Московский Физико-Технический Институт (ГУ МФТИ)

Кафедра радиотехники

20 мая 2005 г.

Эссе по курсу «Защита информации»

Алгоритмы факторизации целых чисел (с экспоненциальной сложностью)

Выполнил:
Точ Дмитрий
116 гр.

http://www.re.mipt.ru/infsec

Введение

Сложностью задачи факторизации (разложение на множители) определяется сложность нахождения секретного ключа некоторых шифрсистем (например RSA). Поэтому при проектировании таких систем необходимо учитывать это обстоятельство. И дело здесь не только в производственной мощности современных компьютеров. В последнее время появляется все больше и больше алгоритмов факторизации, позволяющие факторизовать большие числа за короткий (по сравнению с другими) срок времени. Эта область в теории чисел активно развивается и представяет большой интерес для криптоаналитиков.

Прежде чем непосредственно перейти к рассмотрению алгоритмов, формализуем нашу задачу следующим образом: *требуется найти простые множители p и q числа n, т.е. такие целые числа, что n=p*q. (Следует сразу оговорить ситуацию, когда число множителей более 2-ух: в этом случае один из них находится довольно быстро и сложность задачи резко падает)*
В дальнейшем будем считать, что *n* не является простым (это можно проверить каким нибудь алгоритмом проверки на простоту), и раскладывается в произведение 2-ух простых чисел.
Также нужно отметить, что либо $p \le \sqrt{n}$, либо $q \le \sqrt{n}$ (иначе их произведение превосходило бы *n*), а, найдя хотя бы один из множителей, мы с легкостью получим и второй.

В этой работе будут рассмотрены (без доказательств) следующие алгоритмы с эспоненциальной оценкой сложности:

- «Пробных делений»
- Ферма (P. de Fermat)
- ρ-метод Полларда (Pollard J. M)
- Шермана-Лемана (Lehman R. S.)
- Ленстры (Lenstra H. W.)
- Полларда-Штрассена (Strassen V.)
- (P-1)-метод Полларда

Алгоритм «пробных делений».

Самый тривиальный и очевидный. Но даже он имеет несколько вариантов реализации:

1) Перебираем последовательно <u>все простые</u> числа, не превосходящие $\lfloor\sqrt{n}\rfloor$, пока не доберемся до делителя числа *n*.
2) Перебираем <u>все целые</u> числа меньше $\lfloor\sqrt{n}\rfloor$, пока не доберемся до делителя *n*.

У каждой из этих реализаций есть свои преимущества и недостатки по сравнению с другой:

1) необходимо строить таблицу простых чисел
2) требуется больше операций деления

Соответствующие сложности этого алгоритма для данных реализаций:

1) $O\left(\sqrt{n}\log n\right)$
2) $O\left(\sqrt{n}\log^2 n\right)$

Данный алгоритм совсем не пригоден для больших *n*. Его имеет смысл применять только в упрощенном варианте, когда требуется найти делитель, не превосходящий какого-то заданного значения.

Алгоритм Ферма.

Этот алгоритм был впервые предложен французским математиком Пьером Ферма в 1643г. Он позволяет находить наибольший делитель *n* (в отличие от алгоритма пробных делений, который лучше подходит для поиска наименьшего делителя). Поэтому этот алгоритм очень хорош в случае, когда множители числа *n* примерно одинаковы по величине.
Пусть *p* и *q* удовлетворяют условию $1 < p \le q$. Представим их в виде $p = u - v$, $q = u + v$, где числа *u* и *v* – натуральные. Т.к. *p* и *q* – нечетные (иначе один из сомножителей был бы уже известен), то такое представление возможно: $u = \frac{p+q}{2}$, $v = \frac{p-q}{2}$. Алгоритм основан на представлении числа *n* в виде $n = u^2 - v^2$, откуда нетрудно заметить $n = (u-v)(u+v) = pq$.
Введем последовательность $\{x_k, y_k\}, k = 0,1,2,3...$, введем переменную $r = x_k^2 - y_k^2 - n$.
Теперь вступает в силу следующий алгоритм:

1. $(x_0, y_0) = (\lfloor\sqrt{n}\rfloor, 0)$
2. Если $r = 0$, то $n = x_k^2 - y_k^2 = (x_k - y_k)(x_k + y_k)$. Алгоритм останавливается, результатом будут числа $p = x_k - y_k$ и $p = x_k + y_k$.
3. Иначе
   a. Если $r > 0$, то $(x_{k+1}, y_{k+1}) = (x_k, y_k + 1)$.
   b. Если $r < 0$, то $(x_{k+1}, y_{k+1}) = (x_k + 1, y_k)$.
4. $r = x_{k+1}^2 - y_{k+1}^2 - n$ Go to 2.

