УДК 681.3.06

Д.И. ЛАВРИНЕНКО

ПРИМЕНЕНИЕ БЫСТРОГО ПРЕОБРАЗОВАНИЯ ФУРЬЕ В КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЯХ

Введение

Криптографические системы, использующие несимметричные алгоритмы шифрования [1], базируются на специфическом математическом аппарате, который оперирует с целыми числами, разрядность которых превышает размер машинного слова. К таким системам часто предъявляются жесткие требования по производительности, которые напрямую зависят от вычислительной сложности математических алгоритмов. Поэтому решение задач минимизации вычислительной сложности этих алгоритмов и как следствие повышение скорости преобразований является важной задачей. При несимметричных преобразованиях наибольшую вычислительную сложность имеет операция модульного возведения в степень, которая базируется на операции модульного умножения. В данной статье производится краткий обзор существующих методов модульного умножения и вопрос выполнения модульного умножения «больших» целых чисел с использованием быстрого преобразования Фурье.

Для проведения сравнительной оценки вычислительной сложности введем следующие обозначения:

X, Y,... (большие буквы) — целые неотрицательные числа многократной точности. Причём числом **многократной точности** называется целое число, разрядность которого превышает размер машинного слова.

x, y,... (маленькие буквы) — числа однократной точности.

b — длина машинного слова в битах.

$B = 2^b$.

$L(X)$ — длина числа X в битах, т.е. $2^{L(X)-1} \le X < 2^{L(X)}$.

$l(X)$ — длина числа X в словах, т.е. $B^{l(X)-1} \le X < B^{l(X)}$.

X_i — i-й блок числа X; $i=0,1,\ldots,(L(X)-1)/b$, тогда $X=(X_{l(X)-1}\ldots X_1X_0)$

I_{a0} — вычислительная сложность одной машинной команды сложения, вычитания или пересылки.

I_{l0} — вычислительная сложность одной машинной команды цикла или перехода.

I_{m0} — вычислительная сложность одной машинной команды умножения или деления.

$x \bmod y$ — остаток от деления x на y.

$x \equiv y \pmod m$ — x сравнимо с y по модулю m.

$\lfloor x \rfloor$ – целая часть x (наибольшее целое число, не превосходящее x).

Наиболее простым и естественным методом умножения является выполнение операции «в столбик». Эти алгоритмы известны сейчас как арифметика Кнута [2]. Приведем схему алгоритма умножения в арифметике Кнута.

Исходные данные: числа X и Y.

Результат: число $Z = X \cdot Y$, $l(Z) = l(X)\, l(Y)$.

1. $Z:=0$.
2. Для i, принимающего значения от 0 до $l(X)-1$, выполнить шаг 3.
3. Для j, принимающего значения от 0 до $l(Y)-1$, выполнить шаги 4-5.
4. $T:=X_iY_j$; $l(T)=2$.
5. $(Z_{l(Z)-1}\ldots Z_{i+j}) := (Z_{l(Z)-1}\ldots Z_{i+j}) + T$.

Исходя из набора операций в каждом шаге и количества выполнений шага, вычислительную сложность алгоритма оценим как:

$$I_m(l(X), l(Y)) = (3I_{a0}+I_{l0}+I_{m0})\, l(X)\, l(Y)+I_{l0}\, l(X)+(l(X)+l(Y))I_{a0}.$$

При $l(X)=l(Y)=l$ получаем:

$$I_m(l) = I_{m0}\, l^2+I_{l0}(l^2+l)+I_{a0}\,(3l^2+2l).$$

1. Особенности применения БПФ для вычисления произведения

Вычислять произведение двух чисел многократной точности можно также исходя из полиномиального представления числа. Если представить исходные числа в виде полиномов и применить к ним операцию свертки, то получим полиномиальное представление произведения исходных чисел. Здесь мы можем применить для выполнения операции свертки спектральные преобразования, в частности преобразование Фурье. Кратко изложим сущность и свойства преобразования Фурье [3].

Преобразование Фурье связано с вычислением полиномов и их интерполяцией. Пусть

$$p(x) = \sum_{i=0}^{n-1} a_i x^i$$

– полином (n-1)-й степени. Его можно однозначно представить двумя способами: списком его коэффициентов $a_0, a_1 \ldots a_{n-1}$ и списком его значений в n различных точках $x_0, x_1 \ldots x_{n-1}$. Вычисление преобразования Фурье вектора $[a_0, a_1 \ldots a_{n-1}]^T$ эквивалентно превращению представления полинома $\sum_{i=0}^{n-1} a_i x^i$ списком его коэффициентов в представление его списком значений в точках $\omega^0, \omega^1 \ldots \omega^{n-1}$ (физический аналог операции – получение спектра сигнала). Вычисление обратного преобразования Фурье эквивалентно интерполяции полинома по его значениям в точках $\omega^0, \omega^1 \ldots \omega^{n-1}$ (восстановление сигнала по его спектру). Поясним смысл обозначений $\omega^0, \omega^1 \ldots \omega^{n-1}$.

