# О параллельном вычислении булевых функций на нескольких независимых наборах и оптимизации параллельного доступа к криптографически защищенным базам данных

М.Н. Назаров

## 1. Введение

В работах [1] и [2] представлен параллельный алгоритм вычисления булевой функции на нескольких независимых наборах, который может быть интерпретирован как алгоритм параллельного извлечения информации из базы данных по нескольким независимым адресам. Исследован случай максимального распараллеливания алгоритмов при полиномиально ограниченном количестве процессоров. Главным ограничением является необходимость считывания всех запрашиваемых бит за одно обращение к внешнему носителю. Это ограничение связано с тем, что скорость считывания информации с любого носителя существенно ниже скорости работы процессора. Поэтому общее время доступа к базе данных может быть минимально при одном обращении к внешнему носителю, хотя это и потребует дополнительных вычислений.

Очевидно, что при хранении базы данных на одном носителе извлечение нескольких произвольных бит информации за одно обращение невозможно. Следовательно, необходимо разбить булеву функцию на части, каждая из которых будет храниться на своем носителе и обслуживаться отдельным процессором. Однако, простое разбиение не решит задачи, так как может потребоваться извлечь

несколько бит из одной части, что приведет к нарушению главного ограничения алгоритма. Поэтому в [1] и 2 были введены дополнительных булевы функций, расширяющие базу данных.

Вызывает интерес задача криптографической защиты базы данных при соблюдении всех описанных выше ограничений. Кроме того, интересен случай параллельного вычисления сразу нескольких булевых функций на независимых наборах и возможность криптографической защиты такой базы данных. Большое практическое значение имеет задача минимизации числа процессоров при соблюдении главного ограничения алгоритмов. Все указанные задачи были рассмотрены в данной работе.

## 2. Основные определения и известные ранее результаты

Рассмотрим специальную параллельную модель, описывающую некоторую абстрактную базу данных. Пусть вся хранимая информация может быть представлена в виде таблиц для булевых функций $n$ переменных $f(x_1, x_2, ..., x_n)$. Более точно, база данных это набор булевых функций реализованных таблицами, причем входные последовательности для каждой функции это некоторая фиксированная область в памяти. Процессоры могут различать индексы этих булевых функций и получать значение функции (один бит) на выходе. Другими словами, каждый набор значений переменных является адресом, по которому хранится один бит информации. Под доступом к базе данных понимается считывание информации с внешнего носителя по определенным адресам, что фактически означает вычисление булевой функции на заданных наборах переменных. Отметим, что под процессором в такой модели понимается булева схема постоянного размера, которая может выполнять запросы к внешней памяти (то есть базе данных), затрачивая специальное время, обозначенное **extime**.

Опишем некоторые предварительные результаты из [1, 2], учитывая, что сложность всех алгоритмов оценивается по трем параметрам:

**ptime** – время извлечения информации из базы данных;
**bp** – количество процессоров;
**mem** – размер базы данных.

Пусть $GF(q)$ – поле Галуа размерности $q = 2^k$, $(k \geqslant 0)$. Рассмотрим множество $GF(q)^4$ как четырехмерное линейное пространство. Через $l(A, u)$ обозначим прямую, проходящая через точку $A$ параллельно вектору $\vec{h} = (1, u, u^2, u^3)$. Если $U$ является подмножеством $GF(q)$ размерности $r^3$, то через $SL_4(U)$ обозначим множество всех прямых $l(A, u)$ таких, что $A \in GF(q)^4$ и $u \in U$. Кроме того, определим набор точек $l^{\#}(A, u) = l(A, u) \backslash A$. В работе [2] описан алгоритм $\mathcal{A}$, который для любой входной последовательности $A_1, A_2, \ldots, A_r$, где $A_i \in GF(q)^4$ возвращает последовательность $u_1, u_2, \ldots, u_r$, $(u_i \in U \subseteq GF(q))$ такую, что для любой пары $j_1$, $j_2$, $(j_1 \neq j_2)$ имеем

$$l^{\#}(A_{j1}, u_{j1}) \cap l^{\#}(A_{j2}, u_{j2}) = \varnothing.$$

Этот алгоритм нам понадобится позднее. Отметим только, что он имеет следующую сложность:

$$\mathbf{bp}(\mathcal{A}) = O(r^7 k^2), \qquad \mathbf{ptime}(\mathcal{A}) = O(\log r + \log k).$$

Перейдем теперь к описанию процедуры построения алгоритма $DP_{r,f}$ [1, 2] для параллельного вычисления булевой функции $f : \{0,1\}^n \to \{0,1\}$, на $r$ независимых наборах переменных $X_1$, $X_2, \ldots, X_r$.

