УДК 681.3.06

А.В. СВИНАРЕВ

МЕТОДЫ УСКОРЕНИЯ ПРОЦЕДУР ЦИФРОВОЙ ПОДПИСИ КЛАССА ЭЛЬ-ГАМАЛЯ

Одной из особенностей цифровых подписей класса Эль-Гамаля является временная несимметрия — существенное различие временных затрат на формирование и проверку подписи. Несимметрия может представлять собой ограничение, которое необходимо учитывать при реализации подписей данного класса во многих практических приложениях. Поясним это на примере алгоритма ГОСТ Р 34.10 — 94.

Процедура формирования подписи сообщения включает в себя следующие этапы:

1. Вычислить $h(M)$ — значение хеш-функции h от сообщения M. Если $h(M) = 0$, то присвоить $h(M)$ значение $\underbrace{00\ldots01}_{255}$.
2. Выработать целое число k, $0 < k < q$.
3. Вычислить два значения:
   $r := a^k \pmod p$; $r' = r \pmod q$.
4. Если $r' = 0$, перейти к этапу 2 и выработать другое значение числа k.
5. С использованием секретного ключа x пользователя (отправителя сообщения) вычислить значение $s := (xr' + kh(M)) \pmod q$.
6. Если $s = 0$, перейти к этапу 2, в противном случае закончить работу алгоритма.

Подписью для сообщения M является вектор $< r' >_{256} \| < s >_{256}$.

Процедура проверки включает в себя следующие этапы :

1. Проверить условия $0 < s < q$; $0 < r' < q$.

Если хотя бы одно из этих условий не выполнено, то подпись считается недействительной.

2. Вычислить $h(M_1)$ — значение хеш-функции h от полученного сообщения M_1. Если $h(M_1) \pmod q = 0$, присвоить $h(M_1)$ значение $\underbrace{00\ldots01}_{255}$.
3. Вычислить значение
   $v := (h(M_1))^{q-2} \pmod q$.
4. Вычислить значения
   $z_1 := sv \pmod q$; $z_2 := (q - r')v \pmod q$.
5. Вычислить значение

$$u := (a^{z_1} y^{z_2} \pmod p) \pmod q.$$

6. Проверить условие : $r' = u$.

Обозначим через I_{p0} вычислительную сложность операции модульного возведения в степень $Z=X^Y \pmod N$, где длины X и N составляют 512 бит, а длина Y — 256 бит. Выразим через эту величину вычислительные сложности процедур формирования и проверки цифровой подписи. При этом будем учитывать только операции возведения в степень, поскольку вычислительная сложность остальных операций мала по сравнению с ними. Вычислительной сложностью операции возведения в степень на шаге 3 также можно пренебречь, поскольку она эквивалентна вычислению обратного элемента $h(M_1)^{-1} \pmod q$. В результате получим, что вычислительная сложность формирования подписи составляет

$$I_{\text{фп}} \approx I_{\text{p0}}, \tag{1}$$

а вычислительная сложность проверки подписи

$$I_{\text{пп}} \approx 2I_{p0}, \tag{2}$$

поскольку на шаге 5 выполняются два возведения в степень.

Из выражений (1) и (2) следует, что алгоритм обладает несимметрией: время проверки подписи приблизительно в 2 раза больше времени ее формирования.

Уменьшения несимметрии можно добиться за счет разработки специальных алгоритмов вычисления выражения вида

$$Z = X_1^{Y_1} X_2^{Y_2} \bmod N, \tag{3}$$

которое встречается на шаге 5 процедуры проверки подписи.

Рассмотрим два алгоритма — модификации бинарного [1] и блочного [2] алгоритмов возведения в степень. Они выполняют вычисление произведения общего вида:

$$Z = \prod_{i=1}^{k} X_i^{Y_i} \bmod N. \tag{4}$$

При описании алгоритмов используем следующие обозначения: X, Y,... (большие буквы) — числа многократной точности; x, y,... (маленькие буквы) — числа однократной точности; b — длина машинного слова в битах; $B = 2^b$ — количество чисел, представимых с однократной точностью; $L(X)$ — длина числа X в битах, т.е. $2^{L(X)-1} \leqslant X < 2^{L(X)}$; $l(X)$ — длина числа X в словах, т.е. $B^{l(X)-1} \leq X < B^{l(X)}$.

