КРИПТОГРАФИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ

УДК 004 056 55

Е.Г. КАЧКО, канд. техн. наук, Д.С. БАЛАГУРА, канд. техн. наук,
К.А. ПОГРЕБНЯК, канд. техн. наук, Ю.И. ГОРБЕНКО, канд. техн. наук

ИССЛЕДОВАНИЕ МЕТОДОВ ВЫЧИСЛЕНИЯ ИНВЕРСИИ В АЛГОРИТМЕ NTRU

Операция инверсии используется для генерации личных ключей шифрования и цифровой подписи и является вычислительно сложной операцией. Цель данной работы - исследование различных методов вычисления инверсии, их оптимизация с учетом возможностей параллельных вычислений. Исследуются алгоритмы [1 – 3] с точки зрения их возможной параллелизации и сравниваются по вычислительной сложности.

В алгоритме NTRU используется кольцо усеченных полиномов [1] $Z(X)/(X^N-1)$, которые содержат не более чем N целочисленных элементов. Значения коэффициентов полинома обычно ограничены. Кольцо полиномов, коэффициенты которых меньше p, обозначается как: $(Z/pZ)[X]/(X^N-1)$. Принято обозначать полиномы, принадлежащие такому кольцу, как $a(X), b(X), \ldots$ Инверсия полинома $a(X)$ в кольце $(Z/pZ)[X]/(X^N-1)$ – операция определения полинома $b(X)$, такого, что $a(X)*b(X)=1$ в этом же кольце.

Рассмотрим алгоритм инверсии, предложенный в [1]

Алгоритм 1 (А1). Вычисление инверсии.

Вход. Полином $a(X)$, порядок полинома N, модуль q [1]

Выход. Статус (Успех или ошибка), обратный элемент $b(X)$ (в случае успешного статуса).

1. Приведение $a(X)$ в элемент кольца $(Z/2Z)[X]/(X^N-1)$ (все коэффициенты полинома приводятся по модулю $p=2$, отрицательные значения заменяются положительными $(2+a[i]\%2)$.
2. Используя расширенную теорему Эвклида для полиномов $a(X)$ и X^N-1, находим наибольший общий делитель $d(X)$, и коэффициенты $u(X)$, $v(X)$ Диофантового уравнения $u(X)*a(X)+v(X)*(X^N-1)=d(X)$ для модуля $p=2$.
3. Если наибольший общий делитель $d(X)$ не равен 1 ($d[0]\neq\pm1$ или $d[1], d[2], \ldots, d[N-1]\neq 0$) – обратного элемента нет, статус равен *Ошибка* и *Выход*.
4. Вычислить $b(X)=(d(X)^{-1}(\mathrm{mod}\ p)*u(X))\ \mathrm{mod}\ p$.
5. Для всех модулей p = 4, 16, 256, 65536 ($p = p^2$).

5.1. Приводим $a(X)$ по модулю p (отрицательные значения заменяем значениями $p + a[i]$).

5.2 Вычисляем новое значение $b(X)=(2*b(X)-a(X)*b(X)^2)\ \mathrm{mod}\ p$.

6. Значение *b(x)* приводим по модулю $q = 2048$.
7. Статус равен *Успешно*.
8. Выход.

Алгоритмы, рассмотренные далее (А2, А3, А4), предназначены для вычисления инверсии для модуля 2 (пункты 2 – 4 алгоритма А1).

[1] Для всех полиномов в [1] q = 2048.

Алгоритм 2 (А2)

Вход. Порядок p[2], полиномы $a(X)$, $c(X)$[3].

Выход. Полиномы $d(X), u(X), v(X)$, удовлетворяющие уравнению: $a(X)*u(X)+c(X)*v(X)=d(X)$ в кольце.