Сложность алгоритма Ферма $O(C \log^2 n)$, где *C* - некоторый параметр, определяемый практически из конкретных вычислительных возможностей.

ρ - Метод Полларда.

С помощью этого метода было впервые факторизовано число Ферма $F_8 = 2^{2^8} + 1$.
Впервые этот алгоритм был предложен Дж. Поллардом в 1975 г. Суть его заключается в следующем:

1. Выбираем отображение $f : Z_n^* \to Z_n^*$, где Z_n^* - кольцо вычетов по модулю *n*. В качестве функции $f(x)$ обычно берут полиномы степени ≥2 (например $f(x) = x^2 + 1$).
2. Выбираем некоторое число $x_0 \in Z_n^*$ и строим рекурсивную последовательность $\{x_k\}, k = 0,1,2,...$ по следующему принципу $x_{k+1} = f(x_k) \bmod n$.
3. Для некоторых номеров i и j вычисляем $d = \gcd(x_i - x_j, n)$. Если $d = 1$ или $d = n$, то рассматриваем другую пару (i, j). Если $1 < d < n$, то *d* – делитель числа *n*.

Как видим, алгоритм может оказаться довольно громоздким из-за большого количества пар (i, j). Эта проблема решаема с помощью некоторых способов выбора номеров i и j.
Рассмотрим некоторые из них:

1. $i = 2j$, т.е. $d = \gcd(x_{2j} - x_j, n)$
2. Если *j* удовлетворяет условию $2^h \le j < 2^{h+1}, h \in N$, то берут $i = 2^h - 1$

Если окажется, что алгоритм не нашел делителя, то можно попробовать другие функции *f*, например $f = x^2 + c$, где *c* – некоторая константа.

Сложность этого алгоритма оценивается, как $O\left(n^{1/4}\log^3 n\right)$ битовых операций.

Существует теорема, которая формулируется так: Пусть n – составное число, $\exists C : \forall\lambda > 0$ вероятность того, что метод Полларда не сможет найти нетривиальный делитель n за время $C\lambda^{1/2}n^{1/4}\log^3 n$ не превышает величину $e^{-\lambda}$.
ρ - Метод Полларда обычно используется для нахождения небольших делителей числа n.

Алгоритм Шермана-Лемана.

Предположим, что n – нечетно и $n > 8$.

1. Проверяем, являются ли числа $2,3,\ldots,\left\lfloor\sqrt[3]{n}\right\rfloor$ делителями числа n. Если получен отрицательный результат, идем дальше.
2. Т.к. делителя не найдено, то $n^{1/3} < p \le q < n^{2/3}$. Для всех $k = 1,2,\ldots,\left\lfloor\sqrt[3]{n}\right\rfloor$ и $d = 0,1,\ldots,\left[\dfrac{n^{1/6}}{4\sqrt{k}}\right]+1$ проверяем, является ли число $\left(\left[\sqrt{4nk}\right]+d\right)^2 - 4nk$ квадратом натурального числа. Если да, то числа $A = \left\lfloor\sqrt{4nk}\right\rfloor + d$ и $B = \sqrt{A^2 - 4nk}$ удовлетворяют условию $A^2 \equiv B^2 \bmod n$.
   Теперь проверяем условие $1 < \gcd(A \pm B, n) < n$, и если оно выполнено, то алгоритм завершается.

Сложность этого алгоритма составляет $O\left(n^{1/3}\right)$ арифметических операций. Этот алгоритм позволяет эффективно использовать параллельные вычисления.

Алгоритм Ленстры.

Приведем без доказательства следующую теорему: Пусть r, s, и n – натуральные числа такие, что $1 \le r < s < n$, $n^{1/3} < s$ и $\gcd(r,s) = 1$. Тогда найдется не более 11 чисел r_i - делителей n, и таких, что $r_i \equiv r \bmod s$, и имеется алгоритм, который находит все эти делители за $O(\log n)$ арифметических операций.

Также можно доказать, что $\forall\alpha : \dfrac{1}{3} > \alpha > \dfrac{1}{4}\ \ \exists C(\alpha) > 0$: при условии $1 \le r < s < n$, $\gcd(r,s) = 1$ и $s > n^{\alpha}$ существует не более $C(\alpha)$ положительных делителей n, сравнимых с r по модулю s.
Перейдем теперь непосредственно к самому алгоритму. Полагаем, что на входе у нас даны числа r, s, и n, удовлетворяющие условию теоремы.

