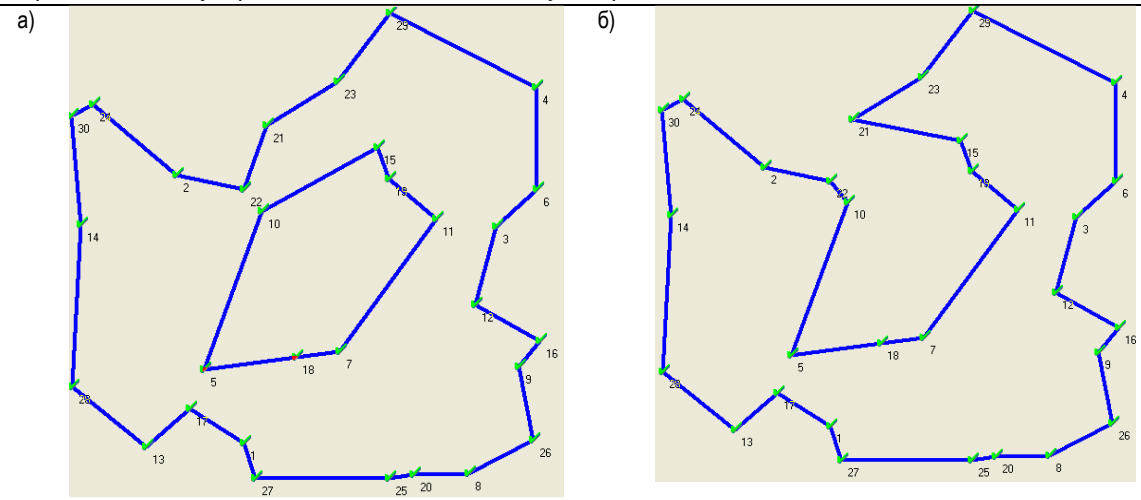

Рис. 3. *Выделение конечного маршрута коммивояжера: а – вид после определения всех контуров; б – вид после объединения отдельных контуров*

двумя точками, которые проводятся в большом количестве при каждой итерации алгоритма. Определение расстояния требует двух операций умножения и одно вычисление квадратного корня за раз: $l = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$, где x_1, y_1, x_2, y_2 – координаты точек $F_1(x_1,y_1)$ и $F_2(x_2,y_2)$, l – расстояние между точками. Кроме того, вычисление условия попадания точки в область эллипса, требует двух операций деления за одну проверку: $\frac{l_1 + l_2}{3} \le \frac{l}{2}$ (при эксцентриситете равном $\frac{2}{3}$), где l_1 и l_2 – расстояния от новой точки до фокусов эллипса, l – расстояние между фокусами эллипса.

Что касается временных затрат на вычисление, то получим следующую зависимость: $t \to O\left(\frac{1+n}{2} \cdot n\right)$, при количестве городов, равном n. В приведенной формуле учтено время на определение принадлежности точки к области эллипса, и не учтено время на вычисление крайних точек всей области их расположения, образующих выпуклый многоугольник, в связи с незначительностью затрат на их определение. Формула временной зависимости объясняется наличием в алгоритме двух основных циклов. Первый цикл выполняется до тех пор, пока все города не будут включены в маршрут. Второй цикл (вложенный) выполняется для каждого отдельного ребра, т.е. участка пути, непосредственно соединяющего **только** 2 города, и в нем, путем перебора всех не включенных в маршрут городов, определяется город, ближайший к текущему ребру, и включается в маршрут.

Сокращение объема вычислений может быть достигнуто путем хранения динамически изменяющихся данных о расстояниях между городами, еще не попавшими в маршрут коммивояжера, и городами, уже попавшими в маршрут. В то же время это приведет к увеличению затрат на хранение данных и соответственно к увеличению требований к оперативному запоминающему устройству ЭВМ.

Кроме сказанного выше, можно еще отметить, что вычислительный алгоритм достаточно простой и легко программируется на ЭВМ, что является безусловным плюсом при возможной в дальнейшем модернизации алгоритма.

Решение примеров на ЭВМ типа IBM PC показало практическую реализуемость алгоритма и его достаточную эффективность. Разработанный метод может быть применен для решения самых разнообразных технических, экономических, социальных и других задач.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. Пападимитриу Х., Стайглид К. Комбинаторная оптимизация: Алгоритмы и сложность. - М.: Мир, 1985.
2. Дискретный анализ и исследование операций. Январь-июнь 2000. Серия 2. Том 7, №1.

