

28 Факторизация

28.1 Задача факторизации

Мы установили, что одним из методов решения задачи RSA является факторизация модуля n с последующим определением $\varphi(n)$ и $d = e^{-1} \bmod \varphi(n)$.

Напомним, что проблема факторизации ставится следующим образом.

Задача 28.1 (Factor). *Индивидуальная задача:* n — составное. *Ответ:* $d: d \mid n, d \neq 1, n$.

Пример 28.1 (RSA- k). На сайте <http://www.rsa.com> размещены *вызовы* — задачи факторизации чисел RSA- k . Каждое такое число n состоит из k десятичных разрядов и является произведением двух простых p и q , которые

- сравнимы с 2 по модулю 3, поэтому n можно использовать для шифрования по RSA с открытой экспонентой $e = 3$;
- генерировались с использованием теста Рабина — Миллера на простоту;
- уничтожались сразу после вычисления $n = pq$.

На сегодняшний день даны ответы на следующие вызовы (MIPS — миллион инструкций в секунду):

Число	Дата	Сложность, MIPS-лет	Алгоритм
RSA-100	апрель 1991	7	QS
RSA-110	апрель 1992	75	QS
RSA-120	июнь 1993	830	QS
RSA-129	апрель 1994	5000	QS
RSA-130	апрель 1996	500	GNFS
RSA-140	февраль 1999	2000	GNFS
RSA-155	август 1999	8000	GNFS
RSA-200 (663 бита)	май 2005	≈ 75 лет работы 2.2Ghz Opteron	GNFS
RSA-768 (двоичные разряды)	декабрь 2009	≈ 2000 лет работы 2.2Ghz Opteron	GNFS
RSA-240 (795 битов)	декабрь 2019	≈ 4000 лет работы 2.1Ghz Intel Xeon Gold 6130	GNFS
RSA-250 (829 битов)	февраль 2020	≈ 2700 лет работы 2.1Ghz Intel Xeon Gold 6130	GNFS

Перейдем к рассмотрению алгоритмов факторизации. Отметим, что алгоритм пробного деления n на числа от 2 до $\lfloor \sqrt{n} \rfloor$ выполняется за экспоненциальное время $O(\sqrt{n} \log^2 n)$. Мы рассмотрим два изящных алгоритма, которые выполняются за меньшее время.

28.2 Алгоритм $p - 1$

В 1974 году Поллард предложил алгоритм факторизации, который может выполняться за малое время, если для некоторого простого делителя p числа n простые делители числа $p - 1$ невелики, точнее являются B -гладкими для небольшого B .

Определение 28.1. Число m является B -гладким, если все простые делители m не превосходят $B \in \mathbb{N}$. Множество S_B , составленное из всех простых чисел, не превосходящих B , называется *факторной базой*. \square

АЛГОРИТМ $p - 1$

Вход: n — нечетное составное.

Выход: d — нетривиальный делитель n или \perp — делитель не найден.

Шаги:

1. Выбрать натуральное B .
2. $a \leftarrow 2$.
3. Для $q \in S_B$ выполнить:

а) $e_q \leftarrow \lfloor \log_q n \rfloor$ (e_q есть максимальное целое такое, что $q^{e_q} \leq n$);

$$б) a \leftarrow a^{q^{e_q}} \bmod n.$$

$$4. d \leftarrow (a - 1, n).$$

5. Если $d \notin \{1, n\}$, то вернуть d . Иначе — \perp .

Идея. Обозначим $Q = \prod_{q \in S_B} q^{e_q}$ и пусть $d = (2^Q - 1, n)$.

Предположим, что p — простой делитель n такой, что число $p - 1$ является B -гладким. Тогда:

- 1) $(p - 1) \mid Q$ (если $q^e \mid p - 1$, то $q^e \mid q^{e_q} \mid Q$);
- 2) по малой теореме Ферма $2^Q \equiv 2^{p-1} \equiv 1 \pmod{p}$ и $p \mid (2^Q - 1)$;
- 3) $p \mid n$.

Из 2) и 3) следует, что $p \mid d = (2^Q - 1, n)$. Поэтому если $d \neq n$, то мы получаем нетривиальный делитель n .

Сложность алгоритма: $|S_B| = O(B/\log B)$ (теорема о распределении простых) и требуется выполнить $O(B/\log B)$ возведений в степени q^{e_q} по модулю n или $O(B \log^3 n)$ операций.

Упражнение 28.1. Для метода $p-1$ проанализировать случаи, когда d не является собственным делителем n . Доказать:

- 1) если $d = 1$, то $p - 1$ не является B -гладким ни для одного простого делителя p числа n ;
- 2) если $d = n$, то $p - 1$ является B -гладким для всякого простого делителя p числа n . □

Пример 28.2. Рассмотрим число $n = 1846202297 = pq$, где $p = 37951$, $q = 48647$. Имеем: $p - 1 = 2 \cdot 3 \cdot 5^2 \cdot 11 \cdot 23$, $q - 1 = 2 \cdot 13 \cdot 1871$. Поэтому $d = 1$ при $B < 11$, $d = p$ при $23 \leq B < 1871$ и $d = n$ при $B \geq 1871$. □

28.3 ρ -метод

В 1975 году Поллард предложил еще один алгоритм факторизации — ρ -метод.

