Баранник Владимир Викторович, доктор технических наук, профессор, начальник кафедры Харьковского университета Воздушных Сил.
E-mail: barannik_v_v@mail.ru.
Баранник Володимир Вікторович, доктор технічних наук, професор, начальник кафедри Харківського університету Повітряних Сил.
Barannik Vladimir, Dr. of Technical Science, Chief of Department, professor at Kozhedub Air Force university, Kharkiv.
Туренко Сергій Вікторович, аспірант Харківського національного університету радіоелектроніки**.**
E-mail: barannik_v_v@mail.ru.
Туренко Сергей Викторович, аспирант Харьковского национального университета радиоэлектроники.
Turenko Sergey Viktorovich, postgraduate student of Kharkiv national university of radioelectronics.

УДК 004.056.55

АРИФМЕТИКА С ОТЛОЖЕННЫМ ПЕРЕНОСОМ ДЛЯ ЦЕЛЫХ ЧИСЕЛ

Андрей Охрименко

Криптографические преобразования с открытым ключом широко используются и положены в основу направленного шифрования, выработки общего секрета и электронной цифровой подписи. Поэтому, задача повышения производительности криптографических преобразований с открытым ключом является актуальной. Повысить производительность можно за счет увеличения производительности операций над целыми числами. Предлагается DCF представление целых чисел, в котором число разбивается на машинные слова, где в каждом машинном слове отводится блок под представление самого числа и блок под последующие переносы в старшие разряды, либо займы из старших разрядов. Приводятся алгоритмы основных арифметических операций с отложенным переносом, даются рекомендации по эффективной программной реализации арифметических операций (сложение, вычитание).
Ключевые слова: *DCF представление, отложенный перенос, целые числа, целочисленная арифметика, программная реализация, распараллеливание.*

Введение. Информационно-телекоммуникационные системы основательно вошли в жизнь современного общества, переведя его в новую эпоху информационного общества. Тем не менее, само общество задается вопросом о конфиденциальности персональных данных, личной информации, и т.д. Подобное волнение связано с информацией опубликованной рядом информационных агентств по материалам предоставленных Эдвардом Сноуденом. Для обеспечения защиты информации, которая циркулирует, создается, модифицируется, хранится и уничтожается, создаются системы защиты информации, ядром которых являются методы криптографической защиты информации. Криптографические преобразования с открытым ключом занимают особое место среди криптографических алгоритмов и положены в основу направленного шифрования, выработки общего секрета и электронной цифровой подписи. Актуальность задачи повышения производительности криптографических преобразований с открытым ключом, подтверждает тот факт, что преобразования с открытым ключом существенно уступают в производительности симметричным преобразованиям.

Сейчас получили широкую популярность криптографические преобразования, основанные на сложности решения следующих математических задач [4]:

– Факторизация большого числа.

– Дискретный логарифм в поле целых чисел и в поле полиномов.

– Дискретный логарифм в группе точек эллиптической кривой над полем целых чисел и в поле полиномов.

– Дискретный логарифм в якобиане дивизоров гиперэллиптической кривой над полем целых чисел и в поле полиномов.

– Поиска кратчайшего вектора в Эвклидовом пространстве, и т.д.

Алгоритмы решения данных задач имеют субэкспоненциальную сложность, поэтому при определенных размерах ключей, их решение считается невозможным на современном этапе развития вычислительной техники.

Следует заметить, что среди перечисленных задач, большинство используют операции над целыми числами. Поэтому, предлагается повысить производительность криптографических

преобразований с открытым ключом за счет увеличения производительности операций над целыми числами. Среди направлений по повышению производительности операций над целыми числами, можно выделить представление целых чисел, что в дальнейшем позволяет модифицировать алгоритмы операций над ними.

Представление чисел для выполнения операций с отложенным переносом

Представление целых чисел. Известно множество представлений целых чисел [8], которые можно условно разделить на позиционные и непозиционные. Каждое представление ориентировано для выполнения сугубо специфических задач в современной криптографии [2, 5, 8, 11]:

1. *Обычное* [8].

Отдельно стоит остановиться на представлении элементов поля. Как правило, поля чисел $\mathbf{GF}(p)$ представляются двоичным вектором длиной l (массив машинных слов двоичной длиной w):

$$b_{l-1}2^{l-1}+b_{l-2}2^{l-2}+\cdots+b_1 2^1+b_0 = a_{n-1}2^{(n-1)w}+a_{n-2}2^{(n-2)w}+\cdots+a_1 2^w+a_0, \qquad (1)$$

где $n=\lceil \frac{l}{w} \rceil$ – число машинных слов необходимых для представления числа двоичной длины l; b_i – двоичные коэффициенты; a_j – машинные слова длиной w.

2. *Non-adjacent form (NAF)* [6, 9, 10].

Представление целого числа в двоичной форме $b_{l-1}2^{l-1}+b_{l-2}2^{l-2}+\cdots+b_1 2^1+b_0$ в k-значной NAF-форме:

$$b_{l-1}2^{l-1}+b_{l-2}2^{l-2}+\cdots+b_1 2^1+b_0 = a_{n-1}2^{(k-1)}+a_{n-2}2^{(k-2)}+\cdots+a_1 2+a_0, \qquad (2)$$

где k – число коэффициентов a_j необходимых для представления двоичного числа длиной l; b_i – двоичные коэффициенты; $a_j \in \{-1, 0, 1\}$ – коэффициенты. Более подробно об алгоритмах преобразования чисел к NAF представлению, можно ознакомиться [5, 10].