Материал поступил в редакцию 28.01.08

SHUT V.N. The decisions of a task of the direct-sales representative an ellipse by narrowing

In given clause the algorithm of the approached decision of a task of the direct-sales representative is offered. The algorithm develops idea of construction of a route of the minimal weight. The ready route is exposed to the analysis with the purpose of revealing ways of his improvement. The education of an initial contour occurs by inclusion of extreme tops (if to spend analogy to a map and cities: the most northern, most southern, most east and most western city), not belonging to a route. The estimation of numerical experiment is given.

УДК 004.056.55

Поденок Л.П.

БЫСТРЫЕ АЛГОРИТМЫ ЭКСПОНЕНЦИАЛЬНЫХ ПРЕОБРАЗОВАНИЙ ДЛЯ КРИПТОСИСТЕМ С ОТКРЫТЫМ КЛЮЧОМ

Введение. В настоящее время большое количество информации хранится и обрабатывается в электронном виде. Эта информация широчайшего диапазона, как экономической, так и политической важности, имеет различные требования к конфиденциальности, надежности и доступности. В подавляющем большинстве случаев передача таких конфиденциальных (секретных) материалов осуществляется по незащищенным каналам с использованием криптографических средств защиты. Криптографическими преобразованиями данных с целью обеспечения их конфиденциальности занимается криптография и она обеспечивает поддержку трех основных криптографических услуг, в числе которых обеспечение секретности, аутентификация и контроль целостности.

Криптосистемы делятся на два больших класса: криптосистемы с одним секретным ключом, т.н. симметричные криптосистемы, и криптосистемы с открытым ключом [1-4]. Криптосистемы с открытым ключом требуют хранения всего одного экземпляра секретного ключа на принимающей стороне, в то время как при использовании симметричных криптосистем свой экземпляр секретного ключа тре-

буется хранить каждой из сторон, что вдвое повышает вероятность компрометации. В случае большого количества взаимодействующих пользователей каждому из них при использовании симметричного шифрования потребуется хранить N секретных ключей для всех пользователей, в то время, как в криптосистемах с открытым ключом пользователю требуется хранить только свой секретный ключ и нести ответственность только за него.

Несмотря на функциональные преимущества криптосистем с открытым ключом, область их применимости на практике достаточно ограничена вследствие более низкой производительности по сравнению с симметричными схемами шифрования/дешифрования. Криптосистемы с открытым ключом в основном используются для управления секретными ключами, а также для реализации цифровой подписи.

Стойкость большинства криптосистем с открытым ключом основана либо на проблеме дискретных логарифмов, либо на проблеме разложения целых чисел на простые множители и, в связи с этим, существенно зависит от длины модуля. Чем длиннее модуль криптосистемы, тем выше ее криптостойкость и тем ниже ее производительность. Низкая производительность вызвана прежде всего тем, что обычно в качестве модуля криптосистемы используются очень большие целые числа с разрядностью, существенно превышающей базовую разрядность (длину рабочего слова) процессора. Например для большинства современных процессоров широкого назначения базовая разрядность составляет 64 или 32 бита, при том, что разрядность модуля может превышать 2048 бит. Несмотря на интенсивный рост специализированных вычислительных устройств для криптографических приложений, именно такие процессоры составляют в настоящее время основную вычислительную базу потребителей криптографических средств с открытым ключом. В связи с этим задача эффективного вычисления модулярной экспоненты с помощью процессоров широкого назначения не становится менее актуальной.

Основные преобразования. С формальной точки зрения криптосистемы оперируют с большими целыми числами и их последовательностями. Разрядность двоичного представления таких чисел, лежит в диапазоне от 2^{64} до 2^{2048} и более бит, что обусловливает высокую вычислительную сложность практически всех алгоритмов, включая самые тривиальные. Структура алгебраических образований, в которых выполняются криптографические преобразования, либо не позволяет добиться высокой эффективности вычислений известными методами ($GF(p)$), либо не известна (кольцо по модулю pq).

Модулярная экспонента и дискретное возведение в степень больших целых чисел являются основными преобразованиями, используемыми для реализации криптосистем с открытым ключом. От эффективности реализации этих функций зависит производительность как основных шифрующих преобразований, так и многих вспомогательных алгоритмов, использующихся для генерации определяющих параметров в криптосистемах. С формальной точки зрения, обе функции имеют один и тот же вид

$$C = A^B \bmod N, \qquad (1)$$

где C - значение функции, A - основание, B - показатель, N - модуль. В зависимости от того, что принимается за аргумент функции, а что – за ее параметр, выражение (1) определяет либо класс модулярных экспоненциальных функций $y = a^x \bmod N$, либо класс дискретных степенных функций $y = x^x \bmod N$, каждый из которых требует своего подхода для эффективной реализации.

