — нечетные простые, $k \geqslant 2$. Обозначим $z = xy^{-1} \bmod n$. Тогда $z^2 \equiv 1 \pmod{n}$, т. е. z является решением китайской системы

$$\begin{cases} z \equiv \pm 1 \pmod{p_1^{e_1}}, \\ \dots\dots\dots\dots\dots \\ z \equiv \pm 1 \pmod{p_k^{e_k}}. \end{cases}$$

Если $x \equiv \pm y \pmod{n}$, то $z \equiv \pm 1 \pmod{n}$, т. е. либо $z \equiv 1 \pmod{p_i^{e_i}}$, либо $z \equiv -1 \pmod{p_i^{e_i}}$ одновременно для всех $i = 1, \dots, k$. Таким образом, имеется 2^k решений китайской системы и только два из них являются неподходящими. Поэтому $\mathbf{P}\{x \not\equiv \pm y \pmod{n}\} = \mathbf{P}\{z \not\equiv \pm 1 \pmod{n}\} = (2^k - 2)/2^k \geqslant 1/2$. □

Для нахождения пары (x, y) поступают следующим образом:

1. Выбирают факторную базу $S_B = \{p_1, \dots, p_s\}$.
2. Находят T пар целых чисел (a_t, b_t), таких, что $a_t^2 \equiv b_t \pmod{n}$ и число $|b_t|$ является B-гладким, т. е.
$$b_t = (-1)^{\alpha_{t0}} p_1^{\alpha_{t1}} \dots p_s^{\alpha_{ts}}.$$
3. Составляют $T \times (s+1)$ матрицу $A = (\alpha_{ti})$. Если сумма некоторых строк A является вектором с четными координатами $2(\beta_0, \beta_1, \dots, \beta_s)$ и $\mathcal{T}$ – множество номеров таких строк, то искомое сравнение найдено:
$$\left(\prod_{t \in \mathcal{T}} a_t\right)^2 \equiv \left(\prod_{i=1}^{s} p_i^{\beta_i}\right)^2 \pmod{n},$$
т. е. $x^2 \equiv y^2 \pmod{n}$ при $x = \prod_{t \in \mathcal{T}} a_t$ и $y = \prod_{i=1}^{s} p_i^{\beta_i}$.
4. Если найденное сравнение не позволяет факторизовать n, то находят новые пары (a_t, b_t) и/или новое множество $\mathcal{T}$ и повторяют вычисления.

Обсудим реализацию этапов метода Диксона:

3. От матрицы A переходят к матрице $\tilde{A} = (a_{ti} \bmod 2)$ над полем $\mathbb{F}_2$. Множество $\mathcal{T}$ — это множество номеров строк $\tilde{A}$, сумма которых является нулевой строкой.

   Для определения $\mathcal{T}$ достаточно найти ненулевое решение $v \in \mathbb{F}_2^T$ однородной системы линейных уравнений
   $$v\tilde{A} = \mathbf{0}.$$
   Ясно, что искомое решение будет наверняка существовать при $T > s + 1$.

2. Для нахождения пар (a_t, b_t) поступают следующим образом: наудачу выбирают a, находят $b = a^2 \bmod n$ и пробными делениями пытаются разложить b или $n - b$ на простые из факторной базы.

1. Пусть $L(n) = \exp(\sqrt{\ln n \cdot \ln\ln n})$. Доказано, что оптимальным в алгоритме Диксона является выбор $B = L(n)^{1/2}$. При таком выборе сложность алгоритма составляет $O(L(n)^2)$ операций.

28.2 Квадратичное решето

Основную вычислительную трудность в алгоритме Диксона составляет отыскание подходящих пар (a_t, b_t) на этапе 2. Вместо выбора случайных пар (a, b) естественно использовать некоторые специальные пары целых, в которых $|b|$ окажется B-гладким с более высокой вероятностью.

Данную идею реализуют алгоритмы *квадратичного решета* (Quadratic Sieve, QS) и решета в числовом поле (Number Field Sieve, NFS) — самые эффективные на сегодняшний день алгоритмы, с помощью которых удалось факторизовать некоторые числа `RSA-`k.

Рассмотрим алгоритм квадратичного решета. Данный алгоритм отличается от алгоритма Диксона двумя оптимизациями:

1. Генерация специальных пар (a, b).

   Пусть
   $$Q(x) = (x + m)^2 - n, \quad m = \lfloor \sqrt{n} \rfloor, \quad x \in \mathbb{Z}.$$
   Если $a = x + m$ и $b = Q(x)$, то мы получаем нужное сравнение
   $$a^2 \equiv b \pmod{n}.$$
   При этом для малых $|x|$ значение $|b| = |Q(x)|$ будет невелико.

   Таким образом, если вместо случайных пар (a, b) рассмотреть пары $(x + m, Q(x))$, $x \in \{-M, \ldots, M\}$, то вероятность порождения B-гладких чисел $b = Q(x)$ возрастает.

2. Просеивание (ускорение проверки B-гладкости).

   При просеивании используется массив R, который индексируется числами $x \in \{-M, \ldots, M\}$.