Приведенный список необходим для демонстрации разнообразий подходов и может быть существенно расширен, однако рассмотрение различных представлений целых чисел выходит за рамки данной работы.

Представление целых чисел для операций с отложенным переносом

Несмотря на широкое применение описанных выше представлений целых чисел, они не лишены ряда недостатков, среди которых – низкая эффективность при реализации на современных микропроцессорах. Во всех операциях в приведенных представлениях присутствует операция переноса или займа, которая крайне неэффективно реализуется современными процессорами – не выполняется условия спаривания команд [7]. Добиться существенного повышения производительности в операциях над целыми числами, можно посредством откладывания выполнения операции переноса из старших разрядов в младшие и наоборот – для займа.

Предлагается новое представление целых чисел, в так называемой DCF (Delayed Carry Form) форме с отложенным переносом [1, 3], рис. 1. Число $a = d_{\mathrm{DCF}}$ в DCF представляет собой последовательность из машинных слов $d_{\mathrm{DCF}} = \{d_{n-1}, \ldots, d_1, d_0\}_{\mathrm{DCF}}$ длинной w-бит, такое что каждое машинное слово $d_i^{(w)} = a_i^{(r)} \| a_i^{(v)}$, где r-бит переноса $a_i^{(r)}$, v-бит – заполняются битами из числа $a_i^{(v)}$. В DCF-представлении, обычное число a, двоичной длины l, в виде n блоков длиной v-бит, будет содержать $n \cdot r$-бит избыточной информации, отведенной для хранения переносов.

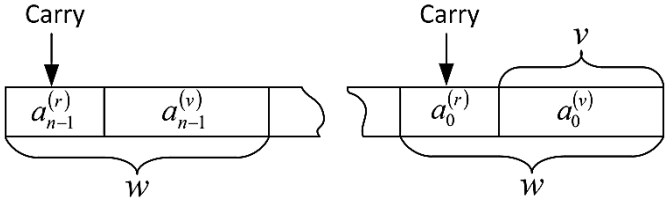

Рис. 1. DCF представление целых чисел

Числами в DCF представлении, процессор оперирует машинными словами, в которых выделены биты для накопления переноса и для хранения непосредственно самого числа, т.е. для процессора операции над машинными словами, содержащими переносы, выглядят прозрачно.

Операции над DCF числами

Для выполнения операций над DCF числами, необходимо модифицировать привычные алгоритмы арифметических операций над целыми числами.

1. Преобразование числа к DCF представлению

Под преобразованием числа в DCF представлением будем понимать резервирование (обнуление) в машинном слове $d_i^{(w)}$ длиной w-бит, r-бит под перенос $a_i^{(r)}$, оставшиеся v-бит – заполняются битами из числа $a_i^{(v)}$ в непрерывной двоичной форме, рис. 2.

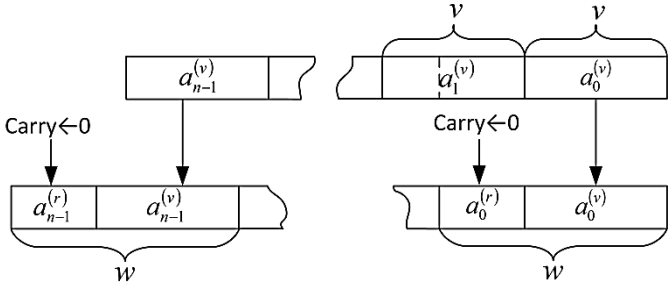

Рис. 2. Преобразование целого числа в DCF

Алгоритм 1. Алгоритм преобразования целых чисел к DCF представлению.

Вход: $a \in Z$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $\log_2 a_i = v$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса.

Выход: $d = [a]_{DCF}$, $d \in Z_{DCF}$, $d = \{d_{n-1}, \ldots, d_1, d_0\}$, $\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $a_i^{(r)} \leftarrow 0$.

1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)}$.

2. Return (d).

2. Сложение

Алгоритм сложения целых чисел, в котором слагаемые представляются в DCF, и сумма также представляется в DCF.

Исходя из особенностей представления слагаемых $a_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)}$, $b_i^{(w)} \leftarrow b_i^{(r)} \| b_i^{(v)}$ и суммы $d_i^{(w)} \leftarrow d_i^{(r)} \| d_i^{(v)}$, классическая запись сложения чисел в DCF, рис. 3, может быть представлена в упрощенной форме рис. 4.

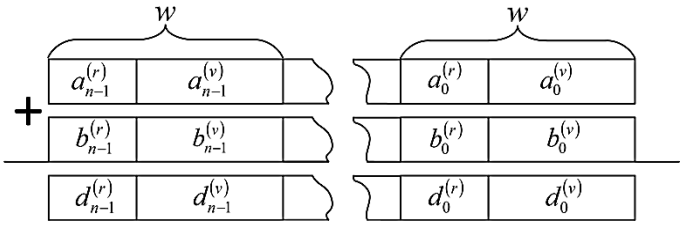

Рис. 3. Сложение чисел в DCF

Данный алгоритм является наиболее простым, среди предложенных алгоритмов сложения, т.к. не оперирует битами, в том числе и переносами: $d_i^{(r)} \| d_i^{(v)} \leftarrow a_i^{(r)} \| a_i^{(v)} + b_i^{(r)} \| b_i^{(v)}$. В упрощенной форме, рис. 4, складываются сразу машинные слова $d_i^{(w)} \leftarrow a_i^{(w)} + b_i^{(w)}$, без выделения областей хранения самого числа и переноса, что позволяет достичь максимальной производительности при его реализации на современных процессорах. Процессоры оперируют исключительно машинными словами, при этом перенос из младшего машинного слова в старшее не выполняется.

