УДК 681.3.06

О.А. МЕЛЬНИКОВА, канд. техн. наук, О.Н. ТОВМА

АНАЛИЗ ВОЗМОЖНОСТЕЙ УМЕНЬШЕНИЯ ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ ЭЦП ДСТУ 4145-2002 ПРИ ИСПОЛЬЗОВАНИИ НОРМАЛЬНОГО БАЗИСА

Нормальный и полиномиальный базисы рекомендуются к использованию во многих стандартах, связанных с криптографией на эллиптических кривых, например ДСТУ 4145-2002 [1], IEEE P1363 [2]. Аппаратная реализация арифметики нормального базиса даёт большие преимущества в сравнении с аппаратной реализацией полиномиального базиса и программной реализацией нормального базиса. Но, так как аппаратная реализация является материально затратной и часто возникает необходимость совместимости реализации стандартов различных стран, поэтому применяется программная эмуляция. Поэтому появляются задачи более эффективной программной реализации преобразований в нормальном базисе.

Представление в нормальном базисе имеет вычислительное преимущество при возведении в квадрат элемента. Данная операция представляет собой циклический сдвиг. Одним из наиболее актуальных вопросов арифметики конечных полей остается уменьшение вычислительной сложности таких операций, как умножение и инверсия. Но так как инверсия является более трудоемкой операцией, то вопрос оптимизации инверсии стоит более остро.

Чаще всего применяется класс нормальных базисов, называемый Гауссовскими нормальными базисами (ГНБ), поскольку он обладает меньшей вычислительной сложностью за счет оптимизации правил умножения. Тип 1 и 2 ГНБ имеют самые эффективные правила умножение среди всех нормальных базисов.

Проведенный анализ научных работ показал, что в настоящее время чаще всего рекомендуются следующие алгоритмы (по именам их авторов):

1) Wang [3];
2) Itoh, Tsujii [4];
3) Feng [5];
4) Chang [6];
5) Takagi, Yoshiki [7].

В ДСТУ 4145-2002 и IEEE P1363 рекомендуется использовать алгоритм [4], однако появились более эффективные варианты. Для перечисленных алгоритмов был проведен теоретический и экспериментальный анализ их характеристик.

Алгоритм [3], как и последующие, использует следствие из малой теоремы Ферма.

Из малой теоремы Ферма следует, что $\forall a \neq 0, a \in \mathrm{GF}\left(2^m\right) \exists a^{-1} = a^{2^m-2}$, где m – степень расширения поля. Как известно, стандартом [1] рекомендуются нечетные, более того, простые m: 173, 179, 191, 233, 431.

Поскольку $2^m - 2 = 2^1 + 2^2 + \ldots + 2^{m-1}$. a^{-1} представляется как $a^{-1} = a^{2^m-2} = a^{2^1} \times a^{2^2} \times \ldots \times a^{2^{m-1}}$.

Алгоритм [3] требует m-2 умножений.

Itoh и Tsujii [4] предложили модернизацию алгоритма [3], а именно – разложение степени 2^m-2 следующим образом: $a^{-1} = a^{2^m-2} = \left(a^{2^{m-1}-1}\right)^2$.

Далее степень 2^m-1 раскладывается:

1) при нечетном m

$$a^{2^{m-1}-1} = 2^{\left(2^{(m-1)/2}-1\right)\left(2^{(m-1)/2}+1\right)},$$

2) при четном m

$$a^{2^{m-1}-1} = 2^{\left(2^{(m-2)/2}-1\right)\left(2^{(m-2)/2}+1\right)+1}.$$

Алгоритм требует $q(m\text{-}1)+w(m\text{-}1)\text{-}1$ умножений, где $w(m)$ – вес Хемминга и $q(m\text{-}1) = \lfloor \log_2(m-1) \rfloor$.

Алгоритм, предложенный Feng [5], по вычислительной сложности практически эквивалентен алгоритму [4].

Алгоритм [5] отличается лишь тем, что он вычисляет квадратные корни каждую итерацию, а потом возводит в квадрат 2^{m-m_0}, в то время как алгоритм [4] выполняет возведение в квадрат каждую итерацию. И оценка вычислительной сложности алгоритма [5] составляет $(q(m) + w(m))$ умножений, где $q(m) = \lfloor \log_2(m) \rfloor$.

Алгоритм [6] основывается на следующем разложении: $m-1 = s \times t$. При этом элемент a^{-1} может быть выражен следующим образом:

$$a^{-1} = (a^{(2^{s-1}-2)})^{((2^s)^{t-1}+(2^s)^{t-2}+\ldots+(2^s)^1+(2^s)^0)}.$$

$a^{(2^{s-1}-2)}$ может быть вычислено с помощью алгоритма [4] с заменой m-1 на s.