1. Если $c(X)$, то $u(X)=1,\ v(X)=0,\ d(X)=a(X)$.
2. $u(X); d(X)=a(X); v_1(X)=0;\ v_3(X)=c(X)$.
3. Пока ($v_3(X)\neq 0$) выполнить:

3.1. $q(X)=d(X)\%v_3(X)$[4]; $t_3(X)=d(X)\%v_3(X)$[5];

3.2. $t_1(X)=u(X)-q(X)*v_1(X)$;

3.3. $u(X)=v_1(X)$;

3.4. $d(X)=v_3(X)$;

3.5. $v_1(X)=t_1(X)$;

3.6. $v_3(X)=t_3(X)$;

4. $v(X)=(d(X)-a(X)*u(X))/b(X)$.

Рассмотрим оптимизацию А2 как составной части А1.

1. Полином $c(X)=X^N-1$ не может быть равным 0, поэтому первый пункт алгоритма можно опустить.
2. При выполнении первой итерации цикла в операции деления $a(X)/c(X)$ отметим что порядок $a(X)$ меньше порядка $c(X)$, следовательно, на первом шаге $q(X)=0$, $t_3(X)=a(X)$. Таким образом, первую итерацию можно опустить целиком. Для второй итерации использовать деление $c(X)/a(X)$, а в качестве начального значения $d(X)$ использовать значение $a(X)$.
3. При вычислении $t_1(X)$ после очередного деления $u(X)=0$, а $v_1(X)=1$, т.е. $t_1(X)=-q(X)$, а с учетом модуля 2 $t_1(X)=q(X)$.
4. Так как все коэффициенты и результат вычисления по модулю 2, операцию умножения (шаг 3.2) следует заменить операциями сложения по модулю 2.
5. Значение $v(X)$ не используется далее, его вычисление можем опустить (пункт 4 алгоритма 2).
6. Везде, где возможно, следует использовать SSE операции.

Алгоритм 2 после оптимизации (А2'):

1. $d(X)=a(X); u(X)=1;\ q(X)=c(X)/a(X); v_3(X)=c(X)\%a(X)$.
2. $v_1(X)=q(X)$.
3. Пока ($v_3(X)\neq 0$) выполнить:

3.1. $q(X)=d(X)/v_3(X);\ \ t_3(X)=d(X)\%_3(X)$;

3.2. $t_1(X)=u(X)\wedge q(X)*v_1(X);\ \ u(X)=v_1(X);\ d(X)=v_3(X)$;

3.4. $v_1(X)=t_1(X);\ v_3(X)=t_3(X)$.

В одной строчке задаются операторы, которые можно выполнять параллельно.

Результаты, полученные после предложенной оптимизации, приведены в табл.1.

[2] Значение p=2.

[3] Последнему полиному соответствует полином $x^N - 1$.

[4] Деление нацело, дробная часть отбрасывается.

[5] Вычисление остатка от деления.

Как видно из таблицы, оптимизация алгоритма A2 позволила ускорить его не менее, чем в четыре раза.

Таблица 1

Оптимизация расширенного алгоритма Эвклида для NTRU[6]

Порядок полинома N	Время, *мс*		Ускорение, S=t1/t2
	A2, t1	A2', t2	
401	4,04	0,826	4,89
449	5,17	0,997	5,19
677	11,56	2,18	5,30
1087	22,99	4,75	4,84
541	5,82	1,38	4,22
613	6,75	1,51	4,47
887	14,49	3,03	4,78
1171	25,76	5,41	4,76
659	8,08	1,99	4,06
761	10,71	2,46	4,35
1087	21,21	4,71	4,50
1499	39,25	8,85	4,44