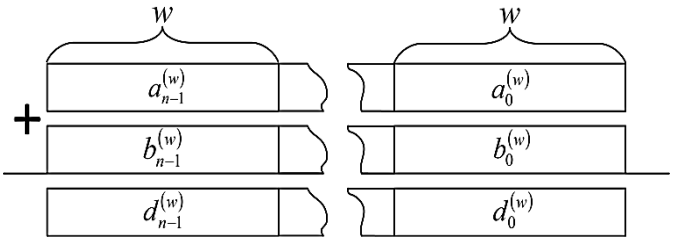

Рис. 4. Упрощенная форма сложение чисел в DCF

Алгоритм 2. Алгоритм сложения целых чисел с отложенным переносом.

Вход: $a, b \in Z_{DCF}$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = \log_2 b_i = w$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса.

Выход: $d = [a+b]_{DCF}$, $d \in Z_{DCF}$, $d = \{d_{n-1}, \ldots, d_1, d_0\}$, $\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $d_i^{(w)} \leftarrow a_i^{(w)} + b_i^{(w)}$.

2. Return (d).

3. Смешанное сложение

Рассмотрим алгоритм сложения, в котором слагаемые представляются в привычной двоичной (непрерывной) форме, а сумма – в DCF. Отличием алгоритма смешанного сложения, от привычного алгоритма сложения, состоит в том, что в процессе сложения слагаемые преобразуются в DCF, в соответствии с Алгоритмом 1, и лишь затем производится их сложение.

Исходя из особенностей представления слагаемых $a_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)}$, $b_i^{(w)} \leftarrow b_i^{(r)} \| b_i^{(v)}$, где области переносов инициализируются следующим образом $a_i^{(r)} \leftarrow 0$, $b_i^{(r)} \leftarrow 0$ и суммы $d_i^{(w)} \leftarrow d_i^{(r)} \| d_i^{(v)}$, классическая запись смешанного сложения DCF чисел – представлена на рис. 5.

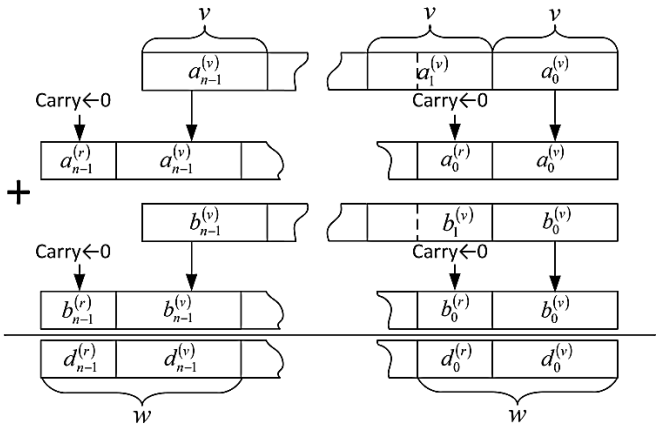

Рис. 5. Смешанное сложение чисел

Алгоритм 3, отличается от Алгоритма 2, тем, что производится выделение v-бит из каждого слагаемого в привычном двоичном непрерывном представлении и заполнение оставшихся r-бит – нулями.

Данное преобразование является довольно трудоемким при программной реализации, что требует выбирать область для хранения переносов длиной r-бит, кратной байту.

Алгоритм 3. Алгоритм смешанного сложения целых чисел с отложенным переносом.

Вход: $a, b \in Z$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = \log_2 b_i = v$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$,
$d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$,
$\log_2 d_i = w$, $i = \overline{0, n-1}$.
1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.
1.1. $a_i^{(r)} \leftarrow 0$, $b_i^{(r)} \leftarrow 0$.
1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)} + b_i^{(r)} \| b_i^{(v)}$.
2. Return (d).

Так же, следует рассмотреть алгоритм, в котором в качестве слагаемых используется DCF число и число в непрерывной двоичной форме, а сумма – в DCF представлении.

Исходя из особенностей представления слагаемых $a_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)}$ и $b_i^{(w)} \leftarrow b_i^{(r)} \| b_i^{(v)}$, где область переноса одного из слагаемых в двоично-непрерывной форме инициализируется следующим образом $b_i^{(r)} \leftarrow 0$ и суммы $d_i^{(w)} \leftarrow d_i^{(r)} \| d_i^{(v)}$, классическая запись сложения DCF чисел, см. рис. 4, может быть представлена в упрощенной форме (рис. 6).

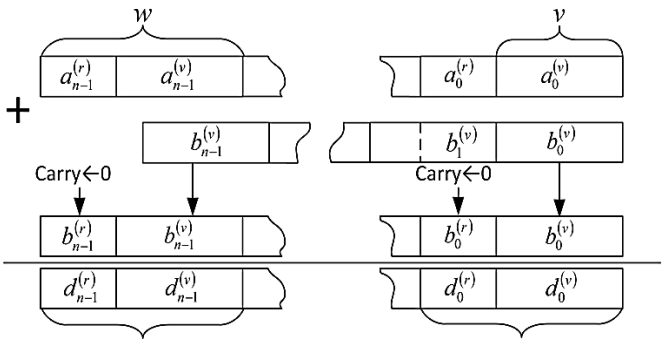

Рис. 6. Упрощенная форма смешанного сложения

Алгоритм 4. Алгоритм смешанного сложения целого числа и DCF числа с отложенным переносом.