Поскольку обе функции, формально представленные выражением (1), имеют общий параметр N, существуют и пригодные для обеих функций методы преобразования исходного выражения, ведущие к сокращению вычислительных затрат. В ряде случаев существенного упрощения вычислений можно достичь, используя известное из теории чисел соотношение $a^b \equiv a^{b \bmod \varphi(n)}$, где $\varphi(n)$ - функция Эйлера [6], которое в ряде случаев позволяет значительно уменьшить показатель.

В подавляющем большинстве практических случаев обе функции вычисляются с помощью некоторой последовательности арифметических операций (сложение, умножение, возведение в квадрат или другую степень). Из основных свойств сравнений [5] следует, что после каждой из этих операций может выполняться приведение по модулю N, при этом окончательный результат будет таким, как если бы приведение по модулю выполнялось единственный раз на конечном этапе вычислений. Это свойство модулярных вычислений позволяет существенно сократить количество нетривиальных элементарных операций, таких как умножение.

Если модуль N является составным и известно его разложение на множители, вычисление (1) можно упростить, используя свойства сравнений и Китайскую теорему об остатках (КТО) для восстановления результата из остаточного представления. Пусть $N = \prod_k p_k$, тогда (1) можно перевести в остаточное представление

$$C_k = A_k^{B_k} \bmod p_k, \qquad (2)$$

где $C_k = C \bmod p_k$, $A_k = A \bmod p_k$, $B_k = B \bmod \varphi(p_k)$. Для случая, когда $N = pq$, причем $\log p \approx \log q$, что имеет место для практически значимых случаев реализации криптосистемы RSA, длины обеих модулей p и q составляют около половины длины модуля N. Умножение по модулям p и q может быть выполнено быстрее в 2 ... 4 раза в зависимости от используемого алгоритма. Как следствие, две составляющие экспоненты Cp и Cq будут вычислены также в 2 ...4 раза быстрее. Метод применим для любой криптосистемы, стойкость которой основана на разложении составного числа на множители. Недостатком метода является использование в процессе дешифрования помимо секретного показателя d не менее секретных элементов p и q, с использованием которых однозначно вычисляется d, что может увеличить вероятность компрометации за счет увеличения количества копий секретных элементов шифра. Однако при надлежащей реализации вычислительной системы и криптосистемы этого можно избежать. Для восстановления значения C из его остатков Cp и Cq можно применить любой подходящий метод, например, представление по смешанным основаниям или КТО.

Для криптосистем, стойкость которых основана на проблеме дискретного логарифмирования, метод вычисления степеней по нескольким меньшим модулям, представленный (2), не работает, поскольку вычисления выполняются в поле $GF(p)$, где p - простое число, или в полях $GF(p^m)$, где использовать представление (2) и КТО невозможно.

Вычисление экспоненты в *GF(p)* методом аддитивно-субтрактивной цепи. Одним из известных алгоритмов возведения в степень является метод, основанный на сканировании двоичного представления показателя. Для каждого бита выполняется возведение в квадрат результата, полученного на предыдущем шаге, за которым следует умножение квадрата на основание, если бит установлен в единицу. Если двоичное представление показателя содержит большое количество единиц, эффективность метода снижается.

При вычислении экспоненты в кольце целых Z_m по некоторому постоянному, или многократно использующемуся основанию b, не являющимся делителем нуля в Z_m, и переменному показателю d может использоваться более эффективный метод [6].

Суть метода состоит в том, что для произвольного показателя d может быть построена последовательность $\{l = a_0, a_1, ..., a_r = d\}$ такая, что любой последующий элемент a_j может быть выражен в виде либо $2a_{j-1}$, либо $2a_{j-1}+1$, либо $2a_{j-1}-1$.

Данную последовательность назовем аддитивно-субтрактивной цепочкой. Эта цепочка определяет последовательность степеней, каждая из которых получается либо возведением в квадрат предыдущей, умножением предыдущей на основание, либо умножением

предыдущей на величину, мультипликативно обратную основанию, что эквивалентно делению в поле.

