УДК 512.542

ОБ АЛЬТЕРНАТИВНЫХ СПОСОБАХ ЛОГАРИФМИРОВАНИЯ В КОНЕЧНЫХ ГРУППАХ

Д.Е. Островский

МГТУ им. Н.Э. Баумана, Москва, Российская Федерация
e-mail: crypto-expert@yandex.ru; crypto.iu8@gmail.com

Задача взятия дискретного логарифма сегодня известна, как одна из наиболее вычислительно трудоемких. Не представлено ни одного алгоритма, способного решать ее за полиномиальное время. Это одно из немногих вычислительно-асимметричных преобразований, используемых в современной криптографии. Выполнена попытка упростить решение задачи с целью оценить обоснованность применения дискретного логарифма в асимметричных протоколах. Разработан табличный алгоритм вычисления дискретного логарифма в произвольной циклической группе порядка n, имеющий сложность $\sqrt{n}$ элементарных операций. Математическое ожидание трудоемкости алгоритма составляет $\sqrt{n}/2$ элементарных операций. Поэтому некоторые показатели вычисляются быстрее остальных, причем они не обязательно по значению большие или маленькие, и анализ таких "небезопасных" показателей весьма затруднителен. На практике алгоритм может оказаться эффективнее алгоритма согласования. Для группы Zp — умножения чисел по простому модулю путем синтеза разработанного алгоритма с техникой факторных баз, применяемой в алгоритме Адлемана, получен алгоритм, имеющий субэкспоненциальную сложность $L_n[1/2, \sqrt{3}]$. В настоящее время известны более эффективные алгоритмы, тем не менее, показатель математического ожидания трудоемкости может сыграть существенную роль на практике.

***Ключевые слова*:** дискретный логарифм, циклические группы, группа Zp, отображение Ньютона, алгоритм Адлемана.

ON ALTERNATIVE WAYS OF TAKING THE LOGARITHM IN FINITE GROUPS

D.E. Ostrovskii

Bauman Moscow State Technical University, Moscow, Russian Federation
e-mail: crypto-expert@yandex.ru; crypto.iu8@gmail.com

A task of taking the discrete logarithm is known today as one of most computationally complicated. There is no algorithm that can solve it in a polynomial time. This is one of the few computationally asymmetric transformations used in modern cryptography. An attempt is made here to simplify the problem solving, in order to assess the reasonableness of using the discrete logarithm in asymmetric protocols. A tabular algorithm for computing the discrete logarithm in an arbitrary cyclic group of the order n is designed that has a complexity of $\sqrt{n}$ elementary operations. An expectation of the algorithm complexity is $\sqrt{n}/2$ elementary operations. Therefore some indicators are computed faster than others, they are not necessarily large or small in magnitude at that, and analysis of these "unsafe" indicators is rather hard. In practice, this algorithm may be more efficient than the matching algorithm. For the Zp group (multiplication of numbers modulo a prime), by synthesizing the developed algorithm with the technique of factor bases used in the Adleman's algorithm, an algorithm is obtained having a subexponential complexity of $L_n[1/2, \sqrt{3}]$. At the present time, more efficient algorithms are known; nevertheless the indicator of complexity expectation may play a significant role in the practice.

***Keywords*:** discrete logarithm, cyclic groups, Zp group, Newton's map, Adleman's algorithm.

Задача вычисления дискретного логарифма в циклических группах по сей день не имеет эффективного решения. На вычислительной сложности этого преобразования основано подавляющее большинство алгоритмов асимметричной криптографии. Задача состоит в следующем. Пусть задана циклическая группа $(G,\ \bullet)$ с образующим (примитивным) элементом g, известен порядок группы $\operatorname{ord}(G) = n$. Для некоторого $h \in G$: $h = g^y$, $y \in \mathrm{N}$, требуется найти показатель степени y, в которую возведен образующий элемент.

В настоящее время самым эффективным алгоритмом решения такой задачи для случая произвольной циклической группы является алгоритм Сильвера–Полига–Хэллмана [1] (теорема В.И. Нечаева, 1965 г.). Алгоритм сводится к переходу для логарифмирования в подгруппы группы G простого порядка и достаточно эффективен в том случае, если порядок группы n не имеет больших простых чисел в разложении на множители.

Логарифмирование в подгруппах проводится с помощью алгоритма Гельфонда – Шенкса, более известного как алгоритм согласования [2]. Его сложность составляет $O\left(\sqrt{p} + \log_2 p\right)$, где p — простой порядок подгруппы, в которой осуществляется логарифмирование.

Таким образом, в случае, если в разложении порядка группы на множители есть простое число, сравнимое с n, например $n = 2p$, задача вычисления дискретного логарифма потребует выполнения $\sim \sqrt{p}$ операций. При этом в памяти будут храниться $2\sqrt{p}$ элементов группы G. В современной криптографии требования к значениям наименьших простых делителей p составляют 250 бит, поэтому для взлома таких криптосистем потребуется $\sim \sqrt{2^{250}}$ операций. Это число космических масштабов, собственно на этом и основана криптографическая стойкость асимметричных систем.

В настоящей статье предлагается еще один способ вычисления дискретного логарифма в произвольных циклических группах. Этот алгоритм не привнесет каких-либо улучшений, оценка сложности логарифмирования произвольного элемента группы по-прежнему будет составлять $\sqrt{p}$, тем не менее, на любом шаге алгоритма после проведения k операций умножения можно успешно находить $k^2/2$ показателей.

Как и хорошо известный ρ-метод Полларда [3, 4], алгоритм использует технику многократного итерирования отображений, но в отличие от ρ-метода справедлив для любых циклических групп, так как не использует каких-либо специфических свойств группы G, кроме свойства цикличности относительно операции " $\bullet$ ".