Вход: $a \in Z_{DCF}$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b \in Z$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = w$, $\log_2 b_i = v$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$,
$d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$,
$\log_2 d_i = w$, $i = \overline{0, n-1}$.
1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.
1.1. $b_i^{(r)} \leftarrow 0$.
1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)} + b_i^{(r)} \| b_i^{(v)}$.
2. Return (d).

4. Вычитание

Второй по значимости операции над целыми числами, после сложения, является вычитание. При вычитании, перенос (заем) производится не из младших разрядов в старшие, а наоборот – из старших разрядов в младшие. При вычитании, значение переноса может становится отрицательным.

Алгоритм вычитания целых чисел, в котором уменьшаемое, вычитаемое представляются в DCF представлении, и разность также представляется в DCF.

Исходя из особенностей представления уменьшаемого $a_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)}$, вычитаемого $b_i^{(w)} \leftarrow b_i^{(r)} \| b_i^{(v)}$ и разности $d_i^{(w)} \leftarrow d_i^{(r)} \| d_i^{(v)}$, классическая запись вычитания чисел в DCF-

представлении, см. рис. 7, может быть представлена в упрощенной форме рис. 8.

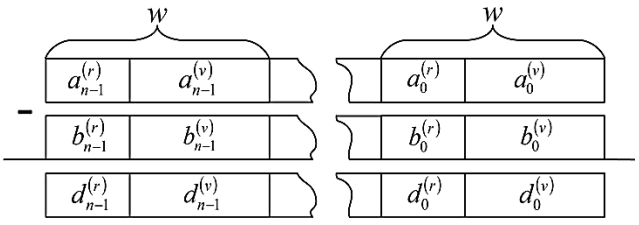

Рис. 7. Вычитание DCF-чисел

Данный алгоритм является наиболее простым, среди предложенных алгоритмов вычитания, т.к. не оперирует битами, в том числе и с заемами: $d_i^{(r)} \| d_i^{(v)} \leftarrow a_i^{(r)} \| a_i^{(v)} - b_i^{(r)} \| b_i^{(v)}$, В упрощенной форме, рис. 8, вычитаются сразу машинные слова $d_i^{(w)} \leftarrow a_i^{(w)} - b_i^{(w)}$, без выделения областей хранения самого числа и заема, что позволяет достичь максимальной производительности при его реализации на современных процессорах. Процессоры оперируют исключительно машинными словами, при этом заем из старшего машинного слова в младшее не выполняется.

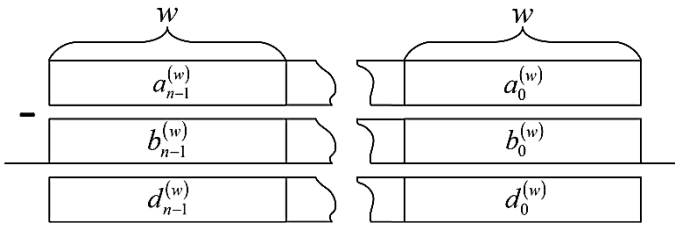

Рис. 8. Упрощенная форма вычитания DCF-чисел

Алгоритм 5. Алгоритм вычитания целых чисел с отложенным заемом.

Вход: $a, b \in Z_{DCF}$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = \log_2 b_i = w$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса, может обладать отрицательным знаком.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$,

$d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$,

$\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $d_i^{(w)} \leftarrow a_i^{(w)} - b_i^{(w)}$.

2. Return (d).

5. Смешанное вычитание

Существует три варианта смешанного вычитания чисел.

Рассмотрим алгоритм вычитания, в котором уменьшаемое и вычитаемое представляются в привычной двоичной (непрерывной) форме, а разность – в DCF-представлении (рис. 9, Алгоритм 6). Отличием алгоритма смешанного вычитания, от привычного алгоритма вычитания, состоит в том, что в процессе вычитания, уменьшаемое и вычитаемое преобразуются в DCF, в соответствии с Алгоритмом 1, и лишь затем производится их вычитание.

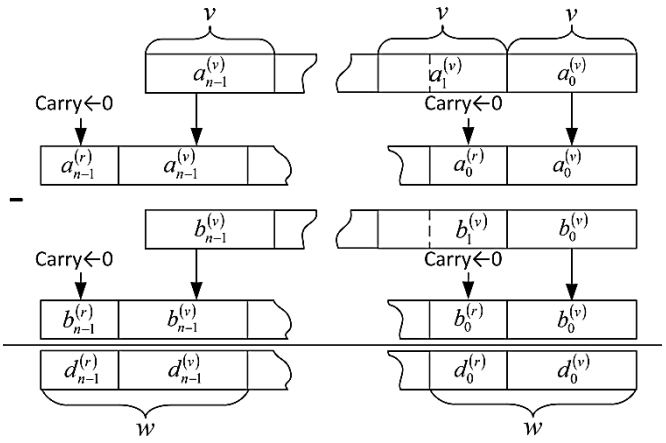

Рис. 9. Смешанное вычитание двоичных чисел в DCF-представлении

Алгоритм 6. Алгоритм смешанного вычитания DCF целых чисел с отложенным переносом.