Здесь следует отметить, что в криптографических приложениях в качестве основания экспоненты всегда выступает элемент, мультипликативно обратный для которого существует, либо всегда можно выбрать основание с такими свойствами. Данный метод в среднем использует вдвое меньше умножений, чем метод сканирования двоичного представления показателя, а также метод разбиения показателя на нечетные парциальные части с выделением групп нулей.

Если число единиц в двоичном представлении числа d велико, его всегда можно представить в виде разности двух чисел, оба из которых содержат гораздо меньшее число единиц, чем исходное. При этом формально можно записать

$$b^d = b^{d+a-a} = b^{d+a}b^{-1a} = b^{d+a}(b^{-1})^a. \qquad (3)$$

Выражения b^{d+a} и $(b^{-1})^a$ вычисляются используя почти оптимальное число умножений, если известно b^{-1}. В $GF(p)$ и кольце Z_m нахождение b^{-1}, мультипликативно обратного к b, не представляет существенной сложности, при этом если b является константой и неоднократно используется в вычислениях, то временные затраты на вычисление b^{-1} могут игнорироваться.

В качестве примера рассмотрим вычисление экспоненты с показателем, состоящим полностью из единиц в двоичном представлении. Пусть $d = 2^n - 1$. Следуя бинарному методу возведения в степень, мы должны выполнить n-1 возведений в квадрат и n-1 умножений, что в целом составит $2(n\text{-}1)$ умножений. С другой стороны, мы можем возвести основание сначала в степень $2n$, выполняя n возведений в квадрат, затем разделить полученный результат на основание, или, что эквивалентно, умножаем на мультипликативно обратный к основанию элемент кольца. Таким образом, процедура занимает всего $n+1$ мультипликативных операций. Метод становится эффективным, если в двоичном представлении показателя d содержится группа единиц длиной $k \geq 3$, что иллюстрируется в таблице 1 для показателя d = 6062, имеющего в двоичном представлении 4 нуля и 9 единиц и требующего выполнения 12 возведений в квадрат и 8 умножений, всего 20 мультипликативных операций.

Таблица 1. *Факторизация показателя d = 6062*

№	Факторизация d = 6062	№$_{mul}$	№$_{div}$	№$_{total}$
0	1011110101110	8	0	8
1	1011110110000 - 10	6	1	7
2	1011111000000 - 10000 - 10	5	2	7
3	1100000000000 - 1000000-10000-10	1	3	4

В первом столбце таблицы указан шаг факторизации показателя, во втором приведен вид показателя, представленного в виде разностей. Третий, четвертый и пятый столбцы содержат число умножений, делений (умножений на мультипликативно обратный элемент поля) и общее число мультипликативных операций соответственно. Количество возведений в квадрат зависит только от длины показателя и остается постоянным.

Как можно заметить, группа длиной k = 2 (011), которая представляется в виде разности 100-1 на шаге 3, не приводит к локальному сокращению количества мультипликативных операций, однако в результате замены происходит перенос единицы на место нуля, примыкающего к группе слева. Если этот нуль изолирован, что как раз имеет место в данном случае, то следующая группа единиц растет по длине, обеспечивая выигрыш на следующем шаге факторизации. Таблица 2 иллюстрирует сказанное выше для показателя d = 5851 = 1011011011011, двоичное представление которого содержит только группы единиц длиной 2.

Таблица 2. *Факторизация показателя d = 5851*

№	Факторизация d = 5851	№$_{mul}$	№$_{div}$	№$_{total}$
0	1011011011011	8	0	8
1	1011011011100 -1	6	1	7
2	1011011100000 -100 -1	5	2	7
3	1011100000000 -100000 -100 -1	1	3	4
4	1100000000000 -100000000 -100000 -100 -1			

Из таблицы видно, что при переносе единицы в изолированный нуль группа единиц длиной k = 2 увеличивается и в результате обеспечивает выигрыш в 3 умножения.

Прежде чем начать вычисление экспоненты представленным методом требуется построить массив, указывающий точки деления. Модифицированное двоичное представление показателя b используется для управления умножениями в то время, как массив точек деления используется для управления умножениями на мультипликативно обратный элемент. Как и алгоритм сканирования бинарного представления показателя, на каждом шаге используется возведение в квадрат.

Алгоритм сканирования и построения массива точек деления принимает на входе строку бит b длиной n, которая является двоичным представлением показателя, и сканируя ее справа налево, заменяет все вхождения подстрок вида 011, 0111, ... на 100, 1000, ... , синхронно с этим формируя вторую строку d с установленными битами в позициях, где алгоритм сканирования начинал замену подстроки.