Алгоритм использует отображение $F\colon G^k \to G^k$, $k \in \mathbb{N}$, которое в некоторых источниках называется *отображением Ньютона*. Пусть вектор $v \in G^k$ и $v\,[i] \in G$ — i-я координата вектора v. Тогда вектор $w = F(v)$ получается следующим образом:

$$w\,[i] = v\,[i] \bullet v\,[i+1]\,, \quad 0 \le i < k-1; \qquad (1)$$
$$w\,[k-1] = v\,[0] \bullet v\,[k-1]\,.$$

Описание алгоритма. ***Постановка задачи.*** Для $b = a^y$, требуется найти y. Порядок группы $\operatorname{ord}(G) = n$. Сначала переходим в подгруппу простого

порядка p, где p — делитель n:

$$d = \frac{n}{p}; \quad g = a^d; \quad h = b^d.$$

Теперь переформулируем задачу логарифмирования для этой подгруппы — необходимо вычислить $x = \log_g h$, g — образующий в подгруппе.

Вычисляем все степени двоек, такие, что $2^i < p$. Возводим образующий элемент g во все эти степени:

$$g^1 \qquad g^2 \qquad g^4 \qquad \dots \qquad g^{2^k}, k = \log_2 p.$$

Каждый из получившихся элементов умножаем на $g^x = h$:

$$s_0: \qquad hg^1 \qquad hg^2 \qquad hg^4 \qquad \dots \qquad hg^{2^k}. \tag{2}$$

Показатели элементов последовательности (2) запишем в виде последовательности

$$t_0: \qquad x + 2^0 \qquad x + 2^1 \qquad x + 2^2 \qquad \dots \qquad x + 2^k \pmod p. \tag{3}$$

К последовательности (2), интерпретируя ее как вектор v в соотношениях (1), применим отображение Ньютона и получим последовательность $s_1 = F(s_0)$. Затем для всех $i \le k$ вычислим $s_i = F(s_{i-1})$. В итоге получим $k + 1$ векторов, которые запишем в виде матрицы S:

$$S = \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_k \end{pmatrix} = \begin{pmatrix} s[0][0] & s[0][1] & \dots & s[0][k] \\ s[1][0] & s[1][1] & \dots & s[1][k] \\ s[2][0] & s[2][1] & \dots & s[2][k] \\ \vdots & \vdots & \dots & \vdots \\ s[k][0] & s[k][1] & \dots & s[k][k] \end{pmatrix}. \tag{4}$$

В результате между элементами матрицы установятся следующие соотношения:

$$s\,[i]\,[j] = s\,[i-1]\,[j] \bullet s\,[i-1]\,[j+1], \quad 1 \le i \le k, \quad 0 \le j \le k-1$$

$$s\,[i]\,[k] = s\,[i-1]\,[0] \bullet s\,[i-1]\,[k], \quad 1 \le i \le k.$$

Каждый элемент матрицы S есть образующий g, возведенный в некоторую степень. Обозначим ее через $t[i][j]$ и запишем матрицу показателей T:

$$T = \begin{pmatrix} t_0 \\ t_1 \\ \dots \\ t_k \end{pmatrix} = \begin{pmatrix} & 2^0 & 2^1 & \dots & 2^k \\ 2^0 \times x & t[0][0] & t[0][1] & \dots & t[0][k] \\ 2^1 \times x & t[1][0] & t[1][1] & \dots & t[1][k] \\ \dots & \dots & \dots & \dots & \dots \\ 2^k \times x & t[k][0] & t[k][1] & \dots & t[k][k] \end{pmatrix}. \tag{5}$$

Соотношения для элементов матрицы T следующие:

$$t\,[i]\,[j] = t\,[i-1]\,[j] + t\,[i-1]\,[j+1] \pmod p, \quad 1 \le i \le k, \; 0 \le j \le k-1$$

$$t\,[i]\,[k] = t\,[i-1]\,[0] + t\,[i-1]\,[k], \quad 1 \le i \le k.$$

В результате, каждый элемент матрицы T будет иметь вид

$$t[i][j] = 2^i x + \left(\sum 2^q\right) \bmod p,$$

где коэффициентом при x будет двойка в степени номера строки, а свободным членом — некоторая сумма степеней двоек, взятая по модулю p.

В случае совпадения некоторых двух элементов в матрице S, их показатели на соответствующих позициях в матрице T также совпадут. Если это будут элементы из разных строк, то коэффициенты при x будут различными. Поэтому, приравняв эти элементы, мы получим уравнение вида

$$2^i x + a_1 = 2^j x + a_2 \,(\mathrm{mod}\ p) \tag{6}$$

с известными коэффициентами a_1 и a_2. Если $a_1 \neq a_2$, то уравнение (6) будет иметь единственное, отличное от нуля решение. Тем самым, удастся вычислить искомый показатель x.

Пример. В качестве примера рассмотрим циклическую группу по умножению по модулю простого числа $p = 503$. Вычислим дискретный логарифм числа 318 по основанию 5.

$$p - 1 = 502 = 2 \bullet 251; a = 5^2 = 25;\ b = 318^2 = 21;$$

2^i	0	2	4	8	16	32	64	128
a^{2^i}	1	122	297	184	155	384	77	396
ba^{2^i}	21	47	201	343	237	16	108	268
$2^1 \times x$	484	393	32	308	271	219	273	95
$2^2 \times x$	**78**	1	299	473	498	433	282	207
$2^3 \times x$	**78**	299						

Матрица S

Для верхнего 78, соответствующий показатель равен:

$$2^2 x + (0+2) + (2+4) \mod 251.$$

Показатель для 1: $2^2 x + (2+4) + (4+8) \mod 251$.
Для нижнего 78: $2^3 x + (0+2) + (2+4) + (2+4) + (4+8) \mod 251$.
Приравнивая показатели, получаем уравнение

$$2^2 x + 8 = 2^3 x + 26 \quad (\mathrm{mod}\ 251),$$

$$4x = 233 \quad (\mathrm{mod}\ 251).$$

По алгоритму Евклида $4a + 251b = 1$;

a	B	
0	1	251
1	0	4
-62	1	3
63	-1	1

$$4^{-1} = 63 \quad (\mathrm{mod}\ 251); x = 121 \quad (\mathrm{mod}\ 251);$$

$$x = ? \quad (\mathrm{mod}\ 2);\ \text{т.к. } 318^{251} = 5^{251} = 502,$$

$$\log_{502} 502 = 1 \ \Rightarrow x = 1 \ (\mathrm{mod}\ 2).$$

Поэтому ответ: $\log_5 318 = x = 121$.