Вход: $a, b \in Z$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = \log_2 b_i = v$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса, может обладать отрицательным знаком.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$,

$d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$,

$\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $a_i^{(r)} \leftarrow 0$, $b_i^{(r)} \leftarrow 0$.

1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)} - b_i^{(r)} \| b_i^{(v)}$.

2. Return (c).

Рассмотрим алгоритм вычитания, в котором уменьшаемое представлено в DCF, а вычитаемое представляется в двоичной форме (рис. 10, Алгоритм 7). Разность представляется в DCF. Отличием от предыдущего алгоритма смешанного вычитания, состоит в том, что в процессе вычитания, вычитаемое преобразуется в DCF, в соответствии с Алгоритмом 1, и лишь затем производится вычитание.

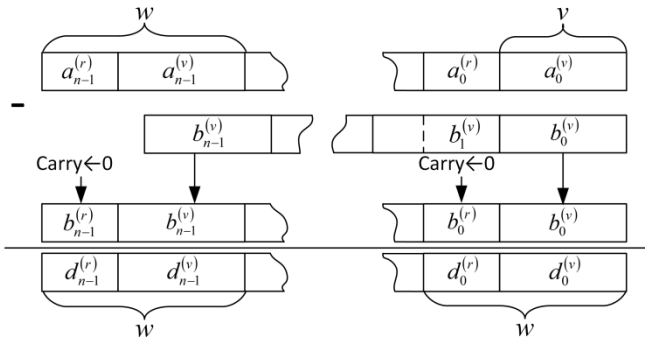

Рис. 10. Смешанное вычитание DCF-числа и двоичного числа

Алгоритм 7. Алгоритм смешанного вычитания DCF числа и целого числа с отложенным переносом.

Вход: $a \in Z_{DCF}$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b \in Z$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = w$, $\log_2 b_i = v$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса, может обладать отрицательным знаком.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$, $d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$, $\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $b_i^{(r)} \leftarrow 0$.

1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)} - b_i^{(r)} \| b_i^{(v)}$.

2. Return (d).

Рассмотрим алгоритм вычитания, в котором уменьшаемое представлено в двоичной форме, а вычитаемое представляется в DCF (рис. 11, Алгоритм 8). Разность представляется в DCF. В процессе вычитания, уменьшаемое преобразуется в DCF, в соответствии с Алгоритмом 1, и лишь затем производится вычитание.

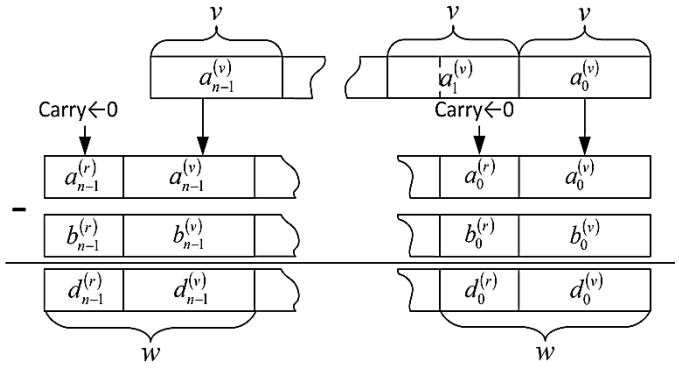

Рис. 11. Смешанное вычитание двоичного числа и DCF-числа

Алгоритм 8. Алгоритм смешанного вычитания DCF числа и целого числа с отложенным переносом.

Вход: $a \in Z$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $b \in Z_{DCF}$, $b = \{b_{n-1}, \ldots, b_1, b_0\}$, $\log_2 a_i = v$, $\log_2 b_i = w$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса, может обладать отрицательным знаком.

Выход: $d = [a + b]_{DCF}$, $d \in Z_{DCF}$, $d = \{d_{n-1}, \ldots, d_1, d_0\}$, $d_i^{(w)} = d_i^{(r)} \| d_i^{(v)}$, $\log_2 d_i = w$, $i = \overline{0, n-1}$.

1. For $i \leftarrow 0$, $i < n$, $i{+}{+}$.

1.1. $a_i^{(r)} \leftarrow 0$.

1.2. $d_i^{(w)} \leftarrow a_i^{(r)} \| a_i^{(v)} - b_i^{(r)} \| b_i^{(v)}$.

2. Return (d).

6. Корректировка переносов

После выполнения операций сложения или вычитания необходимо учесть отложенные переносы. Для этого для всех машинных слов DCF числа $a \in Z_{DCF}$ необходимо выполнить перенос $a_i^{(r)}$ с младшего машинного слова $a_i^{(w)}$ в старшее $a_{i+1}^{(w)}$. При этом из полученной последовательности блоков длины v-бит $d = \{d_{n-1}^{(v)}, \ldots, d_1^{(v)}, d_0^{(v)}\}$, таких что $d_i^{(v)} \leftarrow t^{(v)}$, $t^{(r)} \| t^{(v)} \leftarrow a_i^{(w)} + t^{(r)}$, необходимо сформировать последовательность машинных слов длинной w-бит, рис. 12. Фактически коррекция переносов является преобразованием DCF числа в двоичную непрерывную форму.

Алгоритм 9. Алгоритм корректировки переносов DCF числа.

