УДК 681.3.067

И.А. Кукало, П.А. Миклин, Р.В. Литвинов

Алгоритмы генерации псевдопростых чисел в «Borland C++ 3.1»

Выполнено экспериментальное исследование быстродействия алгоритмов генерации простых чисел методом пробных делений и методами малой теоремы Ферма, Рабина — Миллера и Соловэя — Штрассена, использующими фильтр деления на малые простые числа. Для каждого из методов представлены временные зависимости скорости генерации от числа разрядов, полученные усреднением большого ансамбля частных реализаций.

В конце XX – начале XXI в. мировое сообщество вступило в новую технологическую эпоху — эру информационных технологий. Эта индустрия занимается производством, обработкой, хранением и передачей информации. Он стала неотъемлемой частью мировой хозяйственной системы, вполне самостоятельным и довольно значительным сектором экономики. Зависимость современного общества от информационных технологий настолько высока, что сбои в информационных системах способны привести к значительным общественным и межгосударственным инцидентам.

Передача информации — ключевая отрасль для информационных технологий, ведающая вопросами транспортировки информации. Однако именно при транспортировке информация более всего уязвима к различного рода злоупотреблениям. Действительно, узлы хранения и обработки данных ввиду их компактности можно физически защитить от доступа злоумышленников, чего не скажешь о линиях связи протяженностью многие сотни или тысячи километров, защитить их практически невозможно. Поэтому именно для каналов передачи информации весьма актуальна проблема защиты информации. Важнейшую роль в решении этой проблемы играет криптография. Ее привлекательность состоит в том, что в отличие от других подходов криптографические методы защиты информации основаны на преобразовании самой информации и никак не связаны с характеристиками ее материальных носителей, вследствие чего наиболее универсальны и потенциально дешевы в реализации.

Важным понятием в криптографии является ключ — сменный элемент шифра, который применяется для шифрования конкретного сообщения. При создании ключей в современных ассиметричных криптосистемах и их распределении между пользователями используются большие простые числа, разрядность (биты) которых может достигать нескольких сотен или даже тысяч. Арифметические операции с такими числами на обычных ПК требуют создания специального программного обеспечения [1], в том числе реализации быстрых алгоритмов генерации простых чисел [2]. Огромная разрядность этих чисел, диктуемая современными требованиями безопасности, делает детерминированные алгоритмы малоэффективными. В настоящее время для этой цели используются методы генерации чисел, простоту которых можно обеспечить с вероятностью не менее $1-2^{-128}$.

1. Базовый метод генерации больших простых чисел с использованием фильтра деления на малые простые числа

Этот метод опирается на теорему Евклида о бесконечности множества простых чисел. Алгоритм работы метода следующий: генерируется псевдослучайное число n, затем осуществляется тестирование этого числа на простоту каким-либо из методов [3]. Попытка продолжается до тех пор, пока не найдется простое число либо количество попыток N не станет равным

$$N = K\left(\left[\log_2 n\right]+1\right), \qquad (1)$$

где $K > 1$ — коэффициент запаса; [...] — операция взятия целой части.

Формула (1) получена на основе закона распределения простых чисел, который показывает, что вероятность случайно выбранного числа оказаться простым равна $1/\ln n$.

Алгоритм базового метода может быть представлен следующими шагами:

1) задаём l — нижняя граница диапазона, в котором должно находиться простое число;

2) задаём u — верхняя граница диапазона, в котором должно находиться простое число;

3) проверим корректность задания диапазона $2 < l \le u$;

4) подсчитаем максимальное количество попыток $r \leftarrow 100([\log_2 u]+1)$;

5) $r \leftarrow r - 1$;

6) если $r < 0$, то лимит попыток исчерпан (событие имеет очень малую вероятность; в наших исследованиях ни разу не реализовалось; в случае его реализации, необходимо перезапустить программу);

7) выберем в заданном интервале $u \le n \le l$ случайное число n;

8) если n меньше, чем 1999, то тестирование выполняется методом пробного деления на все известные (табулированные) простые числа, меньшие, чем n. Если делитель существует, то переходим на шаг 5;

9) если n больше, чем 1999, то тестирование выполняется методом пробного деления на все известные (табулированные) простые числа, меньшие, чем 1999. (Этот шаг позволяет исключить 85 % составных чисел [1].) Если делитель существует, то переходим на шаг 5;

10) тестируем n на простоту выбранным методом;

11) если тест отрицателен, переходим на шаг 5;

12) если тест положителен, то сгенерировано простое число в заданном диапазоне.