Поскольку размер матрицы S невелик, возникает вопрос, всегда ли найдутся совпадающие элементы. Очевидно, что нет. Вычислим верхнюю границу числа показателей x, для которых в матрице S всегда найдутся совпадения. Для этого будем считать, что каждой паре элементов из матрицы T соответствует уникальное уравнение, имеющее ненулевое решение, отличное от остальных (кроме пар, лежащих в одной строке). Общее число таких пар

$$\max = \frac{N(N-1)}{2} - \frac{k(k+1)}{2}(k+1) \cong N^2, \tag{7}$$

$N = (k+1)^2$ — число элементов в матрице T. Таким образом, верхняя граница показателей, для которых всегда найдется совпадение, соответствует квадрату числа элементов в матрице, а число элементов в матрице — это число умножений в группе, которые требуются для ее вычисления.

Чтобы оценить, насколько близко лежит реальное число показателей к верхней границе, потребуется подсчитать число линейно зависимых уравнений, которые дают одинаковые решения. Но сначала приведем результаты экспериментов, в которых перебирались все пары и проводился подсчет уникальных решений. Эксперименты проводились для различных p (табл. 1).

Таблица 1

Эксперименты для двумерной матрицы

Простой модуль	$k = \log_2 p$	Число различных показателей	Верхняя граница, (max)	Разность, Δ
17 623	14	13 037	23 625	10 588
114 553	16	32 592	39 304	6 712
$(2^{17} - 1)$ 131 071	16	33 898	39 304	5 406
573 402 943	29	388 439	391 500	3 061
992 421 757	29	388 473	391 500	3 027
1 742 359 279	30	443 664	446 865	3 201
$(2^{31} - 1)$ 2 147 483 647	30	441 006	446 865	5 859

Разность Δ также была получена в ходе экспериментов, как число решений, которые совпадали с уже найденными, и полностью соответствует значениям из табл. 1 ($\Delta = \max$ минус число найденных различных показателей). Это очень важный момент, поскольку могли получиться решения, равные нулю, которые не учитывались. Но оказалось, что их ни при каких значениях p нет. Поэтому Δ — число линейно зависимых уравнений.

Теперь выполним теоретическую оценку. Линейный оператор отображения Ньютона (1) имеет квадратную матрицу следующего вида:

$$F = \begin{pmatrix} 1 & 0 & \dots & 0 & 1 \\ 1 & 1 & \dots & 0 & 0 \\ 0 & 1 & \dots & \vdots & 0 \\ \vdots & 0 & \dots & 1 & 0 \\ 0 & \vdots & \dots & 1 & 1 \end{pmatrix}. \qquad (8)$$

Таким образом, i-я строка матрицы T выражается через первую строку t_0 следующим образом:

$$t_i = t_0 F^i, \;\; 1 \le i \le k,$$

матрица F имеет размер $(k+1) \times (k+1)$; J-й элемент строки t_i получается умножением t_0 на j-й столбец матрицы F^i.

Поскольку одно уравнение составляется на основе двух элементов матрицы T, то два уравнения будут линейно зависимы в том и только в том случае, если существует нетривиальная, равная нулю линейная комбинация из четырех элементов матрицы T. Как было отмечено ранее, $T[i][j] = t_0 F^i[j]$, где $F^i[j] - j$-й столбец матрицы F^i. Поэтому линейная комбинация из четырех элементов матрицы T сводится к линейной комбинации из четырех столбцов матриц F^i. Рассмотрим эти матрицы.

$$F^2 = \begin{pmatrix} 1 & 0 & \dots & 1 & 2 \\ 2 & 1 & \dots & 0 & 1 \\ 1 & 2 & \dots & \vdots & 0 \\ \vdots & 1 & \dots & 1 & 0 \\ 0 & \vdots & \dots & 2 & 1 \end{pmatrix};$$

$$F^3 = \begin{pmatrix} 1 & 0 & \dots & 3 & 3 \\ 3 & 1 & \dots & 1 & 3 \\ 3 & 3 & \dots & 0 & 1 \\ 1 & 3 & \dots & \vdots & 0 \\ \vdots & 1 & \dots & 1 & 0 \\ 0 & \vdots & \dots & 3 & 1 \end{pmatrix};$$

$$F^4 = \begin{pmatrix} 1 & 0 & \dots & 6 & 4 \\ 4 & 1 & \dots & 4 & 6 \\ 6 & 4 & \dots & 1 & 4 \\ 4 & 6 & \dots & 0 & 1 \\ 1 & 4 & \dots & \vdots & 0 \\ \vdots & 1 & \dots & 1 & 0 \\ 0 & \vdots & \dots & 4 & 1 \end{pmatrix} \dots F^k.$$

Столбцы матриц получены следующим образом. Обозначим

$$\boldsymbol{s}_0 \qquad \boldsymbol{s}_1 \qquad \boldsymbol{s}_2 \qquad \boldsymbol{s}_3 \qquad s_4 \quad \dots \quad s_k$$

столбцы матрицы F. Тогда столбцы матриц F^2, F^3... соответственно выражаются следующим образом:

$$(s_0+s_1)\quad(\boldsymbol{s}_1+\boldsymbol{s}_2)\quad(s_2+s_3)\quad(s_3+s_4)\quad\ldots\quad(s_0+s_k),$$

$$(\boldsymbol{s}_0+2\boldsymbol{s}_1+\boldsymbol{s}_2)\quad(s_1+2s_2+s_3)\quad(s_2+2s_3+s_4)\quad\ldots\quad(s_1+2s_0+s_k),$$

$$(\boldsymbol{s}_0+3\boldsymbol{s}_1+3\boldsymbol{s}_2+\boldsymbol{s}_3)\quad(s_1+3s_2+3s_3+s_4)\quad\ldots\quad(s_2+3s_0+3s_1+s_k),$$

$$(s_0+4s_1+6s_2+4s_3+s_4)\quad\ldots\quad(s_3+4s_0+6s_1+4s_2+s_k)$$

и т.д.