Продолжим оптимизацию Алгоритма 1. Шаг 3 проверяет, что наибольший общий делитель равен 1 (-1). Заметим, что полином $x^N - 1$. Он имеет корень $X = 1$, который не может быть корнем полинома $f(X)$, используемого для вычисления личного ключа. Действительно, $f(X) = 3 * F(X) + 1$, полином $F(X)$ имеет одинаковое число коэффициентов равных 1 и -1, следовательно, $X = 1$ – корень полинома $F(X)$, т.е. не является корнем $f(X)$. Второй множитель $x^N - 1$ имеет порядок N, как и полином $f(X)$, но не совпадает с ним, так как все его коэффициенты равны 1. Для полинома $g(X)$ количество 1 и -1 отличается на 1, что обеспечивает отсутствие корня $X = 1$. Следовательно, проверка наибольшего общего делителя не имеет смысла и может быть опущена. Шаг 4, в котором выполняется вычисление по формуле $b(X) = (d(X)^{-1} (\mathrm{mod}\ p) * u(X)) \bmod p$, фактически также может быть опущен, так как $d(X) = 1$, обратный элемент для единичного элемента равен 1, а вычисления $u(X)$ изначально выполняются по модулю 2.

Таким образом, алгоритм 2 в улучшенном варианте можно рассматривать не только как реализацию расширенного алгоритма Эвклида, но и как вычисление обратного элемента по модулю 2.

Использование almost inverses (почти инверсий)[2]

При вычислении инверсии по модулю 2 предыдущим алгоритмом использовались операция деления, в результате которой порядок полинома на каждом шаге уменьшался. Вместо обнуления старших коэффициентов полинома при вычислении инверсии предложено обнуление младших коэффициентов [2] и вычисление общего числа таких коэффициентов k. В этом случае фактически решается уравнение $a(X) * u(X) + c(X) * v(X) = X^k$. Значение $u(X)$ авторы [2] называют Almost Inverses (почти инверсия) и предлагают использовать этот алгоритм для вычисления инверсии для эллиптических кривых. Для получения значения инверсии результат достаточно разделить на X^k.

[6] Результаты приведены для порядков полинома, рекомендованных [1].

В работе [3] показана применимость этого метода для вычисления инверсии по модулю 2 в алгоритме NTRU. В качестве необходимых условий возможности применения этого алгоритма используются условия:

$$\gcd(a(X), c(X)) = 1;\ c(0) = 1.$$

Как показано выше, данные условия выполняются.

Алгоритм 3 (А3) [3]

Вход. Порядок $p = 2$, полиномы $a(X); c(X) = X^N - 1$.

Выход. Полином $b(X)$, такой что $b(X) = a(X)^{-1}$ в кольце $(Z/2Z)/[X]/(X^N - 1)$.

1. $k = 0;\ b(X) = 1;\ c(X) = 0;\ f(X) = a(X);\ g(X) = x^N - 1$.
2. Выполнить цикл:

2.1. Определение количества младших нулевых коэффициентов (r) в $f(X)$;

2.2. Сдвиг $f(X)$ вправо на r элементов;

2.3.Сдвиг $c(X)$ влево на r элементов;

2.4. $k += r$;

2.5.Если $f(X) == 1$ то Выход из цикла;

2.6.Если порядок $f(X)$ меньше порядка $g(X)$, то поменять местами $f(X)$ и $g(X)$, $b(X)$ и $c(X)$;

2.7. $f(X) = f(X)\hat{}g(X); b(X) = b(X)\hat{}c(X)$.

3. $b(X) = b(X) * X^{(N-k)\%N}$.

Оптимизация алгоритма А3.

1. Вместо замены местами полиномов использовать замену местами адресов этих полиномов.
2. Для выполнения этапа 3 использовать циклический сдвиг полинома.
3. Везде, где возможно, использовать SSE команды

Результаты сравнения алгоритмов А2' и оптимизированного А3 будут приведены ниже после рассмотрения очередного алгоритма.

Использование улучшенного алгоритма вычисления GCD и обратного элемента [5]

Этот алгоритм отличается от алгоритма А2 тем, что вычисление выполняется с помощью прямого и обратного преобразований. При прямом преобразовании выполняется последовательное вычисление частного и остатка, как в алгоритме вычисления GCD. При этом все частные запоминаются в стеке. При обратном преобразовании выполняется фактическое вычисление обратного элемента, если GCD = 1. В противном случае делается вывод об отсутствии обратного элемента.