Вход: $a \in Z_{DCF}$, $a = \{a_{n-1}, \ldots, a_1, a_0\}$, $\log_2 a_i = w$, $i = \overline{0, n-1}$, $w = v + r$, w-длина машинного слова, v – двоичная длина числа, r – двоичная длина переноса, может обладать отрицательным знаком.

Выход: $d = Z$, $d = \{d_{l-1}, \ldots, d_1, d_0\}$, $\log_2 d_i = w$, $i = \overline{0, l-1}$, $l = \lceil \frac{nv}{w} \rceil$.

1. $t^{(r)} \leftarrow a_0^{(r)}$, $d_0^{(v)} \leftarrow a_0^{(v)}$

2. For $i \leftarrow 1$, $i < n$, $i{+}{+}$.

2.1. $t^{(r)} \| t^{(v)} \leftarrow a_i^{(w)} + t^{(r)}$ //

$a_i'^{(r)} \| a_i'^{(v)} \leftarrow a_i^{(w)} + t^{(r)}$

2.2. $j \leftarrow \lfloor \frac{i \cdot v}{w} \rfloor$.

2.3. $s \leftarrow (r \cdot i) \bmod w$.

2.4. $d_j^{(w)} \leftarrow d_j^{(w)} | \left(t^{(v)} >> s\right)$.

2.5. if $(s > 0)$ then

$d_{j+1}^{(w)} \leftarrow d_{j+1}^{(w)} | \left(t^{(v)} << (w - s)\right)$.

3. Return (d).

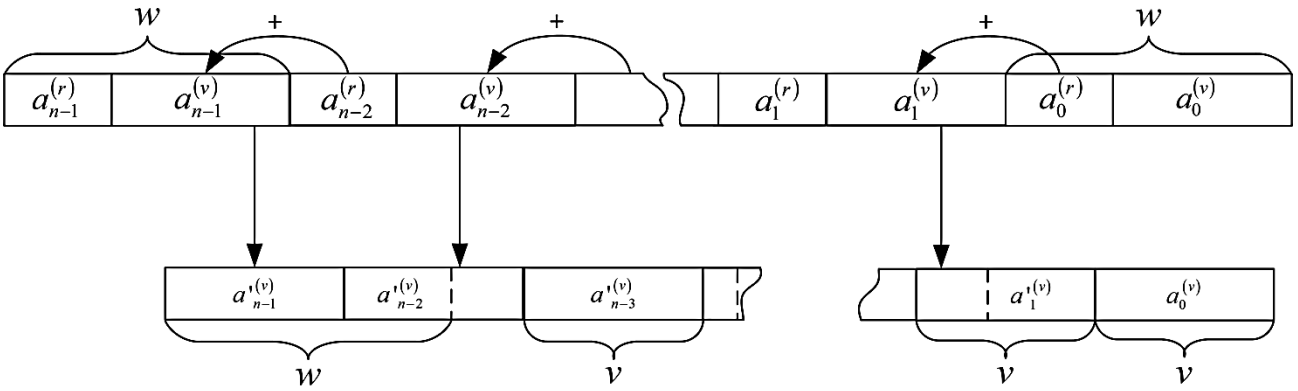

Рис. 12. Коррекция переносов

Рекомендации по программной реализации и распаралелливанию

Эффективность программной реализации зависит от разбиения машинного слова длиной w-бит на блок переноса длиной r-бит и информационный блок длиной v-бит. Это соотношение влияет на:

- объем сохраняемых переносов и заемов;
- избыточность представления целых чисел;
- производительность.

Количество бит r, выделенных под хранение переноса, следует определять из соображений производительности – выравнивание по границе одного, двух и более байт, исходя из соображений максимально возможного хранения переносов. Так, для хранения переносов в диапазоне {-127, 128}, которые могут быть накоплены в результате 128 операций сложения и 127 операций вычитания, либо 255 операций сложения/вычитания, необходимо выделить 1 байт под перенос. При большом числе операций, которые могут привести к переполнению блока переносов, рекомендуется выполнять промежуточную корректировку переносов (учет переносов), при достижении возможной границы.

При использовании такого подхода появляется избыточность представления целых чисел в DCF представлении, за счет выделения r бит в каждом машинном слове под перенос или заем. Аналитическая оценка избыточности может быть представлена в (3) или (4).

$$R\left(d_{DCF}\right)=\left\lceil w\cdot n\cdot\left(\frac{w}{w-r}-1\right)\right\rceil \qquad (3)$$

или

$$R\left(d_{DCF}\right)=\left\lceil w\cdot n\cdot\left(\frac{w}{v}-1\right)\right\rceil. \qquad (4)$$

В таблице 1 показаны оценки избыточности DCF-представления в битах, для различной длины машинных слов n при использовании блоков отложенного переноса разных размеров.

Таблица 1

Зависимость избыточности представледставления целых чисел от размера блока отложенного переноса

Число бит для отложенного переноса	Возможность использования		Избыточность, байт	
	w = 32 бит	w = 4 бит	w = 32 бит	w = 64 бит
8	+	+	4/3 n	8/7 n
16	+	+	4 n	8/3 n
24	+	+	12 n	24/5 n
32	-	+	-	8 n

Знаками «+» помечены длины машинных слов, которые поддерживают блоки переносов заданного размера в битах.

Особенностью предлагаемого DCF представления целых чисел, является отсутствие необходимости учета переносов и заемов, что позволяет избавиться от лишних операций присвоения и проверок при реализации на языках высокого уровня, а также от анализа регистра флагов на возможный перенос. В свою очередь, это приводит к повышению эффективности программной реализации на процессорах с суперскалярной архитектурой и возможностями современных компиляторов по предсказанию переходов, параллельному выполнению команд, разворачиванию циклов и т.д., в соответствии с [7].