Обычно преобразование Фурье определяется над кольцом комплексных чисел. Обобщая, можно определить преобразование Фурье над произвольным коммутативным кольцом (R, +, ·, 0, 1). Элемент ω из R, такой что

1. $\omega \neq 1$,
2. $\omega^n = 1$,
3. $\sum_{j=0}^{n-1} \omega^{jp} = 0$ для $1 \leq p < n$,

называется *примитивным корнем n-й степени из единицы*. Элементы $\omega^0, \omega^1 \ldots \omega^{n-1}$ называются *корнями n-й степени из единицы*.

Можно осуществлять преобразование Фурье на множестве точек, отличных от корней из единицы. Но выбирая степени корня ω, мы делаем вычисления преобразований особенно простыми.

Прямое или обратное преобразование Фурье вектора a из R^n можно вычислить за $O(n^2)$ операций. Но если n – степень числа 2, то можно сократить число операций до $O(n \log n)$. Эта модификация известна как быстрое преобразование Фурье (БПФ) [3].

Одно из основных приложений преобразования Фурье – вычисление свёртки двух векторов [4]. Свёртка двух векторов ***a*** и ***b*** равна обратному преобразованию, применённому к покомпонентному произведению их образов прямого преобразования. Формально это записывается так: $a \otimes b = F^{-1}(F(a) \cdot F(b))$. Заметим, что коэффициенты произведения полиномов – это в точности компоненты свёртки векторов коэффициентов этих полиномов. Если теперь представить числа в полиномиальном виде, то станет очевидным использование преобразований Фурье для вычисления произведения двух чисел.

Обычно преобразование Фурье выполняется в кольце комплексных или вещественных чисел. Но для вычисления произведения нужен точный результат. Избежать погрешностей при работе с вещественными числами можно, если производить вычисления в конечном кольце – кольце R_m целых чисел по модулю m, где m будет таким, чтобы в R_m существовал примитивный корень n-й степени из единицы. Если n – степень числа 2, то подходящее m существует всегда. В частности, если ω и n – степени числа 2, то m можно выбрать как $\omega^{n/2} + 1$. Числа вида $2^p + 1$ известны как числа Ферма, а вычисления по такому модулю известны как теоретико-числовые преобразования [3]. Модульные вычисления по такому модулю заметно проще, чем по произвольному модулю.

Приведем схему алгоритма БПФ:

Исходные данные: число A.

Промежуточные данные: массив S, $l(S) =$ n.
Результат: массив B = БПФ(A).
1. Для i, принимающего значения от 0 до n, выполнить шаг 2.
2. S[i] = A_i.
3. Для l, принимающего значения от 0 до $\log_2 n$-1, выполнить шаг 4.
4. Для i, принимающего значения от 0 до n-1, выполнить шаг 5.
5. S[i] = S[i & ~(2^l)] + $\omega^{inv(i)>>l<<l}$·S'[i | (2^l)] mod m.
6. Для i, принимающего значения от 0 до l(S)–1, выполнить шаг 7.
7. B[i] = S[inv(i)].

Здесь inv(i) обозначает битово-инверсное представление числа i в разрядной сетке из $\log_2 n$ разрядов. Строка 5 алгоритма представляет собой не что иное, как базовую (двухточечную) операцию БПФ [5], известную также под названием операции «бабочка» [3].

Обратное преобразование Фурье аналогично прямому, если в шаге 5 ω^p заменить на ω^{-p} и на шаге 7 добавить умножение на n^{-1}.

Если рассматривать машинную реализацию БПФ, то можно заметить, что если ω и n – степени числа 2, то все умножения на ω^p и n^{-1} можно заменить операциями двоичного сдвига, которая обычно более эффективна, чем операция умножения. Все значения степеней ω можно вычислить заранее.

2. Анализ вычислительной сложности БПФ

Для оценки вычислительной сложности БПФ введем обозначения
I_{Fs} — вычислительная сложность одной операции сдвига.
I_{Fa} — вычислительная сложность одной операции модульного сложения или вычитания.
I_{Fm} — вычислительная сложность одной операции модульного умножения.
I_{Fv} — вычислительная сложность одной операции пересылки.

Разрядность операндов этих операций, в отличие от операций в классическом умножении, зависит от разрядности исходных данных и от параметров ω и n. Кроме того, операции выполняются по специальному модулю поэтому они обозначаются отдельно.