Алгоритм вычисления обратного элемента с учетом идеи автора [5], в применении к полиномам:

Алгоритм А4.

Прямой ход:

1. $d(X) = c(X); v_3(x) = a(x); r = 0$.
2. Пока ($v_3(X) \neq 1$) выполнить:

2.1. $q(X) = d(X)/v_3(X);\ t_3(X) = d(X)\%v_3(X)$;

2.2. push ($q(x)$) r++;

2.3. $d(x) = v_3(x)$;

2.4. $v_3(x) = t_3(x)$.

Обратный ход

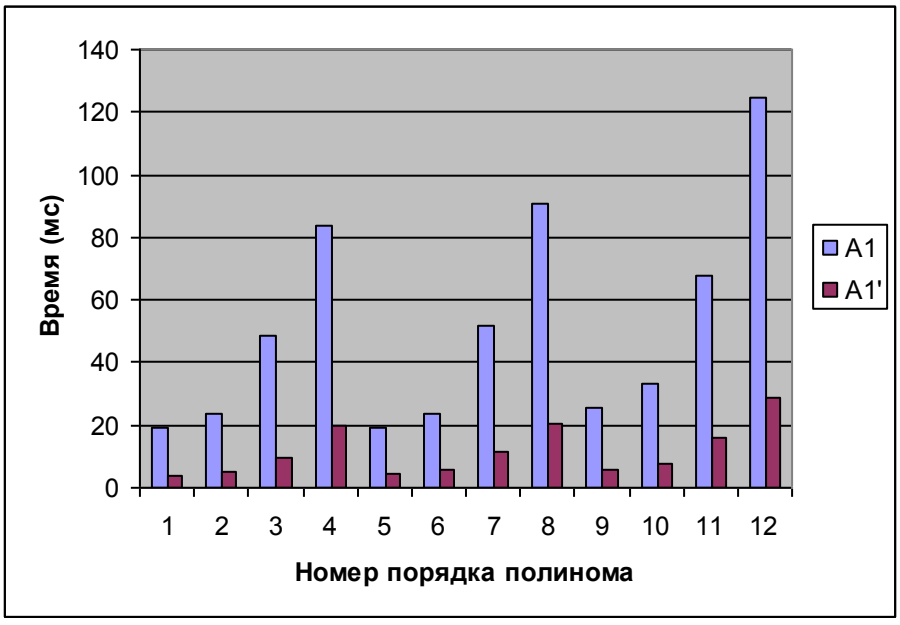
1. $S(x) = 0$;
2. Для $i = 1, 2, \ldots, r$ выполнить:

2.1. $q(x) = pop()$;
2.2. $b(x) = S(X) + q(x) * v_3(x)$;
2.3. $S(x) = v_3(x)$;
2.4. $v_3(x) = b(x)$.

Таблица 2

Сравнение всех алгоритмов вычисления инверсии по модулю 2

Порядок полинома N	Время, *мс*			Ускорение, S1=t2/t3	Ускорение, S1=t2/t4	Ускорение, S3=t4/t3
	A2'(t2)	A3(t3)	A4 (t4)			
401	0,826	0,25	0,80	3,30	1,03	3,20
449	0,997	0,32	0,96	3,12	1,04	3,00
677	2,18	0,63	1,9987	3,46	1,09	3,17
1087	4,75	1,62	4,63	2,93	1,03	2,86
541	1,38	0,41	1,38	3,37	1,00	3,37
613	1,51	0,53	1,47	2,85	1,03	2,77
887	3,03	1,04	3,00	2,91	1,01	2,88
1171	5,41	1,96	5,44	2,76	0,99	2,78
659	1,99	0,61	1,83	3,26	1,09	3,00
761	2,46	0,82	2,43	3,00	1,01	2,96
1087	4,71	1,61	5,12	2,93	0,92	3,18
1499	8,85	3,19	-	2,77	-	-