Конкретная реализация данного алгоритма зависит от архитектуры вычислительного устройства. На универсальных архитектурах алгоритм может строиться на базе команд арифметического сдвига или циклического через бит переноса.

Как и вычисление экспоненты, данный алгоритм является последовательным, однако вычислительные затраты на сканирование показателя, его модификацию и создание битовой строки с точками деления пренебрежимо малы по сравнению с затратами на вычисление экспоненты. Способ представления точек деления слабо влияет на производительность алгоритма, в частности, алгоритм сканирования и вычисления экспоненты с использованием стека для хранения точек деления представлен в [6].

В отличие от алгоритмов, использующих только возведение в квадрат и умножение на основание экспоненты, данный алгоритм использует деление, в связи с чем необходимым условием является существование мультипликативно обратного элемента. В частности, алгоритм можно использовать в конечных полях $GF(p)$ при постоянном или многократном использовании основания. В этом случае можно раз и навсегда вычислить элемент, мультипликативно обратный основанию.

Представленный подход с использованием деления (умножения на мультипликативно-обратный элемент) может использоваться для модификации методов возведения в степень, отличных от метода сканирования двоичного представления.

Например, в методе с разбиением показателя на нулевые и ненулевые нечетные группы данный подход может использоваться с целью удлинения нулевых групп.

Модулярное умножение с использованием вычетов, не являющихся наименьшими. В процессе вычислений модулярной экспоненты $c = C^x (\mathrm{mod} N)$ или степенной функции $c = x^C (\mathrm{mod} N)$ возведением в квадрат и умножением используются наименьшие приведенные вычеты

$$c = (a \cdot b) \mathrm{mod} N = ((a \mathrm{mod} N) \cdot (b \mathrm{mod} N)) \mathrm{mod} N, \qquad (4)$$

найти которые можно, например, используя алгоритм деления с остатком. Целочисленное деление исходного числа (неприведенного вычета) на модуль с целью получения точного частного является одним из методов нахождения приведенного вычета, при этом используется соотношение

$$a \mathrm{mod} N = a - \left\lfloor \frac{a}{N} \right\rfloor \cdot N \qquad (5)$$

Нахождение точного частного $Q = \left\lfloor \frac{a}{N} \right\rfloor$ является достаточно сложной проблемой с вычислительной точки зрения, если разрядность чисел, участвующих в операциях, существенно превышает разрядность процессора. Однако в процессе вычисления модулярной экспоненты можно использовать любой вычет из подходящего класса. Окончательный результат приводится к наименьшему вычету после того, как будет выполнено последнее умножение. Например, вместо наименьших вычетов $c = a \cdot b \bmod N$ в процессе вычислений можно использовать вычеты, не являющиеся наименьшими (неприведенные)

$$c = a \cdot b \bmod N + n \cdot N, n \in \{0,1,2\}, \quad (6)$$

вычислить которые значительно легче, чем полностью приведенные к наименьшему значению.

Если $2^{n-3} < N < 2^{n-2}$, то для любого $x < 2^{2n}$ приближение к частному вычисляется следующим образом

$$Q = \left\lfloor \left\lfloor x \cdot 2^{-(n-3)} \right\rfloor \left\lfloor \frac{2^{2n}}{N} \right\rfloor \cdot 2^{-(n+3)} \right\rfloor. \quad (7)$$

Если ввести символические операции $w[i]$ и $[i]w$, означающие взятие i младших и старших бит двоичного представления числа w соответственно, а также записать $\left\lfloor \frac{2^{2k+4}}{N} \right\rfloor$ как $\tilde{N}$, то алгоритм нахождения неприведенного вычета можно представить как

$$\begin{array}{lll} x[2n] & \leftarrow & a[n] \cdot b[n] \\ Q[2n+6] & \leftarrow & \tilde{N}[n+3] \cdot [n+3]x \\ y[n] & \leftarrow & x[n+3] - [n+3]Q \cdot N[n] \end{array} \quad (8)$$

Данный алгоритм в большей степени ориентирован на аппаратную реализацию. В этом случае операции $w[i]$ и $[i]w$ выполняются тривиально, таким образом вычислительные затраты на приведение x в диапазон $[0 \ldots 2^n]$ составят два длинных умножения и одно длинное сложение. В случае программной реализации алгоритма на универсальной архитектуре потребуются дополнительные затраты на получение $n+3$ старших бит чисел x и Q.