Оценим вычислительную сложность прямого БПФ как:

$$I_{FFTD}(n) = I_{Fs}(n\cdot\log_2 n) + I_{Fa}(n\cdot\log_2 n) + I_{Fv}(n\cdot(\log_2 n+2)) + I_a(n\cdot\log_2 n).$$

Вычислительная сложность обратного преобразования есть

$$I_{FFTR}(n) = I_{Fs}(n\cdot(\log_2 n+1)) + I_{Fa}(n\cdot\log_2 n) + I_{Fv}(n\cdot(\log_2 n+2)) + I_a(n\cdot\log_2 n).$$

Для того, чтобы умножить два числа с использованием БПФ, необходимо получить прямые БПФ-преобразования исходных чисел, осуществить их покомпонентное умножение и получить обратное преобразование результата умножения. Но если рассматривать умножение как составную часть алгоритма возведения в степень, то можно опустить одно прямое преобразование, т.к. второй аргумент можно вычислять заранее и оставлять в базисе БПФ. Оценим теперь общую вычислительную сложность умножения с применением БПФ.

$$I_{FFTR}(n)=I_{Fs}(n\cdot(2\cdot\log_2 n+1))+I_{Fa}(2\cdot n\cdot\log_2 n)+I_{Fv}(2\cdot n\cdot(\log_2 n+2))+I_a(n\cdot\log_2 n)+I_{Fm}\,n.$$

Анализируя применение БПФ для умножения чисел многократной точности, нельзя не отметить несколько особенностей и ограничений метода.

Во-первых, БПФ требует, чтобы оно осуществлялось на количестве точек $n = 2^p$. Это также требует, чтобы полиномиальное представление исходных чисел тоже имело 2^p коэффициентов. Если $n \neq 2^p$, то нужно преобразовать схему БПФ, как это указано в [3], что повлечет за собой дополнительные вычислительные затраты. Другой выход из положения – добавить к полиномиальному представлению столько нулевых коэффициентов, чтобы их общее количество стало 2^p, что влечет за собой увеличение размерности БПФ, что также увеличивает вычислительную сложность преобразования. Из всего сказанного выше следует, что применение БПФ наиболее эффективно при умножении чисел определенной длины.

Во-вторых, при выборе количества точек БПФ также имеются ограничения. Это следует из $m = \omega^{n/2} + 1$, $\omega = 2^p$, и $L(X) = n \cdot L(b)$, где b – блок, на которые разбивается исходное число (по своей математической сути блоки – это коэффициенты полиномиального представления числа). Покажем это.

m должно быть не меньше величины максимального элемента свертки, чтобы не вносить искажений, значит $L(m) \geq 2 \cdot L(b) \cdot \log_2 n$. Но для реализации удобнее, если m как можно меньше и кратно разрядной сетке ЭВМ, поскольку в противном случае цена отдельной операции по модулю m значительно (квадратично) возрастает. Это предопределяет $L(m) = 2 \cdot L(b)$, а как исправить возникающие искажения, связанные с компонентом $\log_2 n$, будет сказано ниже.

$m = \omega^{n/2} + 1$, $\omega = 2^p$, причём $\omega \geq 2$, отсюда следует $L(m) \geq n/2$. Из $n = L(X) / L(b)$ и $L(m) = 2 \cdot L(b)$ следует $n = 2L(X) / L(m)$. В результате получаем

$$L(m) \geq L(X) / L(m) \Rightarrow L(m) \geq \sqrt{L(X)},$$

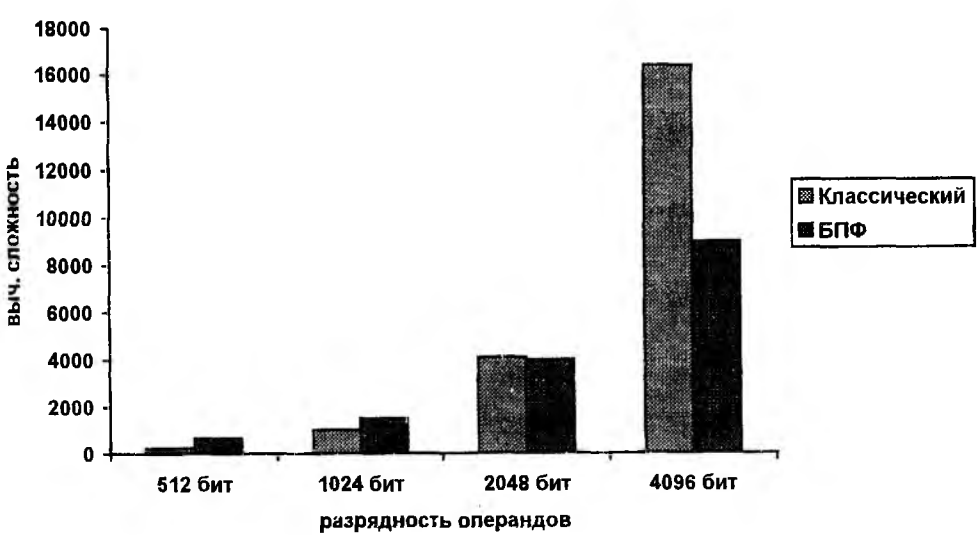
$$n \leq 2 \cdot L(m) \Rightarrow n \leq 2 \cdot \sqrt{L(X)}$$