Отметим, что возможны две типовые ситуации линейной зависимости между четырьмя столбцами с участием следующих элементов (соответствующие элементы выделены жирным):

$$s_0; s_1; s_2; (s_0+2s_1+s_2)$$

и

$$s_0; s_3; (s_1+s_2); (s_0+3(s_2+s_3)+s_3).$$

Каждая типовая ситуация встречается $(k+1)\times(k+1)$ раз при последовательном перемещении по строкам и столбцам ввиду симметричности применяемых операций. Таким образом, общее число линейно-зависимых столбцов, а следовательно, и число линейно зависимых уравнений составляет

$$\Delta_{\text{теор}} = 2(k+1)^2.$$

Видно, что экспериментальное Δ всегда больше теоретического (см. табл. 1). Это связано с тем, что при выводе теоретического числа не учитывались приведения по модулю. Они как раз и увеличивают число линейно-зависимых уравнений.

Построение последовательностей большего размера, многомерные матрицы S и T. Чтобы увеличить общее число элементов N, распространим преобразование (1) на случай трехмерных массивов.

Сначала строится квадратная матрица $S[0]$: $(k+1)\times(k+1)$ описанным способом построения матриц, с помощью отображения Ньютона, затем достраиваются k слоев квадратных матриц следующим образом:

$$S[i][j][w] = S[i-1][j][w] \bullet S[i-1][j+1][w],$$

$$1\le i\le k,\ \ 0\le j\le k-1;\ \ 0\le w\le k;$$

$$S[i][k][w] = S[i-1][0][w] \bullet S[i-1][k][w],\quad 1\le i\le k,\ 0\le w\le k.$$

Оценим, как и в случае квадратной матрицы, верхнюю границу числа различных решений, которые можно получить, решая уравнения, составленные из всевозможных пар элементов с разными коэффициентами при x:

$$\max = \frac{N(N-1)}{2} - \frac{k(k+1)}{2}(k+1)^2 = \left|N\sim k^3\right| \sim \left(k^6-k^4\right) \sim N^2.$$

Верхняя граница снова определяется как N^2, N — число элементов в массиве. Результаты экспериментов с трехмерными массивами представлены в табл. 2.

Таблица 2

Эксперимент с трехмерными массивами

Простой модуль	Число различных показателей	Верхняя граница (max)	Разность, Δ
115 351 339	92 916 501	193 444 524	100 528 023
573 402 943	264 890 324	364 095 000	99 204 676
724 321 489	280 872 622	364 095 000	83 222 378
992 421 757	298 719 546	364 095 000	65 375 454
1 324 324 513	369 115 716	443 290 080	74 174 364
1 548 023 513	377 469 238	443 290 080	65 820 842
1 742 359 279	383 145 258	443 290 080	60 144 822

Как следует из табл. 2 число линейно зависимых уравнений стало заметно больше, но по-прежнему число линейно независимых соответствует по порядку значению max.

Преобразование (1) можно распространить на случай массивов произвольной мерности естественным образом:

$$T[i][j]\dots[w] = T[i-1][j]\dots[w] \; + \; T[i-1][j+1]\dots[w] \pmod p;$$

$$1 \le i \le k, \;\; 0 \le j \le k-1; \;\; 0 \le w \le k;$$

$$T[i][k]\dots[w] = T[i-1][0]\dots[w] + T[i-1][k]\dots[w] \pmod p;$$

$$1 \le i \le k, \;\; 0 \le w \le k.$$

Обобщенный алгоритм логарифмирования в группе Z_p на основе отображения Ньютона и алгоритма Адлемана. Поскольку элементами группы Z_p являются числа, воспользуемся возможностью представления числа в виде произведения простых множителей. Аналогичную технику факторизации использует алгоритм Адлемана [3, 5]. Описанный далее подход немного видоизменяет этапы этого алгоритма за счет применения приведенной техники сравнения пар в матрице. Неизменным остается использование факторной базы, способ построения которой несколько другой. Проведем поэтапное описание алгоритма. Вопрос о таких параметрах алгоритма, как размер факторной базы, мерность матрицы S, рассмотрим несколько позднее, на этапе теоретических и экспериментальных оценок.

Требуется вычислить $\log_a b$ в мультипликативной группе Z_p. Обозначим порядок группы через $n = p - 1$.

1 этап. Составление факторной базы F. Для простых чисел p_i, начиная с 2, определяем порядок $\operatorname{ord}(p_i)$ в группе Z_p. Если $\operatorname{ord}(p_i) = n$, то добавляем это простое число в факторную базу: $F = F \cup p_i$. Иначе, переходим к следующему простому числу.

2 этап. Переход в подгруппу группы Z_p простого порядка r, в которой будем вычислять логарифм

$$h = b^{\frac{n}{r}}, \;\; g = a^{\frac{n}{r}}, \;\; g_1 = p_1^{\frac{n}{r}}, \;\; g_2 = p_2^{\frac{n}{r}}, \dots, \; g_{|F|} = p_{|F|}^{\frac{n}{r}}; \;\; p_i \in F,$$

исходим из того, что $p_i = a^{x_i}$, $b = a^x$; x, x_i — **неизвестные**, $i = 1 \div |F|$.