   Просеивание осуществляется следующим образом:

   1) массив R заполняется значениями $R[x] \leftarrow \log |Q(x)|$;
   2) для всякой степени $p^e \leqslant C$ простого $p \in S_B$ находят решение $x^* \in \{0, 1, \ldots, p^e - 1\}$ сравнения $Q(x^*) \equiv 0 \pmod{p^e}$;
   3) если $x \equiv x^* \pmod{p^e}$, то
      $$Q(x) = (x + m)^2 - n \equiv (x^* + m)^2 - n \equiv 0 \pmod{p^e},$$
      т. е. $Q(x)$ делится на p^e.

      Учтем данный факт следующим образом: $R[x] \leftarrow R[x] - \log p$ для всех x таких, что $x \equiv x^* \pmod{p^e}$;
   4) если после обработки степеней всех чисел из базы множителей (*просеивания*) $R[x]$ равняется 0, то число $|Q(x)|$ является B-гладким. Действительно, равенство $R[x] = 0$ означает, что
      $$\log |Q(x)| = \sum_{i=1}^{s} \alpha_i \log p_i$$
      или
      $$|Q(x)| = \prod_{i=1}^{s} p_i^{\alpha_i}.$$

При практической реализации метода квадратичного решета значения логарифмов вычисляются приближенно. По окончании просеивания для всех x таких, что $R[x] \approx 0$ проверяется B-гладкость числа $Q(x)$.

Пример 28.1 (факторизация `RSA-155`, 1999). Факторизация числа `RSA-155` проведена с помощью метода NFS, основанного на похожих идеях. Для факторизации потребовалось использовать:

- 160 рабочих станций `SGI` и `Sun` (175–400 МГц);
- 8 процессоров `SGI Origin 2000` (250 МГц);
- 120 персональных компьютеров `Pentium II` (300–450 МГц);

– 4 кластера `Digital/Compaq` (500 МГц);

– 9 недель для настройки параметров GNFS;

– 5.2 месяца для сбора пар (a_t, b_t);

– 3.2 Гб оперативной памяти, 224 часа вычислений на `Cray C916` для обработки матрицы A размера 6699191×6711336 с 417132631 ненулевыми элементами. □

28.3 Индекс-метод

Пусть дискретное логарифмирование производится в циклической группе $G = \langle g \rangle$ простого порядка q. Пусть G является подгруппой $\mathbb{F}_p^*$, p – простое число. Тогда для определения $\log_g y$ можно использовать индекс-метод.

Этапы индекс-метода имеют следующий вид:

1. Выбирают факторную базу $S_B = \{p_1, \ldots, p_s\}$.

2. Находят T пар натуральных чисел (a_t, b_t) таких, что $1 \leqslant a_t < q$, а число $b_t = g^{a_t} \bmod p$ является B-гладким, т. е.
$$b_t = p_1^{\alpha_{t1}} \ldots p_s^{\alpha_{ts}}$$
или
$$\alpha_{t1} \log_g p_1 + \ldots + \alpha_{ts} \log_g p_s \equiv a_t \pmod q, \quad t = 1, \ldots, T. \qquad (\star)$$

3. Система линейных (над $\mathbb{F}_q$) уравнений $(\star)$ решается относительно неизвестных $\log_g p_i$, $i = 1, \ldots, s$. На шаге 2 находят такое количество пар T, что решение единственно.

4. Находят пару натуральных чисел (a', b'), таких, что $1 \leqslant a' < q$, $b' = yg^{a'} \bmod p$ является B-гладким, т. е.
$$b' = p_1^{\beta_1} \ldots p_s^{\beta_s}$$
или
$$\beta_1 \log_g p_1 + \ldots + \beta_s \log_g p_s \equiv a' + \log_g y \pmod q.$$

5. Искомый логарифм
$$x = (\beta_1 \log_g p_1 + \ldots + \beta_s \log_g p_s - a') \bmod q.$$

Нахождение дискретного логарифма по индекс-методу выполняется за время

$$\exp((c + o(1))\sqrt{\ln p \ln \ln p})$$

где c — положительная константа. Наиболее эффективная на сегодняшний день модификация метода — метод решета числового поля — позволяет выполнять логарифмирование за время

$$\exp((c + o(1))\sqrt[3]{\ln p (\ln \ln p)^2}), \quad c = 1.923.$$

Пример 28.2 (`СТБ 1176.2`)**.** Простые p и $q \mid p-1$ в `СТБ 1176.2-99` имеют битовые длины l и r соответственно. Пара (l, r) может принимать одно из 10 значений: от $(143, 638)$ до $(257, 2462)$.

Длины p и q выбираются так, что
$$\sqrt{q} \approx \exp(1.923\sqrt[3]{\ln p (\ln \ln p)^2}).$$

Такой выбор означает, что сложность логарифмирования с помощью экспоненциальных методов (ρ-метод, λ-метод) сравнима со сложностью логарифмирования субэкспоненциальными методами. Отметим, что поддержание паритета между методами логарифмирования приводит к значительному росту длины p, что чувствительно сказывается на быстродействии алгоритмов выработки и проверки подписи. □