Он требует $(q(s)+w(s)\text{-}2)+(q(t)+w(t)\text{-}2)$ умножений, где $m-1 = s \times t$. Данный алгоритм эффективен, однако область его применения ограничена (он не может быть использован, когда m-1 является простым числом). Однако это ограничение не является критичным для реализации отечественного стандарта [1] (рекомендовано использовать простые m, т.е. m-1 гарантированно непростое число).

Количество умножений уменьшается по сравнению с алгоритмом [4]. Например, для $m = 233$ [1], количество умножений для алгоритма [6] равно 8, а для алгоритма [4] количество умножений равно 10. Необходимо отметить, что количество умножений зависит от способа разложения числа m-1 на сомножители (когда возможны различные варианты) m-1 может быть разложено на несколько комбинаций чисел. Кроме того, при некоторых значениях m, количество умножений в оценке вычислительной сложности алгоритма не всегда меньше, чем для алгоритмов [4, 5] (зависит от веса сомножителей s, t). Однако это не относится к вариантам из [1].

Takagi и др. [7] предложили усовершенствование алгоритма [6], которое основывается на следующем типе разложения: $m-1 = s \times t + h$. Так как

$$2^m - 2 = 2^{m-1} + 2^{m-1} - 2 = 2^{m-1} + 2^{m-2} + \ldots + 2^{m-h} - 2, \text{ то}$$
$$a^{-1} = a^{2^m-2} = a^{2^{m-1}} \times a^{2^{m-2}} \times \ldots \times a^{2^{m-h}-2},$$

где $a^{2^{m-i}}$ вычисляется при помощи i-битового циклического сдвига влево. Таким образом, a^{-1} получают из $a^{2^{m-h}-2}$ при помощи h умножений, а значение $a^{2^{m-h}-2}$ вычисляется при помощи алгоритмов [4] или [6] (при замене m на m-h). Далее $s \times t$ можно разложить на более мелкие сомножители, и для них использовать описанный выше подход повторно.

В общем случае, при представлении $m-1 = \prod_{j=1}^{k} s_j + h$ и вычислительная сложность оценивается как $\sum_{j=1}^{k}(q(s_j) + w(s_j) - 2) + h$.

Очевидно, что количество умножений зависит от способа разложения. Может существовать несколько разложений, которые уменьшают количество умножений. В таком случае, лучше всего адаптировать наипростейшее разложение, то есть разложение с наименьшим количеством компонентов, с целью сделать контроль вычисления a^{-1} проще. Далее разложение, минимизирующее количество требуемых умножений и состоящее из минимально возможного количества компонентов, называется «оптимальным разложением». Оптимальных разложений для одного и того же числа может быть несколько.

При нахождении оптимального разложения руководствуются следующими утверждениями:

1) При m-1 = 2^n оптимальным разложением является само значение m-1 и количество умножений в оценке вычислительной сложности равно n.

2) При $m-1=2^n+2^{n'}$ ($n>n'$) вес $w(m\text{-}1)=2$, оптимальным разложением является само значение m-1 и количество требуемых умножений $n+1$.

3) При $m-1=2^{n'}s+h$, где s – нечетное, наименьшее количество требуемых умножений по разложению $m-1$ как $\prod_{j=1}^{k}s_j+h$ равно $n'+h$ плюс количество требуемых умножений для оптимального разложения s.

При представлении m-1 = $\prod_{j=1}^{k}s_j+h$, если s_1 далее не раскладывается, то количество умножений в оценке вычислительной сложности не менее $\sum_{j=1}^{k}q(s_j)+h\ge q(m-1)+h$, т.к. $w(s_j)\ge 2$. При дальнейшем использовании оптимального разложения s_1 требуется не менее $q(s_1)$ умножений, т. е. количество умножений при оптимальном разложении m-1 не менее $q(m-1)+h$.

4) В оптимальном разложении m-1, выбирают остаток $h < w(m\text{-}1)\text{-}2$.

В табл. 1 обобщены теоретические оценки вычислительной сложности рассмотренных алгоритмов в количестве умножений элементов поля.

Таблица 1

Алгоритм	Сложность, кол. умножений
Wang [3]	m-2
Itoh, Tsujii [4]	$q(m\text{-}1)+w(m\text{-}1)-1$
Feng [5]	$q(m)+w(m)$
Chang [6]	$q(s)+w(s)+q(t)+w(t)$-4
Takagi, Yoshiki [7]	$\sum_{j=1}^{k}(q(s_j)+w(s_j)-2)+h$

При заданном m мы можем найти все варианты оптимальных разложений m-1. При этом стоит заметить, что можно использовать предложения для нахождения оптимального разложения первого сомножителя s_1. Проблема нахождения оптимального разложения не является критичной, потому как мы должны решать её, только когда мы выбираем m

В табл. 2 приведены результаты предварительного просчета количества умножений при использовании всех возможных разложений для значений m, рекомендованных в [1].