Этот алгоритм реализован в виде программы, написанной на языке программирования C++ и не зависит явно от особенностей того или иного теста, оформленного в виде подпрограммы, вызываемой на шаге 8.

2. Алгоритмы тестирования на простоту

2.1. Метод пробных делений

Этот тест представляет собой специально написанную на языке программирования C++ функцию. Он основан на пробном последовательном делении сгенерированного псевдослучайного числа n на все целые числа от 2 до $\left[\sqrt{n}\right]$ [2].

2.2. Метод на основе малой теоремы Ферма

Тест также оформлен в виде специально написанной функции. Он использует утверждение малой теоремы Ферма о том, что если n простое, то выполняется условие: при всех $a \in \{2,3,\ldots,n-1\}$, для которых НОД(a,n)=1, имеет место сравнение

$$a^{n-1} \equiv 1 \pmod{n}. \qquad (2)$$

Обратное утверждение неверно. Если сравнение (2) не выполнено, хотя бы для одного $a \in \{2,3,\ldots, n-1\}$, то n — составное.

Для сокращения времени генерации удобно использовать в качестве a («свидетеля» простоты), известные (табулированные) простые числа, что позволяет исключить проверку условия НОД(a,n)=1. Тогда можно применить следующий вероятностный алгоритм тестирования на простоту: $r \leftarrow 1$.

Выбираем из упорядоченного по величине массива простых чисел r-е простое число.

Проверяем выполнимость сравнения (2):

если сравнение не выполнено, то ответ «n — составное»;

если сравнение выполнено, то $r \leftarrow r + 1$;

если $r < U$, то переходим к шагу 2;

если $r > U$, то ответ «n — псевдопростое с точностью $1 - 2^{-U}$», где $U > 1$ — коэффициент точности.

2.3. Метод на основе теста Соловэя — Штрассена

Этот метод основан на следующей теореме. Для любого нечетного n следующие условия эквивалентны: n — простое, для любого $a \in Z$ выполняется сравнение

$$a^{\frac{n-1}{2}} = \left(\frac{a}{n}\right) \pmod{n}. \qquad (3)$$

Тогда можно использовать следующий вероятностный алгоритм тестирования на простоту:

1) $r \leftarrow 1$;

2) выбираем из упорядоченного по величине массива простых чисел r-е простое число;

3) проверяем выполнимость сравнения (3);

4) если сравнение не выполнено, то ответ «n — составное»;

5) если сравнение выполнено, то $r \leftarrow r + 1$;

6) если $r < U$, то переходим к шагу 2;

7) если $r > U$, то ответ «n — псевдопростое с точностью $1 - 2^{-U}$», где $U > 1$ — коэффициент точности.

2.4. Метод на основе теста Миллера — Рабина

Пусть n — нечетное и $n - 1 = 2^s t$; t — нечетное. Если число n является простым, то при всех $a \geq 2$ выполняется сравнение

$$a^{n-1} \equiv 1 \pmod{n}. \qquad (4)$$

Поэтому, рассматривая элементы $a^t, a^{2t}, \ldots, a^{2^{s-1}t}$, можно заметить, что либо среди них найдется равный $-1 \pmod{n}$, либо $a^t \equiv 1 \pmod{n}$. На этом замечании основан следующий вероятностный тест простоты:

1) $r \leftarrow 1$;

2) выбираем из упорядоченного по величине массива простых чисел r-е простое число;

3) вычисляем $a^t \pmod{n}$;

4) если $a^t \equiv \pm 1 \pmod{n}$, то переходим к шагу 3;

5) $\left(a^t\right)^2, \left(a^t\right)^4, \left(a^t\right)^6 \ldots \left(a^t\right)^{2^{s-1}} \pmod{n}$ до тех пор, пока не появится -1;

6) если ни одно из этих чисел не равно -1, то ответ «n — составное»;

7) если мы достигли -1, то $r \leftarrow r + 1$;

8) если $r < U$, то переходим к шагу 2;

9) если $r > U$, то ответ «n — псевдопростое с точностью $1 - 4^{-U}$», где $U > 1$ — коэффициент точности.