Если n кратно 8, то выделение $n+3$ старших бит может быть выполнено посредством сдвига числа вправо на 3 бита. Но в связи с тем, что система команд универсальных процессоров не позволяет быстро выполнять сдвиги чисел произвольной длины на произвольное число бит, использование алгоритма в приведенном виде на такой архитектуре нецелесообразно. В частности, при использовании процессоров типа x86 сдвиг числа длиной m слов на 3 бита потребует m сдвигов двойной точности, столько же команд маскирования результатов сдвига, и объединения по «или» в выходном результате без учета чтения и записи в память. Однако данный алгоритм может быть изменен таким образом, чтобы обойтись без сдвигов вообще.

Рассмотрим соотношение (7) для аппроксимации частного. Выражение $\left\lfloor \frac{2^{2n}}{N} \right\rfloor$ определяет количество удерживаемых значащих разрядов двоичного представления рациональной дроби $\frac{1}{N}$

$$\overbrace{\underbrace{00\ldots01}_{n-2} xx\ldots x}^{2n} . \varepsilon\varepsilon\ldots,$$

которое равно $n+3$. Если n кратно длине машинного слова, для того чтобы удержать и обработать дополнительные 3 бита на универсальной архитектуре потребуется целое слово. Поэтому мы можем увеличить точность двоичного представления рациональной дроби до $n+8$ бит, сохранив при этом как производительность алгоритма, так и количество используемой памяти

$$Q = \left\lfloor \left\lfloor x \cdot 2^{-(n-8)} \right\rfloor \left\lfloor \frac{2^{2n}}{N} \right\rfloor \cdot 2^{-(n+8)} \right\rfloor. \quad (9)$$

Алгоритм, соответствующий (9), имеет вид

$$\begin{array}{lll} x[2n] & \leftarrow & a[n] \cdot b[n] \\ Q[2n+16] & \leftarrow & \tilde{N}[n+8] \cdot [n+8]x \\ y[n] & \leftarrow & x[n+8] - [n+8]Q \cdot N[n] \end{array} \quad (10)$$

В отличие от (8) реализация алгоритма (10) для универсального процессора и побайтно адресуемой памяти не требует выполнения сдвигов, поскольку результат может быть достигнут чтением по другому адресу. Поскольку N является модулем криптосистемы и, в связи с этим, используется многократно, значение выражения $\left\lfloor \frac{2^{2n}}{N} \right\rfloor$ вычисляется один раз и сохраняется как параметр криптосистемы.

СПИСОК ЦИТИРОВАННЫХ ИСТОЧНИКОВ

1. R. L. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems // Communications of the ACM, - Vol.21, - N2, - Feb 1978, - P.120-126.
2. Diffie W., Hellman M. New directions in cryptography. // IEEE Trans. on Information Theory, - Vol.IT-22, - 6 November 1976. - P.644-654.
3. T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms // IEEE Trans. Information Theory, - Vol.IT-31, - 1985, - P.469-472.
4. Nat'l Inst. of Standards and Technology (NIST), FIPS Publication 186: Digital Signature Standard. May 19, - 1994.
5. И. М. Виноградов. Основы теории чисел. - М.: Наука, 1981, - 176с.
6. Л. П. Поденок, Р. Х. Садыхов. Быстрый алгоритм экспоненциального преобразования в системах защиты информации / Тезисы доклада. Труды научно-технической конференции Брестского политехнического института. – 1996. - С.102-105.

Материал поступил в редакцию 22.01.08

PODENOK L.P. Fast algorithms exponention of transformations for crypto-systems with an open key

The method is submitted and the fast algorithm of calculation the module exhibitors based on representation of a parameter as additive-subtraction circuits allowing in average to reduce number of multidigit multiplication in 2 times. The fast method of calculation not of the completely given deductions is submitted.

УДК 577.21

Кузавко Ю.А.

О МАТЕМАТИЧЕСКОЙ ДОПУСТИМОСТИ НЕЕДИНСТВЕННОСТИ СТАНДАРТНОГО, НЕСТАНДАРТНЫХ И ИСКУССТВЕННЫХ КОДОВ ГЕНОМА ЧЕЛОВЕКА

Введение. В молекулярной генетике доказано, что соматическая клетка организма (растения) содержит полную программу воспроизведения организма (растения). Для растений такое явление давно наблюдается в природе, реализовано в экспериментах in vitro. В 2003 г. были завершены многолетние международные экспериментальные исследования кода генома человека, содержащего