Идея алгоритма. Проведем следующие построения:

1. Пусть наудачу выбрано преобразование $\varphi: \mathbb{Z}_n \rightarrow \mathbb{Z}_n$, выбран произвольный элемент $x_0 \in \mathbb{Z}_n$ и определена последовательность

$$x_t = \varphi(x_{t-1}), \quad t = 1, 2, \dots$$

Можно сказать, что «частицы» x_t размещаются случайно независимо равновероятно по n «ячейкам» — элементам \mathbb{Z}_n . Согласно парадоксу «дней рождения» (см. § 26), коллизии начнут происходить при $t = O(\sqrt{n})$ в среднем.

2. Пусть p — минимальный (неизвестный) простой делитель n . Данному p поставим в соответствие (ненаблюдаемую) последовательность

$$y_t = x_t \bmod p, \quad t = 0, 1, 2, \dots$$

Теперь «частицы» y_t размещаются случайно независимо равновероятно по p «ячейкам» — элементам множества $\{0, 1, \dots, p-1\}$. Поэтому коллизии начнут происходить при $t = O(\sqrt{p})$ в среднем. Коллизия

$$y_t = y_\tau, \quad t < \tau,$$

означает, что траектория последовательности y_0, \dots, y_τ принимает форму буквы ρ , что и обуславливает название метода факторизации.

Поскольку $p \leq \sqrt{n}$, то с высокой вероятностью $x_t \neq x_\tau$.

3. Совпадение $y_t = y_\tau$ означает, что

$$p \mid (y_\tau - y_t) \Rightarrow p \mid (x_\tau - x_t) \Rightarrow p \mid (x_\tau - x_t, n).$$

Таким образом, если $x_t \neq x_\tau$, то $(x_\tau - x_t, n)$ — нетривиальный делитель n .

Мы свели задачу поиска делителя числа n к задаче поиска коллизии в (ненаблюдаемой!) последовательности (y_t) . Для поиска коллизии можно воспользоваться алгоритмом Брента в котором прямая проверка $y_t \stackrel{?}{=} y_\tau$ заменяется на косвенную проверку $(x_t - x_\tau, n) \stackrel{?}{>} 1$.

АЛГОРИТМ ρ -МЕТОД

Вход: $n \in \mathbb{N}$.

Параметры: φ .

Выход: d — нетривиальный делитель n или \perp — делитель не найден.

Шаги:

1. $A \xleftarrow{R} \mathbb{Z}_n$.
 2. Для $i = 0, 1, \dots$ выполнить:
 - 1) $B \leftarrow \varphi(A)$;
 - 2) для $r = 1, \dots, 2^i$:
 - а) $d \leftarrow (A - B, n)$;
 - б) если $d = n$, то вернуть \perp (коллизия в (x_t));
 - в) если $d > 1$, то вернуть d ;
 - г) $B \leftarrow \varphi(B)$.
 - 3) $A \leftarrow B$.
-

Сложность алгоритма. Для определения нетривиального делителя требуется вычислить значения φ порядка $O(\sqrt{p})$ раз, где p — наименьший простой делитель n . Остается сказать, что на практике выбирают простую функцию φ , которая, впрочем ведет себя как случайная функция (напр., $\varphi(x) = (x^2 + 1) \bmod n$). В этом случае алгоритм работает за время $O(\sqrt{p} \log^2 n)$.

28.4 Выбор модуля RSA

Сложность ρ -метода и алгоритма $p - 1$ — $O(\sqrt{p} \log^2 n)$ и $O(B \log^3 n)$ операций соответственно. Здесь p — наименьший простой делитель n , B — наименьший по простым $p \mid n$ из наибольших простых делителей числа $p - 1$ (факторная база S_B должна содержать все простые делители $p - 1$). Существует «близнец» алгоритма $p - 1$ — алгоритм $p + 1$, сложность которого определяется наибольшим простым делителем не числа $p - 1$, а числа $p + 1$.

При выборе простых делителей p и q модуля RSA $n = pq$ руководствуются следующими правилами:

- 1) $\log p \approx \log q$ (усложняется факторизация n по ρ -методу);
- 2) число $|p - q|$ должно быть большим (в противном случае для факторизации n можно применить метод Ферма);
- 3) числа p и q должны быть сильно простыми, p — *сильно простое*, если:
 - а) $p - 1$ имеет большой простой делитель r (усложняется факторизация n по методу $p - 1$);
 - б) $p + 1$ имеет большой простой делитель (усложняется факторизация по методу $p + 1$);
 - в) $r - 1$ имеет большой простой делитель (защита от циклических атак).

Такой выбор модуля n приводит к тому, что факторизация с помощью рассмотренных алгоритмов выполняется за субэкспоненциальное время (см. далее).