Нумерация бит в числах производится от старших разрядов к младшим, начиная с 1.

Алгоритм 1 (параллельный бинарный). Исходные данные: числа X_i, Y_i, N; $N \neq 0$; $X_i < N$; $i = \overline{1,k}$.

Результат: число $Z = \prod_{i=1}^{k} X_i^{Y_i} \bmod N$, $l(Z) = l(N)$.

1. Вычислить значения
   $m := \max L(Y_i)$; $Z := 1$.
2. Дополнить все Y_i слева нулями до длины m бит.
3. Для j, принимающего значения от 1 до m, выполнить шаги 4—7.
4. Вычислить значение $Z := Z^2 \bmod N$.
5. Для i, принимающего значения от 1 до k, выполнить шаг 6.
6. Если j-й бит Y_i равен 1, то вычислить $Z := Z \cdot X_i \bmod N$.
7. Закончить работу алгоритма.

При оценке вычислительной сложности данного алгоритма будем учитывать только наиболее трудоемкие операции — возведение в квадрат на шаге 4 и умножение на шаге 6. Шаг 4 всегда выполняется m раз, а количество умножений на шаге 6 совпадает с количеством единичных бит в показателях Y_i.

Получим приближенную оценку вычислительной сложности алгоритма 1:

$$I_{mpow1}(l(N),m,k) = m\, I_{msqr}(l(N)) + \sum_{i=1}^{k} d_i\, I_{mm}(l(N)), \qquad (5)$$

где $m = \max L(Y_i)$; $I_{msqr}(l)$ — вычислительная сложность модульного возведения в квадрат при длине модуля l блоков; d_i — количество единичных бит в показателе степени Y_i; $I_{mm}(l)$ — вычислительная сложность модульного умножения двух чисел при длине модуля l блоков.

Сравним алгоритм 1 с обычным бинарным алгоритмом, вычислительная сложность которого составляет [1]:

$$I_b(l(N), L(Y)) = L(Y)\, I_{msqr}(l(N)) + d I_{mm}(l(N)).$$

Оценим выигрыш от использования алгоритма 1 при вычислении выражения (4):

$$\Delta I = \sum_{i=1}^{k} I_b\left(l(N), L(Y_i)\right) - I_{mpow1}(l(N),m,k) =$$

$$= \left(\sum_{i=1}^{k} L(Y_i) - m \right) I_{msqr}(l(N)). \qquad (6)$$

Выражение (6) приобретает наиболее простой вид при равных длинах Y_i:

$$\Delta I = m(k-1)\, I_{msqr}(l(N)).$$

Алгоритм 2 (параллельный блочный). В блочном алгоритме для уменьшения вычислительной сложности используется разбиение показателей степени Y_i на блоки. Блоком называется:

— последовательность нулевых бит любой длины;

— последовательность, начинающаяся и оканчивающаяся на единичный бит, если ее длина не превышает некоторой константы $d_{\max}$.

В тексте алгоритма используем следующие обозначения: **getblock**(Y_i, j) — операция выделения очередного блока из Y_i, начиная с j-го бита; **end**(C_i) — номер бита в показателе Y_i, на котором заканчивается блок C_i.

Исходные данные: числа X_i, Y_i, N; $N \neq 0$; $X_i < N$; $i = \overline{1, k}$.

Результат: число $Z = \prod_{i=1}^{k} X_i^{Y_i} \bmod N$, $l(Z) = l(N)$.

В алгоритме используется вспомогательная таблица $T[2^{d_{\max}}, k]$.

1. Вычислить значения
$m := (\max L(Y_i)) - 1$; $Z := 1$.
2. Дополнить все Y_i слева нулями до длины m бит.
3. Заполнить таблицу T для $i = \overline{1, k}$; $j = \overline{1, 2^{d_{\max}} - 1}$:
T[j,i] := 0, если j – четное;
T[j,i] := $X_i^j \bmod N$, если j – нечетное.
4. Выделить блоки: C_i:= **getblock**(Y_i, 1), $i = \overline{1, k}$.
5. Для j, принимающего значения от 1 до m, выполнить шаги 6 — 7.
6. Вычислить значение $Z := Z^2 \bmod N$.
7. Для i, принимающего значения от 1 до k, выполнить шаги 8 — 9.
8. Если j = end(C_i) и C_i 0, то вычислить значение $Z := Z\, T[C_i, i] \bmod N$.
9. Если j = end(C_i), то вычислить C_i:= **getblock**(Y_i, j+1).
10. Закончить работу алгоритма.