3 этап. Сначала представим p_i в виде $p_k^{y_{ik}}$, выразив y_{ik} через неизвестные x_i,

$$p_i = a^{x_i} = p_k^{y_{ik}} = (a^{x_k})^{y_{ik}} \Rightarrow y_{ik} = x_i x_k^{-1} \bmod n. \quad (9)$$

Поскольку $\forall k \ \mathrm{ord}\,(p_k) = n$, то $НОД\,(x_k,\ n) = 1$, поэтому существует x_k^{-1}.

Аналогичным образом представим b в виде $p_k^{z_k}$:

$$b = a^x = p_k^{z_k} = (a^{x_k})^{z_k} \Rightarrow z_k = x x_k^{-1} \bmod n. \quad (10)$$

Теперь переходим к третьему этапу алгоритма. Подразумевая представление $b = p_k^{z_k}$, после перехода в подгруппу получаем аргумент логарифма h, а из p_k — основание логарифма g_k. Затем, подавая значения h и g_k на вход соотношений (2), вычисляем многомерную матрицу S_k. Для всех значений $k = 1 \ldots |F|$ вычисляются матрицы S_k. Также вычисляется матрица S для h и g.

4 этап. Для всех вычисленных матриц проводятся последовательные деления чисел из матриц на числа из факторной базы до тех пор, пока они не перестанут делиться.

5 этап. На основе совпадающих элементов в матрицах после деления составляются уравнения следующим образом. Пусть совпали два числа из матриц S_m и S_n. Это может быть одна и та же матрица. Обозначим число из матрицы S_m через v, а из матрицы S_n через w. Пусть в ходе делений на числа из факторной базы число v делилось на числа из $F_1 \subseteq F$ с набором степеней $D_1 \subseteq \mathrm{N}^{|F|}$, а число w — на числа из $F_2 \subseteq F$ с набором степеней $D_2 \subseteq \mathrm{N}^{|F|}$. Теперь представим эти числа через определенные переменные:

$$v = p_m^{deg1}, \ \ deg1 = t\,[i]\,[j]\,[q] - \sum_{k:\ p_k \in F_1} (y_{km} \cdot D_{1k}) \bmod r,$$

где $t\,[i]\,[j]\,[q]$ — элемент матрицы T, $i, j, q-$ позиция v в матрице S_m, D_{1k} — степень p_k в разложении ($t\,[i]\,[j]\,[q] = c_1 z_m + c_2 \pmod r$, где коэффициенты c_1 и c_2 определены, исходя из позиции i, j, q, как описано выше). Учитывая (10) выразим z_m:

$$t\,[i]\,[j]\,[q] = c_1 x x_m^{-1} + c_2 \pmod r.$$

В итоге получаем

$$v = p_m^{deg1}, \ \ deg1 = c_1 x x_m^{-1} + c_2 - \sum_{k:\ p_k \in F_1} (\ y_{km} D_{1k}) \bmod r.$$

Аналогичным образом для w имеем

$$w = p_n^{deg2}, \ \ deg2 = c_3 x x_n^{-1} + c_4 - \sum_{k:\ p_k \in F_2} (\ y_{kn} D_{2k}) \bmod r.$$

Далее, зная, что $p_m = a^{x_m}$, $p_n = a^{x_n}$, запишем

$$v = a^{x_m \cdot deg1}, \ \ w = a^{x_n \cdot deg2}.$$

Учитывая, что $v = w$, приравниваем показатели и получаем следующее уравнение $\bmod r$:

$$x_m \left(c_1 x x_m^{-1} + c_2 \ - \sum_{k:\ p_k \in F_1} (\ y_{km} D_{1k}) \right) =$$

$$= x_n \left(c_3 x x_n^{-1} + c_4 \ - \sum_{k:\ p_k \in F_2} (\ y_{kn} D_{2k}) \right).$$

Подставив вместо y_{km} и y_{kn} соответствующие выражения, получаем:

$$x_m \left(c_1 x x_m^{-1} + c_2 - \sum_{k:\ p_k \in F_1} (x_k x_m^{-1} D_{1k}) \right) =$$

$$= x_n \left(c_3 x x_n^{-1} + c_4 - \sum_{k:\ p_k \in F_2} (x_k x_n^{-1} D_{2k}) \right) \pmod r.$$

Раскроем скобки:

$$c_1 x + c_2 x_m - \sum_{k:\ p_k \in F_1} (x_k D_{1k}) =$$

$$= c_3 x + c_4 x_n - \sum_{k:\ p_k \in F_2} (x_k D_{2k}) \pmod r. \quad (11)$$

Получили линейное уравнение от $|F|+1$ неизвестного.

Аналогичным образом составляются еще $|F|$ линейных уравнений вида (11). Таким образом, для решения логарифма необходима $(|F|+1)$ пара совпадающих чисел среди матриц $S, S_1, \ldots, S_{|F|}$.

6 этап. Решение системы из $(|F|+1)$ линейных уравнений с $(|F|+1)$ неизвестным и нахождение $x \pmod r$.

7 этап. В остальных подгруппах группы Z_p определяют $x \pmod{r_i}$, повторяя этапы 2...6 для нахождения искомого $x \pmod n$ (см. алгоритм Полига–Хэллмана [1]).

Сложность обобщенного алгоритма. Для оценки сложности алгоритма применим технику, которая использовалась для оценки сложности алгоритма Диксона для факторизации чисел [6]. В основе этой техники лежит теорема де Брейна–Эрдеша [7], которая утверждает следующее. Обозначим через $\Psi(n, M)$ — число натуральных чисел $a \le n$, в разложении которых присутствуют только числа из факторной базы $F = \{p_i : p_i < M\}$. Теорема утверждает, что

$$\Psi(n, M) = n u^{-u}, \quad u = \frac{\ln n}{\ln M}.$$

Как и в описании алгоритма, n — это порядок мультипликативной группы Z_p, наибольшее число в группе. В нашем случае, в отличие от алгоритма Диксона, требуется оценить не вероятность попадания в гладкое число (т.е. число, разложимое по факторной базе), а вероятность нахождения пары чисел, которые имеют общий делитель, взаимно простой со всеми числами из факторной базы. Тогда при попадании в такую пару после четвертого этапа алгоритма в матрицах найдется пара совпадающих чисел и составится уравнение. Рас-