Учитывая ранее описанную специфику DCF-арифметики, когда при сложении и вычитании чисел порядок выполнения операций над машинными словами не является существенным, само по себе напрашивается возможность параллельного выполнения операций сложения и вычитания над соответствующими словами.

В программной реализации возможно применять схему распараллеливания с использованием множества потоков, например, с использованием технологии OpenMP, или вплоть до выполнения каждой операции в отдельном потоке с использованием технологии NVIDIA CUDA, рис. 13.

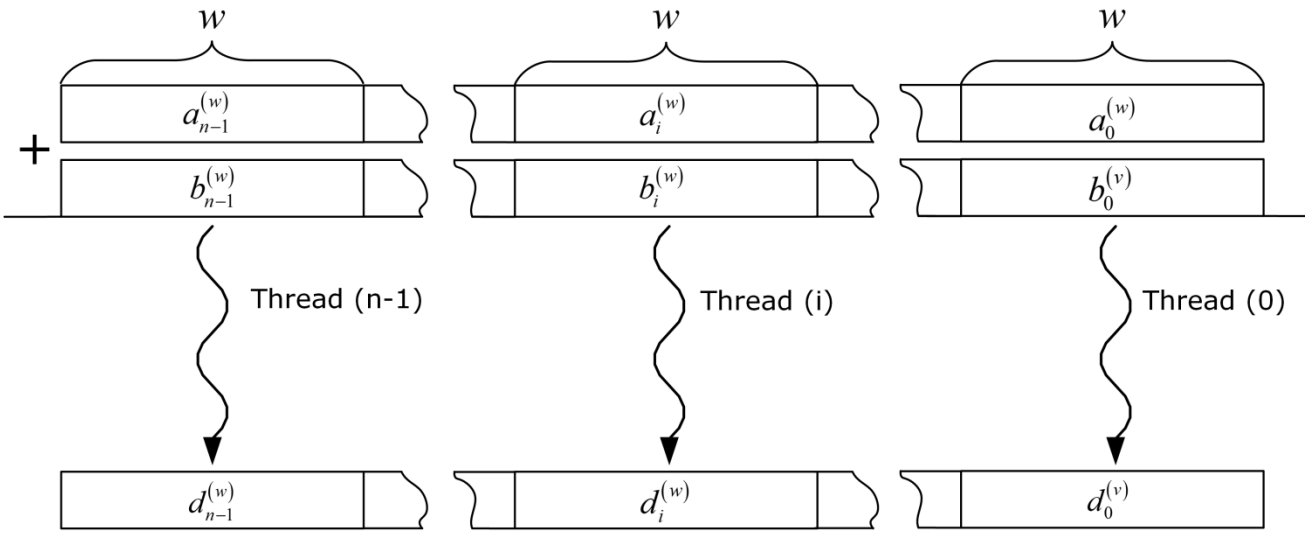

Рис. 13. Модель распараллеливания на основе технологий NVIDIA CUDA

Выводы. По результатам проведенных исследований можно сделать следующие выводы:

1. Предложено новое представление для целых чисел с отложенным переносом (DCF), для решения задачи повышения производительности криптографических преобразований с открытым ключом.

2. Разработаны новые алгоритмы арифметических операций над DCF числами – сложение и вычитание, перевод из двоичной формы в DCF и наоборот.

3. Исключены операции присвоения и проверок при программной реализации, благодаря отсутствию необходимости учета переносов и заемов в DCF представлении, что позволяет повысить производительность программной реализации на современных суперскалярных процессорах.

4. Показана возможность применения различных технологий распараллеливания программной реализация алгоритмов арифметических операций DCF чисел.

ЛИТЕРАТУРА

[1]. Умножения целых чисел с использованием отложеного переноса для криптосистем с открытым ключом / В.Ю.Ковтун, А.А.Охрименко [и др.] // Информационные технологи и системы в управлении, образовании, науке: Монография / Под ред. проф. В.С. Пономаренко. – Х.: Цифрова друкарня №1. – 2013. – С. 69-82.

[2]. Richard P. Brent and Paul Zimmermann. Modern Computer Arithmetic // Cambridge Monographs on Computational and Applied Mathematics (No. 18), Cambridge University Press, November 2010. 239 p.

[3]. Brumnik R., Kovtun V., Okhrimenko A. and Kavun S. Techniques For Performance Increasing Of Integer Multiplications In Cryptographic Application. – Mathematical Problems in Engineering. – vol. 2014. – 2014. – p.7. – doi:10.1155/2014/863617

[4]. Cohen H. and Frey G., editors. Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications. – Chapman & Hall/CRC. – 2006. – p. 848.

[5]. Hankerson D., Menezes A., and Vanstone S.A. Guide to Elliptic Curve Cryptography, Springer-Verlag, – 2004. – p. 332.

[6]. Marc Joyel, Christophe Tymen. Compact Encoding of Non-Adjacent Forms with Applications to Elliptic Curve Cryptography // Published In K.Kim, Ed., Public Key Cryptography, vol. 1992 of LNCS, – Springer-Verlag. – 2001. – pp. 353-364.