Таблица 2

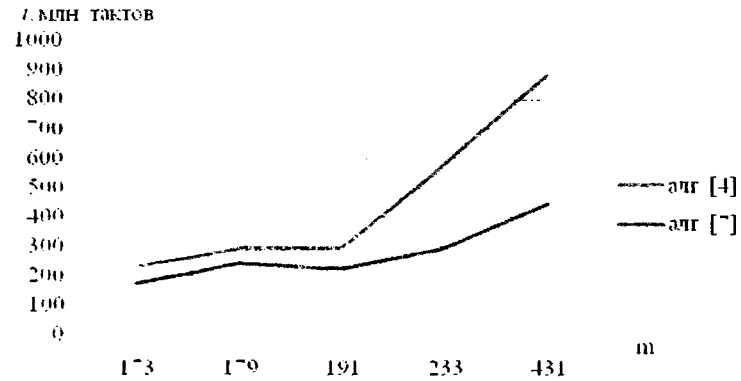
m	Факт.-ция m-1 (для алг. [7])	Варианты разложений m-1 (для алг. [6])			Количество умножений					
					[4]	[5]	[6]			[7]
173	2×2×43	4×43	2×86	-	10	11	7	5	-	6
179	2×89	2×89	-	-	10	11	9	-	-	9
191	2×5×19	10×19	38×5	95×2	12	13	8	8	10	7
233	2×2×2×29	4×58	8×29	2×116	10	11	8	8	8	8
431	2×5×43	10×43	2×215	5×86	13	13	9	11	10	9

Очевидно, что при любых вариантах разложения наименьшей вычислительной сложностью обладают алгоритмы [6, 7], причём для большинства рекомендованных [1] полей оптимален алгоритм [7], однако желательно иметь возможность использовать оба варианта (например, алгоритм [6] для поля с расширением m = 173).

При проведении экспериментальных исследований рассматривались алгоритмы [4, 7], так как алгоритм [4] является рекомендованным ДСТУ 4145-2002 и IEEE P1363, а алго-

ритм [7] характеризуется наилучшими теоретическими оценками вычислительной сложности.

На рисунке приведены соответствующие результаты экспериментальных исследований, а именно оценки среднего количества тактов процессора t при рекомендованных [1] значениях степени m.



Анализ существующих алгоритмов инверсии в нормальном базисе выявил, что алгоритмы [6, 7] показывают лучшие вычислительные характеристики по сравнению с рекомендованным стандартами алгоритмом [4]. Для значений m, рекомендованных [1], теоретические оценки средней вычислительной сложности алгоритмов [6, 7] меньше соответствующей оценки алгоритма [4] приблизительно на 29%.

Экспериментальные исследования показали, что вычислительная сложность алгоритма [7] меньше, чем у алгоритма [4], в среднем на 37,5%. Таким образом, именно алгоритм [7] может быть рекомендован к использованию при реализации операции инверсии в нормальном базисе. Однако желательно иметь возможность программного выбора между реализациями вариантов алгоритмов [6, 7] в зависимости от текущего используемого поля. А также необходимы более детальные экспериментальные исследования возможных наиболее эффективных рекурсивных версий реализации данных методов.

Список литературы: 1. *ДСТУ 4145 – 2002.* Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевіряння: Введ. 1.07.2003. – К. : Державний комітет України з питань технічного регулювання та споживчої політики, 2003 р. – 36 с. 2. *IEEE P1363* / D9 (Draft Version 9). Standard specifications for public key cryptography. Annex A (informative). Number – theoretic background, 1999. – 170 p. 3. *C.C.Wang*, T.K.Truong, H.M.Shao, L.J.Deutsch, J.K.Omura, I.S.Reed. VLSI architecture for computing multiplications and inverses in GF(2^m). IEEE Trans. Computers, vol. 34, no. 8, pp. 709-716, 1985. 4. *T.Itoh, S.Tsujii.* A fast algorithm for computing multiplicative inverses in GF(2^m) using normal basis. Information and Computing, vol. 78, pp. 171-177, 1988. 5. *G.L.Feng.* A VLSI architecture for fast inversion in GF(2^m). IEEE Trans. Computers, vol. 38, pp.1383-1386, 1989. 6. *T.Chang*, E.Lu, Y.Lee, Y.Leu, H.Shyu. Two algorithms for computing multiplicative inverses in GF(2^m) using normal basis, accepted by Information Processing Letters. 7. *N.Takagi. Y.Yoshiki*, K.Takagi. A fast algorithm for multiplicative inversion in GF(2^m) using normal basis // IEEE Transactions on Computers, vol. 50, no. 5, pp. 394-398, 2001.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 15.08.2011