считаем эту вероятность P и сравним ее с вероятностью P' попадания в гладкое число. Пусть W — множество чисел, меньших n, взаимно простых с числами из факторной базы F. $W = \{1 \le a < n \ : \ \forall p_i \in F, \ \ (a, p_i) = 1\}$. Тогда

$$P = \frac{2}{n^2} \sum_{d \in W} \frac{\Psi^2\left(\frac{n}{d}, M\right)}{2} = \\ = \frac{1}{n^2} \sum_{d \in W} \left(\frac{n}{d}\right)^2 \cdot \frac{1}{\left(\frac{\ln \frac{n}{d}}{\ln M}\right)^{\frac{2 \ln \frac{n}{d}}{\ln M}}} = \sum_{d \in W} \frac{1}{\left(d \left(\frac{\ln \frac{n}{d}}{\ln M}\right)^{\frac{\ln \frac{n}{d}}{\ln M}}\right)^2}. \quad (12)$$

Формула (12) получена следующим образом. Вероятность нахождения искомой пары чисел определяется как число пар, имеющих общий делитель из множества W, отнесенное к общему числу всевозможных пар $(n^2/2)$.

Вероятность попадания в гладкое число оценивается как

$$P' = \frac{1}{n} \Psi(n, M) = \frac{1}{\left(\frac{\ln n}{\ln M}\right)^{\frac{\ln n}{\ln M}}}. \quad (13)$$

В сумме (12) присутствует слагаемое с $d = 1$, равное $\left(P'\right)^2$. Тем самым получаем нижнюю оценку для P:

$$\left(P'\right)^2 < P.$$

Чтобы получить более точную оценку, можно воспользоваться очевидным неравенством

$$\left(\frac{\ln \frac{n}{d}}{\ln M}\right)^{\frac{\ln \frac{n}{d}}{\ln M}} < \left(\frac{\ln n}{\ln M}\right)^{\frac{\ln n}{\ln M}} = \frac{1}{P'} \text{ при } 1 < d < n.$$

Заменив в (12) знаменатель с учетом неравенства, получим

$$\left(\left(P'\right)^2 + \left(P'\right)^2 \sum_{d \in W, \ \ d \ne 1} \frac{1}{d^2}\right) < P.$$

Как известно, ряд Дирихле $\sum_{k=1}^{\infty} \frac{1}{k^2}$ сходится и дает в сумме число, меньшее единицы (а именно, $\frac{\pi^2}{6} - 1$). Поэтому наша выборочная сумма будет заведомо меньше единицы. Можно в итоге записать

$$P > \left(P'\right)^2 (1 + C), \ \ C < \left(\frac{\pi^2}{6} - 1\right).$$

Теперь оценим значение ограничения факторной базы M, необходимой, чтобы вычислить логарифм с помощью обобщенного алгоритма, при условии, что используются матрицы размера dim. Порядок подгруппы, в которой

осуществляется логарифмирование, обозначим через r. Поскольку количество простых чисел в факторной базе можно считать равным $h = \dfrac{M}{\ln M}$, то общее число элементов во всех h dim-мерных матрицах будет равно

$$N = h\,(\ln r + 1)^{\dim} = \frac{M}{\ln M}\,(\ln r + 1)^{\dim}.$$

Число всевозможных пар элементов в матрицах равно $N^2/2$. Для простоты считаем, что все элементы в матрицах различны, к случаю совпадения чисел и наличия линейных зависимостей мы вернемся позднее. Поскольку необходимо составить $h+1$ уравнение, необходимо столько же совпадений после четвертого этапа. Чтобы попасть в одну совпадающую пару, число всевозможных пар элементов должно быть $\sim 1/P$. Чтобы попасть в $h + 1$ совпадающую пару, число всевозможных пар должно быть не меньше $(h+1)/P$. Поэтому

$$\frac{N^2}{2} > \frac{h+1}{P};\to \left(\frac{M}{\ln M}\right)^2 (\ln r + 1)^{2\dim} > \frac{2M}{\ln M\,(P')^2\,(1+C)};$$

$$\frac{M}{\ln M}\,(\ln r)^{2\dim} > \frac{2}{(1+C)}\left(\frac{\ln n}{\ln M}\right)^{2\frac{\ln n}{\ln M}}. \qquad (14)$$

Оценим M в стандартных обозначениях субэкспоненциальной сложности:

$M = \exp\left(C'\,(\ln n)^q\,(\ln\ln n)^{1-q}\right)$, $\quad 0 < q < 1$, $\quad C'$ — константа.

Тогда неравенство (14) запишется следующим образом:

$$\exp\left(C'\,(\ln n)^q\,(\ln\ln n)^{1-q} + 2\dim\cdot\ln\ln r - \ln\ln M\right) >$$

$$> \exp\left(\frac{2\ln n}{\ln M}\cdot\ln\left(\frac{\ln n}{\ln M}\right)\right);C' \to \left((\ln n)^q\,(\ln\ln n)^{1-q} + 2\dim\cdot\ln\ln r - \right.$$

$$\left. - q\ln\ln n\right) \;>\; \left(\frac{2\ln n}{C'\,(\ln n)^q\,(\ln\ln n)^{1-q}}\right)(\ln\ln n - q\ln\ln n);\to$$

$$\left(C'\,(\ln n)^q\,(\ln\ln n)^{1-q} + 2\dim\cdot\ln\ln r - q\ln\ln n\right) >$$

$$> \left(\frac{2\,(1-q)}{C'}\,(\ln n)^{1-q}\,(\ln\ln n)^q\right). \qquad (15)$$

Очевидно, что минимальное q, при котором выполняется неравенство (15), это $q = 1/2$.