[7]. Intel® 64 and IA-32 Architectures Optimization Reference Manual, available at: http://www.intel.com/ content / www / us/ en / architecture-and-technology / 64-ia-32-architectures-optimization-manual.html (accessed 15 May 2014)

[8]. Knuth, Donald E. The Art of Computer Programming. Third edn. Vol.2 : Seminumerical Algorithms. - Addison-Wesley. – 1998. – 762p.

[9]. Patrick Longa, Ali Miri New Multibase Non-Adjacent Form Scalar Multiplication and its Application to Elliptic Curve Cryptosystems (extended version) // Cryptological ePrint Archive. – Report 2008/52. – 2008. – p.39. – URL: http://eprint.iacr.org/2008/052.pdf

[10]. Franco P. Preparata. On the Representation of Integers in Nonadjacent Form // SIAM Journal on Applied Mathematics. – Vol. 21. – No. 4. – 1971. – pp. 630-635.

[11]. Yanik T., Savas E., and Koc C. K. Incomplete Reduction in Modular Arithmetic. – IEEE Proceedings – Computers and Digital Techniques. – 149(2). – 2002. – pp. 46-52.

REFERENCES

[1]. Kovtun V., Okhrimenko A. (2013), Integer multiplication with delayed carry in public-key cryptosystems, in Ponomarenko V.S. (Ed.), Information technologies in management, education and science, Cifrova drukarnya №1, Kharkiv, pp. 69-82.

[2]. Richard P. Brent and Paul Zimmermann (2010). Modern Computer Arithmetic, Cambridge Monographs on Computational and Applied Mathematics (No. 18), Cambridge University Press, 239 p.

[3]. Brumnik R., Kovtun V., Okhrimenko A. and Kavun S (2014). Techniques For Performance Increasing Of Integer Multiplications In Cryptographic Application. – Mathematical Problems in Engineering, vol. 2014, p.7. doi:10.1155/2014/863617

[4]. Cohen H. and Frey G. (2006). Handbook of Elliptic and Hyperelliptic Curve Cryptography. Discrete Mathematics and its Applications. Chapman & Hall/CRC.

[5]. Hankerson D., Menezes A., and Vanstone S.A. (2004), Guide to Elliptic Curve Cryptography, Springer-Verlag.

[6]. Marc Joyel, Christophe Tymen (2001). Compact Encoding of Non-Adjacent Forms with Applications to Elliptic Curve Cryptography // Published In K.Kim, Ed., Public Key Cryptography, vol. 1992 of LNCS, pp. 353-364, Springer-Verlag.

[7]. Intel® 64 and IA-32 Architectures Optimization Reference Manual, available at: http://www.intel.com / content / www / us / en / architecture-and-technology/64-ia-32-architectures-optimization-manual.html.

[8]. Knuth, Donald E. (1998), The Art of Computer Programming. Third edn. Vol.2: Seminumerical Algorithms, Addison-Wesley.

[9]. Patrick Longa, and Ali Miri (2008), New Multibase Non-Adjacent Form Scalar Multiplication and its Application to Elliptic Curve Cryptosystems (extended version), Cryptological ePrint Archive, Report 2008/52, available at: http://eprint.iacr.org /2008/052.pdf.

[10]. Franco P. Preparata (1971). On the Representation of Integers in Nonadjacent Form, SIAM Journal on Applied Mathematics, Vol. 21 (4). pp. 630-635.

[11]. Yanik T., Savas E., and Koc C. K. (2002), Incomplete Reduction in Modular Arithmetic, Computers and Digital Techniques, 149(2), pp. 46-52.

АРИФМЕТИКА З ВІДКЛАДЕНИМ ПЕРЕНОСОМ ДЛЯ ЦІЛИХ ЧИСЕЛ

Криптографічні перетворення з відкритим ключем широко використовуються і покладені в основу направленого шифрування, вироблення спільного секрету та електронного цифрового підпису. Тому, завдання підвищення продуктивності криптографічних перетворень з відкритим ключем є актуальним. Підвищити продуктивність можна за рахунок збільшення продуктивності операцій над цілими числами. Пропонується DCF представлення цілих чисел, в якому число розбивається на машинні слова, де в кожному машинному слові відводиться блок під представлення самого числа і блок під подальші переноси в старші розряди, або позики зі старших розрядів. Наводяться алгоритми основних арифметичних операцій з відкладеним перенесенням, даються рекомендації щодо ефективної програмної реалізації арифметичних операцій (додавання, віднімання).

Ключові слова: DCF представлення, відкладене перенесення, цілі числа, цілочисельна арифметика, програмна реалізація, розпаралелювання.

INTEGER ARITHMETIC WITH DELAYAED CARRY

Cryptographic transformation with public key are widely used in directional encryption, shared secret generation and digital signature. Therefore, performance improving of cryptographic transformations with public key is actual problem. Performance improving of operations on integers is a potential solution to this problem. Author propose DCF representation of integers in which the number is split into machine words, where each word has information block and carry block. Author propose algorithms of basic arithmetic operations with delayed carry, provides recommendations for effective software implementation of arithmetic operations (addition, subtraction).

Keywords: DCF representation, delayed carry, integers, integer arithmetic, software implementation, parallelization.

Охріменко Андрій Олександрович, аспірант кафедри безпеки інформаційних технологій, Національний авіаційний університет.

E-mail: andrew.okhrimenko@gmail.com

Охрименко Андрей Александрович, аспирант кафедры безопасности информационных технологий, Национальный авиационный университет.

Okhrimenko Andrew, postgraduate student, Academic Department of IT-Security of National Aviation University.