Теперь для того, чтобы исправить искажения, возникшие в результате применения модуля меньшей длины, воспользуемся китайской теоремой об остатках. Здесь теорема будет применима, если мы вычисляем компоненты свёртки дважды: по модулю m и по модулю n (эти числа будут взаимно просты, так как n – степень 2, а m – нечётное). Вычисления компонентов свёртки по модулю мы вычисляем с помощью преобразования Фурье, как было объяснено выше, а про вычисления свёрток по модулю 2^p объясняются в [4].

3. Экспериментальное исследование вычислительной сложности умножения на БПФ

Известно, что каждая из элементарных операций на разных типах ЭВМ имеет различную вычислительную сложность. Сравнивая формулы для вычислительной сложности умножения многократной точности классическим методом и методом с преобразованиями Фурье, можно заметить, что метод с преобразованиями Фурье будет тем выгоднее, чем меньшую вычислительную сложность имеют операции по модулю $\omega^{n/2} + 1$ (операции сложения, вычитания, сдвига, умножения), которые на большинстве ЭВМ имеют вычислительную сложность примерно в 3-4 раза большую, чем по модулю $\omega^{n/2}$. Поэтому применение БПФ в умножении многократной точности будет тем выгоднее, чем меньше будет вычислительная сложность этих преобразований. Исходя из этого можно дать следующие рекомендации:

- следует добиться минимальной вычислительной сложности для вычислений по модулю $\omega^{n/2} + 1$. Возможно, такие вычисления будет выгодно производить на специализированных ЭВМ;
- поскольку при умножении классическим методом количество операций умножения значительно больше, чем при умножении с БПФ, то последний метод будет выгоднее на ЭВМ, где операции умножения будут иметь большую вычислительную сложность по сравнению с операциями сложения, вычитания и сдвигов.



Сравнивая формулы оценки вычислительной сложности для алгоритмов умножения многократной точности классическим методом и с БПФ , можно заметить, что зависимость вычислительной сложности алгоритма от длины исходных операндов для обоих алгоритмов носит полиномиальный характер, но порядок полинома для классического алгоритма будет выше, чем для алгоритма с БПФ. Поэтому эффект от применения БПФ будет присутствовать всегда, начиная с определённой длины исходных операндов, даже если вычислительная сложность операций в БПФ будет сравнительно велика.

Проведённый вычислительный эксперимент показал (рисунок), что на процессорах Intel Pentium эффект от применения БПФ в умножении будет появляться для исходных операндов длиной не менее 2048 бит. Это объясняется в первую очередь относительно большой вычислительной сложностью команд сдвигов и операций по модулю $\omega^{n/2}+1$ на процессорах этого типа.

Для сравнения возьмём сигнальный процессор ADSP 21063, архитектура которого значительно отличается от процессоров Intel. Операции умножения на этом процессоре имеют такую же вычислительную сложность, как и операции сложения и сдвига, поэтому эффект от применения БПФ здесь будет иметь место при ещё большей длине исходных операндов (порядка 16284 бит и более).

Заключение

Так как в современных криптосистемах в основном применяются операнды разрядностью в 512 и 1024 бит, то имея в виду реализацию на процессорах архитектуры Intel, можно сказать, что БПФ может найти применение в криптосистемах с повышенными требованиями безопасности, т.е. при длинах модулей 2048 и более бит. Что касается реализации на ЭВМ другой архитектуры, здесь перспективность БПФ определяется вышеупомянутыми рекомендациями.

Список литературы **1.** *Diffie W., Hellman M.E.* New directions in cryptography // IEEE trans. on Information Theory.– 1976.– V.IT-22.– №6.– pp. 644-654. **2.** *Кнут Д.* Искусство программирования для ЭВМ. В 3-х т. — Т.2. Получисленные алгоритмы. — М.: Мир, 1977. — 387 с. **3.** *Цифровая* обработка сигналов. Ч.1. Учебное пособие. И.Д. Горбенко, А.С. Трощило, К.В. Бессарабенко. МО СССР, 1988. **4.** *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов – М.: Мир,1979. – 536 с. **5.** *Рабинер Л. Гоулд Б.* Теория и применение цифровой обработки сигналов – М.: Мир, 1978. – 848 с.

Харьковский государственный технический университет радиоэлектроники

Поступила в редколлегию 27.03.2000