Теоретический анализ производительности алгоритма, проведенный автором [5], обещал улучшение производительности приблизительно на 54 %, но при этом не учитывались свойства кэша. В случае полинома элемент кэша – это полином, число элементов кэша приблизительно равно N/2. Таким образом, весь стек во внутреннем кэше не помещается, это приводит к существенным потерям производительности. Более того, для N = 1499 необходимо вместо выделения памяти в стеке использовать выделение динамической памяти, что также замедляет выполнение функции. В табл. 2 приведены результаты сравнения трех оптимизированных алгоритмов вычисления обратного элемента для модуля, равного 2^7

[7] Значение N=1499 для алгоритма A4 не используется.

Таким образом, использование алгоритма А3 вместо оптимизированного алгоритма А2’ позволяет увеличить производительность еще примерно в три раза, получить общее увеличение производительности не менее, чем в 12 раз. Использование же усовершенствованного алгоритма вычисления инверсии (А4) по сравнению с обычным (А2), если и дает выигрыш, то максимум на 9 %, а алгоритм вычисления Almost Invers (А3) существенно выигрывает.

Заметим, что полученные и результаты противоречат результатам из [4 и 5]. Это связано с тем, что в указанных работах определялось только общее число необходимых итераций и не учитывались возможности их параллельного выполнения, а также свойства кэша.

Вернемся к оптимизации алгоритма А1.

При вычислении по формуле $b(X) = (2 * b(X) - a(X) * b(X)^2) \bmod p$ для вычисления $b(x)^2$ используется формула сокращенного умножения, а операции сложения, вычитания и приведения по модулю выполняются с помощью SSE команд.

В табл. 3 приведены окончательные результаты, полученные после оптимизации алгоритма А1 (алгоритм А1'), а на рисунке – диаграмма с полученными результатами.

Таблица 3

Вычисление инверсии по модулю $p = 2048$

Порядок полинома N	Время, *мс*		Ускорение, S=t1/t2
	A1, t1	Улучшенный A1, t2	
401	19,42	3,89	4,99
449	23,89	5,07	4,71
677	48,60	9,73	4,99
1087	83,56	20,10	4,16
541	19,06	4,37	4,36
613	23,70	5,52	4,29
887	51,95	11,56	4,49
1171	90,78	20,29	4,47
659	25,75	5,75	4,48
761	33,26	7,53	4,42
1087	67,96	15,76	4,31
1499	124,85	28,99	4,31

В алгоритме А' для вычисления инверсии по модулю 2 используется алгоритм А3. Для параллелизации вычислений используются SIMD команды. Потоки не используются.

Заметим, что ускорение распределено практически равномерно; это связано с тем, что системные погрешности оказывают на этот алгоритм меньше влияние, чем на предыдущие алгоритмы, так как время выполнения этого алгоритма больше.

Таким образом, оптимизация алгоритма вычисления инверсии привела к более чем четырехкратному увеличению скорости

Список литературы: 1. *American* National Standard for Financial Services ANSI X9.98 – 2010. Lattice-Based Polynomial Public Key Establishment Algorithm for the Financial Services Industry. 2. *R. Schroeppel, H. Orman, S. O'Malley and O. Spatscheck*, “Fast key exchange with elliptic curve systems”.Advances in Cryptology, Proc. Crypto'95, LNCS 963. 3 NTRU Cryptosystems Technical Report.. http://www.securityinnovation.com/uploads/Crypto/NTRUTech014.pdf. 4. *Kyle Wilhelm*. Aspects of Hardware Methodologies for the NTRU Public-Key Cryptosystem. https://ritdml.rit.edu/bitstream/handle/1850/7774/KWillaimsThesis02-2008.pdf?sequence=1. 5. *Boris S. Verkhovsky*. Enhanced Euclid Algorithm for Modular Multiplicative Inverse and Its Application in Cryptographic Protocols. *Int. J. Communications, Network and System Sciences*, 2010, 3, 901-906.

Харьковский национальный университет радиоэлектроники

Поступила в редколлегию 20.09.2012