1. Находим $r^* \in \mathrm{N}$ такой, что $r^* r \equiv 1 \bmod s$, и r' такой, что $r' = r^* n \bmod s, 0 \le r' < s$
2. Вводим последовательность $\{(a_i, b_i, c_i)\}, i = 0,1,2,\ldots$, удовлетворяющую следующим условиям:
   $a_0 = s, b_0 = c_0 = 0$,
   $a_1 \equiv r' r^* \bmod s,\ 0 < a_1 \le s,\ b_1 = s,\ c_1 = \dfrac{n - rr'}{s} \cdot r^* \bmod s$
   и при $i \ge 2$
   $a_i = a_{i-2} - q_i a_{i-1},\ b_i = b_{i-2} - q_i b_{i-1},\ c_i \equiv c_{i-2} - q_i c_{i-1} \bmod s$,

где числа q_i определяются однозначно из условий $\begin{cases} 0 \le a_i < a_{i-1}, & i = 2k \\ 0 < a_i \le a_{i-1}, & i = 2k+1 \end{cases}$

Т.е. фактически q_i - частное от деления a_{i-2} на a_{i-1} за исключением случая, когда *i* – нечетно и остаток от деления равен 0.

3. Для очередного *i* найти все целые *c*, удовлетворяющие следующим условиям: $c \equiv c_i \bmod s$ и $\begin{cases} |c| < s, & i = 2k \\ 2a_i b_i \le c \le \dfrac{n}{s^2} + a_i b_i, & i = 2k+1 \end{cases}$. Таких *c* будет не более 2-ух.
4. Для каждого *c* решаем в целых числах следующую систему уравнений
$\begin{cases} xa_i + yb_i = c \\ (xs + r)(ys + r') = n \end{cases}$
Если $x \ge 0,\ y \ge 0$, то $xs + r$ - искомый делитель.
5. Если $a_i = 0$, то алгоритм закончен. Иначе Go to 2.

Оценка сложности алгоритма Ленстры $O\left(n^{1/3} \log n\right)$.

Алгоритм Полларда-Штрассена.

Этот алгоритм основан на следующей теореме: Пусть $z \in \mathrm{N}$, $y = z^2$. Тогда $\forall t \in \mathrm{N}$ наименьший простой делитель числа $\gcd(t, y!)$ может быть найден за $O(z \log^2 z \log^2 t)$ двоичных операций. Опуская доказательство, перейдем к способу, с помощью которого можно найти наименьший простой делитель числа $\gcd(t, y!)$.

Положим $f(j) = ((j-1)z + 1) \cdot \ldots \cdot ((j-1)z + z),\ j = 1,2,\ldots,z$.

1. Находим $f(1), f(2), \ldots, f(z)$
2. Вычисляем $\gcd(t, f(j)),\ j = 1,2,\ldots,z$ до получения первого нетривиального делителя.
3. Осуществляя пробные деления $\gcd(t, f(j))$ на числа $(j-1)z + 1$, …, $(j-1)z + z$, находим наименьший простой делитель числа $\gcd(t, y!)$.

Для факторизации числа *n* берут $z = \left[n^{1/4}\right] + 1,\ y = z^2 > n^{1/2},\ t = n$. Находят наименьший простой делитель числа $\gcd(n, y!)$. Т.к. $p \le n^{1/2} < y$, то $y!$ делится на наименьший простой делитель *p* числа *n*. Именно число *p* выдаст алгоритм.

Сложность рассмотренного алгоритма $O\left(n^{1/4} \log^4 n\right)$.

(P-1)-метод Полларда.

Прежде чем начать рассмотрение этого алгоритма введем определение: будем говорить, что число *k* является *B*-степенно-гладким для некоторого *B*>0, если $\forall m$: *m* – простое и является делителем числа *k*, выполнено условие $m^{\upsilon_m(k)} \le B$, где $\upsilon_m(k) \in \{0,1,2,\ldots\}$- наибольшее число такое, что $m^{\upsilon_m(k)}$ делит *k*.

Теперь непосредственно сам алгоритм:

1. Исходя из возможностей нашей вычислительной машины, выбираем границу гладкости *B*. Обычно $B \sim 10^5 - 10^6$.

2. Выбираем произвольное целое число a, удовлетворяющее условию $2 \le a \le n-1$, и вычисляем $d = \gcd(a,n)$. Если $1 < d < n$, то d – искомый делитель.
3. Строим таблицу всех простых чисел $q_1 < q_2 < ... < q_k \le B$ и для каждого такого числа q_i находим $l_i = \left[\frac{\log B}{\log q_i} \right]$, т.е. $q_i^{l_i} \le B,\ q_i^{l_i+1} > B$.
4. Вычисляем $P = \prod_{i=1}^{k} a^{q_i^{l_i}} \bmod n$
5. Находим $d = \gcd(P-1,n)$. Если $1 < d < n$, то d – искомый делитель, иначе алгоритм не смог найти делитель. В этом случае можно взять другое основание a или границу гладкости.