2.5. Метод генерации псевдослучайных чисел

Необходимые для работы описанных выше алгоритмов псевдослучайные числа формировались линейным конгруэнтным генератором вида

$$X_{i+1} = (X_i \cdot a + 1) \bmod m,$$

где $a = 6364136223846793005$; $m = 2^{p4}$. Параметры a и m были взяты из таблицы результатов спектрального теста [1]. При этих числах последовательность имеет максимальную длину периода и обладает хорошими статистическими свойствами. Для того чтобы тесты всегда проводились с различными числами, перебор данной псевдослучайной последовательности начинался с разного члена этой последовательности, номер которого, в свою очередь, генерировался встроенным в Borland C++ 3.1 генератором псевдослучайных чисел.

3. Результаты численных экспериментов

Описанные выше алгоритмы были реализованы в виде специальных программ, написанных на языке Borland C++ 3.1 с использованием подключаемых модулей для элементарных арифметических операций с большими числами [1]. В теле программы был организован счетчик времени ее работы. Время генерации простого (псевдопростого) числа являлось выходным параметром программы, так же как само число и количество занимаемых им бит. Программа, реализующая какой-либо из описанных выше методов, запускалась не менее 25 раз на ПК со следующими параметрами: Toshiba Satellite L100–194 15"XGA CM–1700/256Мб DDRII/40Гб/DVD–RW/WiFi/WXPh. Время генерации усреднялось по всем полученным реализациям.

Результаты работы программ представлены на рис. 1 в виде зависимостей времени генерации от числа бит. Особенностью зависимостей 2–4 является их флуктуирующий характер. Амплитуда флуктуаций уменьшается с ростом числа реализаций, используемых при получении усредненной зависимости (на единичных реализациях амплитуда флуктуаций на порядок выше). Наличие флуктуаций обусловлено вероятностным характером алгоритмов генерации простых чисел, отвечающих кривым 2–4. Подобные флуктуации отсутствуют на кривой 1, отвечающей детерминированному алгоритму пробных делений.

Как известно [4], теоретическая сложность вычислений метода пробных делений (кривая 1) оценивается соотношением

$$N = \mathrm{O}\left(\sqrt{n}\right), \tag{5}$$

где n — исследуемое число.