Исходное пространство $\{0,1\}^n$ разобьем на два подпространства

$$\{0,1\}^n = \{0,1\}^{4k} \times \{0,1\}^{n-4k},$$

где $k = \lceil 3 \log r + \log n \rceil$. Тогда все входы $X_1$, $X_2$, . . . , $X_r$ можно представить в виде $X_i = A_i B_i$, $i = 1, 2, ..., r$, где $A_i \in \{0,1\}^{4k}$, а $B_i \in \{0,1\}^{n-4k}$. Первое подпространство рассмотрим как четырехмерное линейное пространство $GF(q)^4$, где $q = 2^k$.

Для всех $A \in GF(q)^4$ рассмотрим булевы функции $f_A : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $f_A(B) = f(A, B)$.

Для всех $l \in SL_4(U)$ рассмотрим булевы функции $g_l : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $g_l(B) = \bigoplus\limits_{A \in l} f_A$.

Пусть каждой функции $f_A$ и $g_l$ соответствует своя таблица, хранимая на отдельном внешнем носителе и обрабатываемая своим процессором. Совокупность этих функций представляют собой всю базу данных, то есть позволяют реализовать булеву функцию $f$. Процессор может обратиться к любой из этих функций, затратив некоторое время **extime**. Для наглядности можно объединить все эти таблицы в одну и получить представление функции $f$ (базы данных) в виде таблицы булевых функций от $n-4k$ переменных (табл. 1).

| $y_1$ | $y_2$ | . . . | $y_{n-4k}$ | $f_{A_1}$ | $f_{A_2}$ | . . . | $f_{A_m}$ | $g_1$ | $g_2$ | . . . | $g_p$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | . . . | 0 | * | * | . . . | * | * | * | . . . | * |
| 0 | 0 | . . . | 1 | * | * | . . . | * | * | * | . . . | * |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| 1 | 1 | . . . | 1 | * | * | . . . | * | * | * | . . . | * |

Таблица 1.

Здесь $m$ равняется количеству точек в первом подпространстве разбиения, то есть $m = |GF(q)^4| = 2^{4k}$, а $p$ равняется количеству прямых в множестве $SL_4(U)$, то есть $p = |SL_4(U)|$. Каждый столбец таблицы хранится на своем носителе и обслуживается отдельным процессором. Главное ограничение алгоритма означает, что для любых $r$ входов из каждого столбца таблицы можно считать только один бит.

Используем описанные функции для реализации функции $f$. Заметим, прежде всего, что

$$g_l(B) \oplus \left( \bigoplus_{A^* \in l \setminus \{A\}} f_{A^*}(B) \right) = \left( \bigoplus_{A \in l} f_A \right) \oplus \left( \bigoplus_{A^* \in l \setminus \{A\}} f_{A^*}(B) \right) = f_A(B).$$

При помощи алгоритма $\mathcal{A}(A_1, A_2, ..., A_r)$, упомянутого выше, можно найти $r$ элементов $(u_1, u_2, \ldots, u_r)$ из $U$ таких, что для любой пары $j_1 \neq j_2$

$$l^{\#}(A_{j_1}, u_{j_1}) \cap l^{\#}(A_{j_2}, u_{j_2}) = \varnothing.$$

Следовательно, можно построить следующую систему

$$f(A_i, B_i) = g_{l(A_i,u_i)}(B_i) \oplus \left( \bigoplus_{A^* \in l^{\#}(A_i,u_i)} f_{A^*}(B_i) \right), \quad i = 1, 2, \ldots, r.$$

Из свойств алгоритма $\mathcal{A}$ следует, что любая из функций $f_A$ или $g_l$ может участвовать только в одном уравнении последней системы. Именно для этого в [2] и был построен алгоритм $\mathcal{A}$. Иначе при пересечении прямых могла возникнуть ситуация, когда необходимо было бы вычислить два значения $f_A(B_1)$ и $f_A(B_2)$ для одной точки $A$, но на различных наборах $B_1$ и $B_2$, а это привело бы к нарушению главного ограничения алгоритма.