Самая трудоемкая часть алгоритма — это этап 4, поскольку каждый элемент из матриц будет последовательно делиться на числа из факторной базы. Поэтому итоговая сложность алгоритма оценивается как

$$S \approx O\,(hN) = O\left(\left(\frac{M}{\ln M}\right)^2 (\ln r + 1)^{\dim}\right) =$$

$$= O\left(\exp\left(2\,(\ln M - \ln\ln M) + \dim\cdot\ln\ln r\right)\right) =$$

$$= O\left(\exp\left(2\left(C'\left(\ln n\right)^{q}\left(\ln\ln n\right)^{1-q} - q\ln\ln n\right) + \dim\ln\ln r\right)\right) =$$

$$= O\left(\exp\left(2C'\left(\ln n\right)^{q}\left(\ln\ln n\right)^{1-q} + \dim\ln\ln r - 2q\ln\ln n\right)\right). \quad (16)$$

Теперь главный вопрос заключается в том, что нельзя ли за счет $\dim$ подобрать как можно меньший коэффициент C' так, чтобы выполнялось неравенство (15). Пусть $\dim = C''(\ln n)^{q}$. Тогда неравенство (15) запишется как

$$\left(\left(C' + 2C''\frac{\ln\ln r}{\left(\ln\ln n\right)^{1-q}}\right)\left(\ln n\right)^{q}\left(\ln\ln n\right)^{1-q} - q\ln\ln n\right) >$$

$$> \left(\frac{2\left(1-q\right)}{C'}\left(\ln n\right)^{1-q}\left(\ln\ln n\right)^{q}\right); \to \left(C' + 2C''\frac{\ln\ln r}{\left(\ln\ln n\right)^{1-q}}\right) > \frac{2\left(1-q\right)}{C'}.$$

Второе условие на коэффициенты вытекает из минимальности коэффициента при $\left(\ln n\right)^{q}\left(\ln\ln n\right)^{1-q}$ в выражении (16):

$$S = O\Bigg(\exp\Bigg(\left(2C' + C''\frac{\ln\ln r}{\left(\ln\ln n\right)^{1-q}}\right)\times$$

$$\times\left(\ln n\right)^{q}\left(\ln\ln n\right)^{1-q} - 2q\ln\ln n\Bigg)\Bigg).$$

Таким образом, получаем соотношения на коэффициенты

$$\begin{cases} \left(C' + 2C''\dfrac{\ln\ln r}{\left(\ln\ln n\right)^{1-q}}\right) > \dfrac{2\left(1-q\right)}{C'}; \\ \min\limits_{C',C''}\left(2C' + C''\dfrac{\ln\ln r}{\left(\ln\ln n\right)^{1-q}}\right) = C. \end{cases}$$

При $q = 1/2$ минимум $C = \sqrt{3} \approx 1{,}73$ достигается при $C' = 1/\sqrt{3}$ и соответственно $C'' = \dfrac{1}{\sqrt{3}}\dfrac{\left(\ln\ln n\right)^{1-q}}{\ln\ln r}$. Таким образом, сложность оптимального варианта алгоритма

$$S = O\left(\exp\left(\sqrt{3}\sqrt{\ln n\cdot\ln\ln n} - \ln\ln n\right)\right) = L_n\left[1/2, \sqrt{3}\right].$$

Соответствующие параметры алгоритма следующие:

$$M = \exp\left(\sqrt{\frac{\ln n\ \cdot\ln\ln n}{3}}\right),$$

$$\dim = \sqrt{\frac{1}{3}\frac{\ln n\cdot\ln\ln n}{\ln\ln r}},$$

где M — ограничение факторной базы, $\dim$ — размер матриц в алгоритме.

При этом объем необходимой памяти при наименьшей вычислительной сложности составляет

$$O(N) = O\Bigg(\exp\Bigg(\Bigg(C' + C'' \frac{\ln\ln r}{(\ln\ln n)^{1-q}} \Bigg) \times$$

$$\times (\ln n)^q (\ln\ln n)^{1-q} - q\ln\ln n \Bigg) \Bigg) =$$

$$= O\left(\exp\left(\frac{2}{\sqrt{3}} \sqrt{\ln n \cdot \ln\ln n} - 0{,}5\ln\ln n \right) \right) = L_n\left[1/2, 1{,}15\right].$$

В основе приведенного алгоритма лежит сравнение двух элементов из набора массивов после их приведения по факторной базе. Возникает закономерный вопрос: дает ли выигрыш в сложности такое разбиение на пары? Ответ на этот вопрос можно получить, оценив сложность другого алгоритма, в котором после этапа приведения по факторной базе (этап 4) проводится поиск элементов, равных единице, т.е. разложившихся по факторной базе. Каждый такой элемент также даст одно уравнение. Чтобы оценить сложность, запишем аналогичное (14) неравенство, но уже для данного случая:

$$N > \frac{h+1}{P'} \;\rightarrow\; (\ln r + 1)^{\dim} > \left(\frac{\ln n}{\ln M} \right)^{\frac{\ln n}{\ln M}}.$$

Проведя преобразования, аналогичные описанным ранее, получаем сложность алгоритма без разбиения на пары $L_n\left[1/2, 2\right]$. Следовательно, сравнение двух элементов на совпадение после приведения по факторной базе дает некоторый выигрыш, который особенно заметен на небольших n.

При оценке сложности считали, что в матрицах после этапа их построения (этап 3) не окажется совпадающих элементов. Но, вообще говоря, совпадения возможны. В случае совпадения каких-либо двух элементов попрежнему можно составить уравнение вида (11), из которого будут исключены суммы по делителям из факторной базы. Следует иметь в виду, что некоторые из полученных уравнений могут оказаться линейно зависимыми. Подсчитать число линейно зависимых уравнений для многомерных матриц пока не представляется возможным из-за сложности теоретических выкладок, но вполне допустима экспериментальная оценка, аналогичная проведенной для трехмерных матриц. Следовательно, в алгоритме присутствует некоторая доля эвристики, обусловленная характеристиками применяемого отображения для генерации последовательности чисел. Был предложен лишь один из способов на основе отображения Ньютона над многомерными матрицами и показано, что для маломерных матриц число линейно зависимых уравнений незначительно. На момент написания статьи эксперименты с матрицами размера больше 3 не проводились.