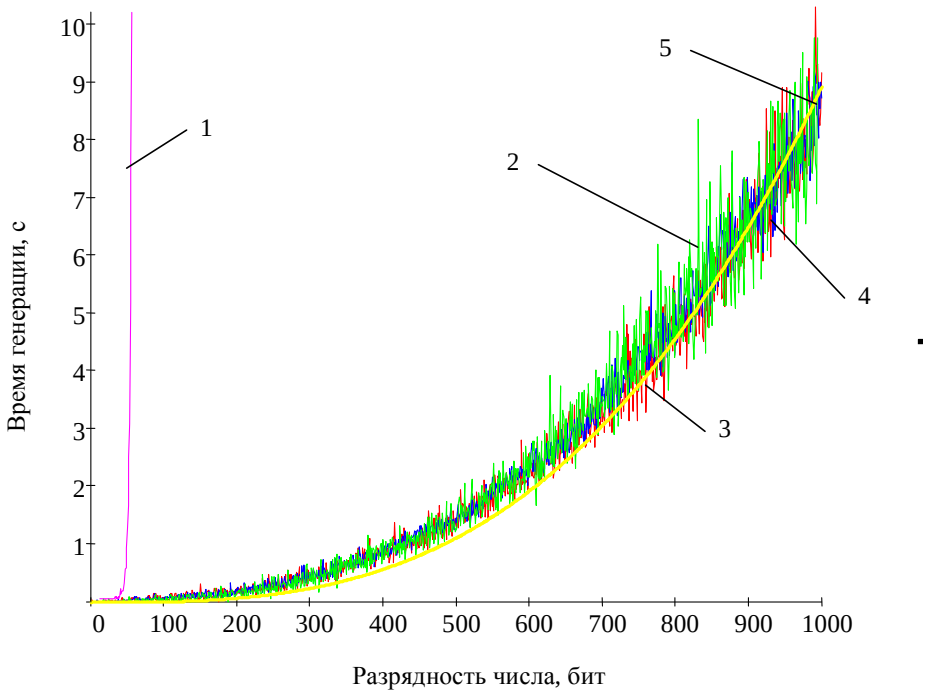

Рис. 1. Зависимость времени генерации простого числа от количества разрядов:
1 — методом пробного деления; 2 — методом на основе теста Миллера — Рабина;
3 — методом на основе малой теоремы Ферма; 4 — методом на основе теста Соловэя — Штрассена;
5 — теоретическая оценка сложности вычислений

В свою очередь теоретическая сложность вычислений методов на основе теста Миллера — Рабина (2), малой теоремы Ферма (3) и теста Соловэя — Штрассена (4) оценивается как [5]

$$N = \mathrm{O}\left(\log^3 n\right) = \mathrm{O}\left(k^3\right), \tag{6}$$

где k — разрядность числа. Таким образом, вычислительная сложность у последних трех тестов не ниже, чем у метода пробных делений ($\mathrm{O}(\log^3 n) << \mathrm{O}(n^{0,5})$), что демонстрируется кривыми на рис. 1. Высокая вычислительная сложность метода пробных делений делает его малопригодным для практической реализации.

Кривая 5 на рис. 1 описывается зависимостью $N = 8{,}913 \cdot 10^{-9} k^3$ и соответствует теоретической оценке с ограничивающей постоянной $C = 8{,}913 \cdot 10^{-9}$, где экспериментальная зависимость была аппроксимирована теоретической зависимостью методом наименьших квадратов. Из сравнения этой кривой с кривыми 2–4 видно, что реализованные программы генерации простых чисел с числом разрядов, превышающим 900 бит, работают чуть быстрее, чем предсказывает теоретическая оценка.

Заключение

Таким образом, в работе представлены алгоритмы генерации больших простых чисел на основе методов пробного деления, Миллера — Рабина, малой теоремы Ферма, Соловэя — Штрассена, непосредственно пригодные для программной реализации в среде Borland C++ 3.1. Выполнены численные расчеты скорости генерации от числа разрядов. Определена нижняя граница числа разрядов, равная 900 бит, начиная с которой разработанные программы работают быстрее, чем предсказывает теоретическая оценка сложности вычислений.

Литература

1. Вельшенбах М. Криптография на Си и C++ в действии / М. Вельшенбах. – М. : Триумф, 2004. – 464 с.

2. Смарт Н. Криптография / Н. Смарт. – М. : Техносфера, 2005. – 528 с.

3. Фергюсон Н., Шнайер Б. Практическая криптография / Н. Фергюсон, Б. Шнайер. – М. : Вильямс, 2005. – 424 с.

4. Василенко О.Н. Теоретико-числовые алгоритмы в криптографии / О.Н. Василенко. – М. : МЦНМО, 2003. – 328 с.

5. Черемушкин А.В. Лекции по арифметическим алгоритмам в криптографии / А.В. Черемушкин. – М. : МЦНМО, 2002. – 104 с.

Кукало Иван Анатольевич
Студент гр. 1А5 ТУСУРа
Эл. почта: ka1ita@ms.tusur.ru

Миклин Павел Александрович
Доцент кафедры радиоэлектроники и защиты информации ТУСУРа
Эл. почта: mpa@gpb.tomsknet.ru

Литвинов Рудольф Викторович
Канд. физ.-мат. наук, доцент кафедры радиоэлектроники и защиты информации ТУСУРа
Эл. почта: litvinovrv@rzi.tusur.ru

I.A. Kukalo, P.A. Miklin, R.V. Litvinov
Algorithms of generation of the pseudo prime numbers in «Borland C++ 3.1»

The experimental investigation of the performance of the generation algorithms of the prime numbers has been carried out. These algorithms are based on the methods of trial divisions, Miller-Rabin, Soloway - Strassen and minor Fermat's theorem, using the filter of division into small prime number. For each type of the method the dependences of the time generation on the byte length have been plotted on the assumption of averaging of the large ensemble of partial realizations.