Теперь можно приступить к непосредственному описанию процедур алгоритма $DP_{r,f}$, следуя работе [1].

Введем следующие обозначения:

$$\vec{A} = (A_1, A_2, ..., A_r), \quad \vec{B} = (B_1, B_2, ..., B_r), \quad \vec{u} = (u_1, u_2, ..., u_r) = \mathcal{A}(\vec{A}).$$

Определим две вспомогательные процедуры:

$$\mathcal{B}_1(A, u, A^*) = \begin{cases} 1, & A^* \in l^{\#}(A, u); \\ 0, & A^* \notin l^{\#}(A, u). \end{cases}$$
$$\mathcal{B}_2(A, u, l) = \begin{cases} 1, & l = l(A, u); \\ 0, & l \neq l(A, u). \end{cases}$$

Далее построим еще две процедуры:

$$\mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u}) = \bigvee_{i=1}^{r} \mathcal{B}_1(A_i, u_i, A) \cdot B_i \ ,$$
$$\mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u}) = \bigvee_{i=1}^{r} \mathcal{B}_2(A_i, u_i, l) \cdot B_i \ .$$

Заметим, что две последние процедуры получают на выходе корректные входы для системы (2). Действительно, для любого $A \in GF(q)^4$, если $A \in l^{\#}(A_i, u_i)$, то $\mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u}) = B_i$ и $\mathcal{B}_4(l(A_i, u_i), \vec{A}, \vec{B}, \vec{u}) = B_i$. Введем обозначения

$$B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u}) \ \text{ и } \ B_l = \mathcal{B}_4(l(A_i, u_i), \vec{A}, \vec{B}, \vec{u}).$$

Определим переменные $Y_A = f_A(B_A)$, $A \in GF(q)^4$ и $Z_l = g_l(B_l)$, $l \in SL_4(U)$.

Пусть $\vec{Z}$ представляет собой набор переменных $Z_l$, где $l \in SL_4(U)$, а $\vec{Y}$ это набор переменных $Y_A$, где $A \in GF(q)^4$.

Построим процедуру

$$\mathcal{B}_6(A, u, \vec{Z}, \vec{Y}) = \left( \bigvee_{l \in SL_4(U)} Z_l \cdot \mathcal{B}_2(A, u, l) \right) \oplus \left( \bigoplus_{A^* \in GF(q)^4} Y_{A^*} \cdot \mathcal{B}_1(A, u, A^*) \right).$$

Заметим, что в первых скобках получаем $Z_{l(A,u)}$, а во вторых скобках получаем $\bigoplus\limits_{A^* \in l^{\#}(A,\, u)} Y_{A^*}$. Следовательно, мы имеем следующий результат

$$\mathcal{B}_6(A_i, u_i, \vec{Z}, \vec{Y}) = Z_{l(A_i,u_i)} \oplus \left( \bigoplus_{A \in\, l^{\#}(A_i,u_i)} Y_A \right) =$$

$$= g_{l(A_i,u_i)}(B_i) \oplus \left( \bigoplus_{A \in l^{\#}(A_i,u_i)} f_A(B_i) \right) = f(A_i, B_i).$$

Таким образом, получены все процедуры необходимые для описания алгоритма и решения поставленных задач.

**Алгоритм** $DP_{r,f}$:

**1.** Применим алгоритм $\mathcal{A}$ ко входу $(A_1, A_2, \ldots, A_r)$, то есть $\vec{u} = \mathcal{A}(\vec{A})$.

**2.** Применим процедуру $\mathcal{B}_5$, чтобы построить все корректные входы для булевых функций базы данных, то есть $\mathcal{B}_5$:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;

ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

**3.** Вычислим выходы булевых функций базы данных, то есть

i) для любых $A \in GF(q)^4$ вычислим $Y_A = f_A(B_A)$;

ii) для любых $l \in SL_4(U)$ вычислим $Z_l = g_l(B_l)$.

Заметим, что это единственный шаг на котором происходит обращение к внешним носителям, которое попарно независимо в силу применения алгоритма $\mathcal{A}$.

**4.** Применим процедуру $\mathcal{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathcal{B}_7$:

для любых $i = 1, 2, ..., r$ вычислим $f(A_i, B_i) = \mathcal{B}_6(A, u, \vec{Z}, \vec{Y})$.