Оценка сложности этого метода в худшем случае составляет $O\left(n^{1/2} \log^c n\right)$ арифметических операций. Однако в некоторых случаях этот алгоритм может довольно быстро найти делитель n.

Другие алгоритмы. Заключение.

Для некоторых чисел n существуют специальные алгоритмы факторизации. Приведем без доказательства следующую теорему: Пусть $b, k \in \mathrm{N}, b > 1, n = b^k - 1$. Если p – простое и делит n, то выполнено одно из 2-ух следующих утверждений:

1. p является делителем $b^d - 1$ при некотором $d<k$, делящем k.
2. $p \equiv 1 \bmod k$. Причем если $p>2$ и k – нечетно, то $p \equiv 1 \bmod 2k$.

Посмотрим, как можно воспользоваться этой теоремой, на следующем примере: $n = 2^{11} - 1$. Первое утверждение не выполняется, значит $p \equiv 1 \bmod 22$, т.е. $p = 23$.

Итак, мы рассмотрели основные алгоритмы факторизации наиболее известные на данный момент. В работу не вошли описания некоторых алгоритмов, которые требуют применения дополнительных теоретических основ. Например, методы Шэнкса (Shanks D.), использующие бинарные квадратичные нормы и имеющие наилучшие из всех рассмотренных выше алгоритмов оценки сложности: $O\left(n^{1/5+\varepsilon}\right)$ для алгоритма, работающего с положительно определенными бинарными квадратичными формами отрицательного дискриминанта, и $O\left(n^{1/4+\varepsilon}\right)$ - положительного (этот метод носит название SQUFOF). Метод SQUFOF работает с числами, не превосходящими $2\sqrt{n}$, и среди алгоритмов с экспоненциальной сложностью считается одним из самых лучших. Есть метод аналогичный (P-1)-методу Полларда, но использующий разложение числа P+1 (с помощью последовательностей чисел Люка (Lucas F. A.)). Он носит название (P+1)-метод Уильямса (Williams H. C.). Но на практике этот алгоритм оказывается довольно медленным, поэтому используется редко.

Также можно упомянуть алгоритм Ривеста-Пинтера (Rivest R. L.) со сложностью $O\left(n^{1/3}\right)$ и метод Лемера-Пауэрса (Lehmer D. H.) с эвристической оценкой сложности $O\left(n^{1/4}\right)$, использующий разложение $\sqrt{n}$ в непрерывную дробь.

Из описанных выше алгоритмов наибольшей популярностью пользуются ρ-метод Полларда, алгоритм Полларда-Штрассена и (P-1)-метод Полларда. Применяются они, как правило, в сочетании с алгоритмами с субэкспоненциальной оценкой сложности (оценка сложности задается формулой $e^{(c+o(1))(\log n)^{\gamma}(\log\log n)^{1-\gamma}}$, где $0 < \gamma < 1$, $c = const$, $c > 0$).
Таким образом процесс факторизации распадается на 3 этапа:

1. Перебор первых 1-2 тыс. простых чисел.
2. С помощью какого-нибудь алгоритма с экспоненциальной сложностью ищут небольшие простые делители.
3. С помощью какого-нибудь алгоритма с субэкспоненциальной сложностью ищут большие простые делители.

Иногда перед вторым этапом можно сделать проверку на простоту с помощью какого-нибудь специального алгоритма.
Из алгоритмов, имеющих субэкспоненциальную оценку сложности, наиболее популярны метод Диксона (Dixon J. D.), алгоритм Бриллхарта-Моррисона (Brillhart J., Morrison M. A.), методы Шнорра-Ленстры (Schnorr C. P.) и Ленстры-Померанса (Pomerance C.), квадратичное решето и (самые эффективные для факторизации больших чисел) алгоритмы решета числового поля. А также следует упомянуть об алгоритме Ленстры для факторизации чисел с использованием эллиптических кривых.

Литература:

1. О. Н. Василенко «Теоретико-числовые алгоритмы в криптографии» МЦНМО, 2003 (http://www.cryptography.ru:8200/pubd/2003/12/04/0001169580/book.pdf)
2. А. В. Черемушкин «Лекции по арифметическим алгоритмам в криптографии» МЦНМО, 2002 (http://www.cryptography.ru:8200/pubd/2003/02/24/0001169266/cherem.pdf)
3. A. Menezes, P. van Oorschot, S. Vanstone „Handbook of Applied Cryptography” CRC Press, 1996
4. М. И. Анохин, Н. П. Варновский, В. М. Сидельников, В. В. Ященко «Криптография в банковском деле» М.: МИФИ, 1997 (http://www.cryptography.ru/db/msg.html?mid=1169307&uri=node189.html)