Алгоритм выполнения операции C_i = **getblock** (Y_i, m).

1. Если m-й бит Y_i равен нулю, то:

1.1. Определить n — номер первого единичного бита, стоящего справа от m-го бита.

1.2. $C_i := 0$.

1.3. **end**$(C_i) := n-1$.

2. Если m-й бит Y_i равен единице, то:

2.1. Выделить из Y_i последовательность бит w, начиная с m-го бита, обладающую такими свойствами: 1) w оканчивается на 1; 2) $l(w) \le d_{\max}$; 3) для всех возможных последовательностей, удовлетворяющих свойствам 1 и 2, длина w максимальна.

2.2. $C_i := w$.

2.3. **end**$(C_i) := m + l(w) - 1$.

Оценку вычислительной сложности алгоритма 2 проведем с учетом только наиболее трудоемких операций (табл. 1).

Таблица 1

№ шага	Операция	Количество повторений
3	Умножение	$k\left(2^{d_{\max}-1}\right)$
6	Возведение в квадрат	m
8	Умножение	$\sum_{i=1}^{k}\mu(Y_i)$

Через $\mu(Y_i)$ обозначено количество ненулевых блоков в показателе Y_i.

Таким образом, вычислительная сложность алгоритма 2 составляет:

$$I_{mpow2}(l(N),m,k) = mI_{msqr}(l(N)) + \left(k\left(2^{d_{\max}-1}\right) + \sum_{i=1}^{k}\mu(Y_i)\right) I_{mm}(l(N)). \quad (7)$$

По сравнению с обычным блочным алгоритмом с вычислительной сложностью [2]

$$I_{bl}(l(N),L(Y)) = L(Y)\, I_{msqr}(l(N)) + \left(2^{d_{\max}-1} + \mu(Y)\right) I_{mm}(l(N))$$

алгоритм 2 дает выигрыш

$$\Delta I = \sum_{i=1}^{k} I_{bl}\left(l(N), L(Y_i)\right) - I_{mpow2}(l(N),m,k) =$$

$$= \left(\sum_{i=1}^{k} L(Y_i) - m\right) I_{msqr}(l(N)).$$

Сравнив (5) и (7), определим выигрыш алгоритма 2 сравнительно с алгоритмом 1:

$$\Delta I_{12} = I_{mpow1}(l(N),m,k) - I_{mpow2}(l(N),m,k) =$$
$$= \left(\sum_{i=1}^{k} (d_i - \mu(Y_i)) - 2^{d_{\max}-1} \right) I_{mm}(l(N)).$$

Изложенные алгоритмы были реализованы программно для 32-разрядной ПЭВМ Пентиум-100. Для модульного умножения и возведения в квадрат использовалась арифметика Монтгомери. В табл. 2 представлены результаты экспериментальной оценки вычислительной сложности алгоритмов при формировании и проверке цифровой подписи по ГОСТ Р 34.10 — 94. Для сравнения приведено время выполнения обычного блочного алгоритма.

Таблица 2

Алгоритм	Время, с			Несимметрия
	формирования подписи	проверки подписи	суммарное	
Разрядность 512 бит				
1	0,041	0,070	0,111	1,70
2	0,041	0,052	0,093	1,27
Блочный	0,041	0,082	0,123	2,00
Разрядность 1024 бита				
1	0,29	0,52	0,81	1,79
2	0,29	0,35	0,64	1,21
Блочный	0,29	0,58	0,87	2,00

Таким образом, при практической реализации параллельных алгоритмов возведения в степень удалось достигнуть значительного уменьшения временной несимметрии и ускорения процедуры проверки цифровой подписи на 30 %.

Список литературы: 1. *Кнут Д.* Искусство программирования для ЭВМ: В 3 т.: Пер. с англ. М.: Мир, 1977. Т. 2: Получисленные алгоритмы. 387 с. 2. *Горбенко И.Д., Качко Е.Г., Свинарев А.В.* Повышение быстродействия алгоритмов арифметики многократной точности // Безопасность информ. 1997. № 1. С. 15—20.

Харьковский государственный технический университет радиоэлектроники

Поступила в редколлегию 26.06.97