Заключение. В настоящее время известны алгоритмы дискретного логарифмирования для числовых полей $GF(p)$, которые имеют сложность $L_n\left[1/3, \text{const}\right]$, $n = p - 1$, это алгоритмы на основе решета числового поля [8, 9]. Все они очень эффективны для больших n, асимптотически стремящихся к бесконечности. Для меньших n ($n < 10^{100}$) существуют более эффективные алгоритмы, например алгоритм Копперсмита – Одлизко – Шреппеля (COS) [10], который имеет сложность $L_n\left[1/2, 1\right]$.

Приведенный алгоритм хуже по значению константы $L_n\left[1/2, \sqrt{3}\right]$, но, тем не менее, субэкспоненциальный. Также недостатком является требование к памяти $L_n\left[1/2, 1{,}15\right]$. В то же время он прост, не требует каких-либо специальных стратегий и дополнительных алгоритмов, основан на элементарных операциях.

Возможно, описанные подходы найдут применение при логарифмировании в некоторых других группах, например, группе точек эллиптической кривой. Выделение общих "множителей" в паре точек вполне может иметь место. А потому может оказаться полезной техника построения последовательностей на отображении Ньютона.

Автор благодарит А.Н. Лебедева и Г.А. Лебедева за содействие в разработке алгоритмов и написании настоящей работы, полезные замечания и обсуждения.

ЛИТЕРАТУРА

1. *Pohlig S.C.*, *Hellman M.E.* An Improved Algorithm for Computing Logarithms Over GF(p) and its Cryptographic Significance // IEEE, Transactions on Information Theory. 1978. Vol. 1. No. 24. P. 106–110.
2. *Shanks D.* The infrastructure of a real quadratic field and its applications // Proceedings of the Number Theory Conference. University of Colorado, Boulder. 1972. P. 217–224.
3. *Глухов М.М.*, *Круглов И.А.*, *Пичкур А.Б.*, *Черемушкин А.В.* Введение в теоретико-числовые методы криптографии. СПб.: Лань, 2011. 400 с.
4. *Pollard J.M.* Monte Carlo methods for index computation (mod p) // Math. Comp. 1978. No. 32. P. 918–924.
5. *Adleman L.M.* A subexponentional algorithm for the discrete logarithm problem with applications to cryptography. Proceedings of the IEEE 20th Annual Symposium on Foundations of Computer Science. 1979. P. 55–60.
6. *Dixon J.D.* Asymptotically fast factorization of integers // Math. Comp. 36 (153). 1981. P. 255–260.
7. *De Bruijn N.G.*, *Erdös P.* A combinatioral [sic] problem // Indagationes Mathematica. Vol. 10. 1948. P. 421–423.
8. *Василенко О.Н.* Теоретико-числовые алгоритмы в криптографии. М.: МЦНМО. 2003. 328 с.
9. *Матюхин Д.В.* Об асимптотической сложности дискретного логарифмирования в поле GF(p) // Дискретная математика. 2003. Т. 15. Вып. 1. С. 28–49.
10. *Coppersmith D.*, *Odlyzko A.M.*, *Schroeppel R.* Discrete logarithms in GF(p). Algorithmica. 1986. Vol. 1. P. 1–15.

REFERENCES

[1] Pohlig S.C., Hellman M.E. An Improved Algorithm for Computing Logarithms Over GF(p) and its Cryptographic Significance. *IEEE, Transactions on Information Theory*, 1978, vol. 1, no. 24, pp. 106–110.
[2] Shanks D. The infrastructure of a real quadratic field and its applications. Proc. Number Theory Conference. Un. of Colorado, Boulder, 1972. pp. 217–224.
[3] Glukhov M.M., Kruglov I.A., Pichkur A.B., Cheremushkin A.V. Vvedenie v teoretiko-chislovye metody kriptografii [Introduction to number-theoretic methods of cryptography]. SPb., Lan' Publ., 2011. 400 p.

[4] Pollard J.M. Monte Carlo methods for index computation (mod p). *Math. Comp.*, 1978, vol. 32, no. 143, pp. 918–924.
[5] Adleman L.M. A subexponentional algorithm for the discrete logarithm problem with applications to cryptography. *Proc. IEEE 20th Annual Symp. on Foundations of Computer Science*. 1979. pp. 55–60.
[6] Dixon J.D. Asymptotically fast factorization of integers. Math. Comp., 1981, vol. 36, no. 153, pp. 255–260.
[7] De Bruijn N.G., Erdős P. A combinatioral [sic] problem. *Indagationes Mathematica*, 1948, vol. 10, pp. 421–423.
[8] Vasilenko O.N. Teoretiko-chislovye algoritmy v kriptografii [Number-theoretic algorithms in cryptography]. Moscow, MTsNMO Publ., 2003. 328 p.
[9] Matyukhin D.V. On the asymptotic complexity of discrete taking the logarithm in the field GF(p) *Diskretnaya matematika* [Discrete Mathematics and Applications], 2003, vol. 15, iss. 1, pp. 28–49 (in Russ.).
[10] Coppersmith D., Odlyzko A.M., Schroeppel R. Discrete logarithms in GF(p). *Algorithmica*, 1986, vol. 1, pp. 1–15.

Статья поступила в редакцию 27.12.2013

Дмитрий Евгеньевич Островский — студент 6-го курса кафедры “Информационная безопасность” МГТУ им. Н.Э. Баумана.
МГТУ им. Н.Э. Баумана, Российская Федерация, 105005, Москва, ул. 2-я Бауманская, д. 5.

D.E. Ostrovskii — 6-year student of “Information Security” department of the Bauman Moscow State Technical University.
Bauman Moscow State Technical University, Vtoraya Baumanskaya ul. 5, Moscow, 105005 Russian Federation.