## 3. Алгоритм параллельного доступа к криптографически защищенной базе данных и его сложность

Рассмотрим задачу криптографической защиты базы данных при соблюдении всех ограничений алгоритма $DP_{r,f}$. Как следует из предыдущего раздела, для этого необходимо криптографически защитить таблицу 1. Выбор конкретной криптосистемы выходит за рамки данной работы, поэтому для нас достаточно только предположить, что надежность используемой криптосистемы достаточно высока. Очевидно, что в случае шифрования столбцов нашей таблицы произойдет нарушение главного ограничения алгоритма, так как перед извлечением нужной информации потребуется извлечь некоторые столбцы целиком для их расшифровки.

Будем зашифровывать двоичные последовательности, соответствующие строкам таблицы 1. Таким образом, исходные сообщения для криптосистемы это двоичные последовательности длины $|GF(q)^4| + |SL_4(U)|$. Каждая строка заменяется на ее криптограмму. Зашифрование базы данных будем производить при ее предварительной обработке для алгоритма $DP_{r,f}$, поэтому в этой работе этот процесс рассматриваться не будет.

Нетрудно показать, что алгоритм $DP_{r,f}$ при своей работе для любых $r$ входов использует только соответствующие $r$ строк рассматриваемой таблицы. Следовательно, для корректной работы алгоритма необходимо расшифровать эти строки. Таким образом, за один шаг алгоритма потребуется считать $r$ бит из каждого столбца таблицы 1, что делает невозможным одно обращение к внешним носителям. Для устранения противоречия с главным ограничением вновь воспользуемся методами алгоритма $DP_{r,f}$. Будем считать, что нам необходимо вычислить значения каждой булевой функции $f_A$ и $g_l$ от $r$ независимых входов $B_1$, $B_2$, . . . , $B_r$. Тогда мы получим задачу, решаемую алгоритмом $DP_{r,f}$, и для ее решения необходимо специальным образом представить таблицу 1.

Каждый столбец, соответствующий $f_A$ и $g_l$, разобьем аналогично разбиению функции $f$, описанному во втором разделе. Для упроще-

ния обозначений положим $n' = n - 4k$.

Исходное пространство $\{0,1\}^{n'}$ представим в виде

$$\{0,1\}^{n'} = \{0,1\}^{4k'} \times \{0,1\}^{n'-4k'},$$

где $k' = \lceil 3\log r + \log n \rceil$. Первое подпространство рассмотрим как четырехмерное линейное пространство $GF(q')^4$, где $q' = 2^{k'}$. Все входы $B_1$, $B_2$, . . . , $B_r$ можно представить в виде $B_i = A'_i B'_i$, $i = 1, 2, ..., r$, где $A'_i \in \{0,1\}^{4k'}$, а $B'_i \in \{0,1\}^{n'-4k'}$. Через $l'(A', u')$ обозначим прямую, проходящая через точку $A'$ параллельно вектору $\vec{h} = (1, u', (u')^2, (u')^3)$. Если $U'$ является подмножеством $GF(q')$, то через $SL_4(U')$ обозначим множество всех прямых $l'(A', u')$ таких, что $A' \in GF(q')^4$ и $u' \in U'$.

Для всех $A' \in GF(q')^4$ рассмотрим булевы функции $f'_{A'} : \{0,1\}^{n'-4k'} \to \{0,1\}$ такие, что $f'_{A'}(B') = f_A(A', B') = f_A(B)$.

Для всех $l' \in SL_4(U')$ рассмотрим булевы функции $g'_{l'} : \{0,1\}^{n'-4k'} \to \{0,1\}$ такие, что $g'_{l'}(B') = \bigoplus\limits_{A' \in l'} f'_{A'}$.

Пусть каждой функции $f'_{A'}$ и $g'_{l'}$ соответствует своя таблица, обрабатываемая своим процессором.

Таким образом, каждый столбец таблицы 1 будет представлен в виде таблицы 2.

| $y'_1$ | $y'_2$ | . . . | $y'_{n'-4k'}$ | $f'_{A'1}$ | $f'_{A'2}$ | . . . | $f'_{A'm'}$ | $g'_{l'1}$ | $g'_{l'2}$ | . . . | $g'_{l'p'}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | . . . | 0 | * | * | . . . | * | * | * | . . . | * |
| 0 | 0 | . . . | 1 | * | * | . . . | * | * | * | . . . | * |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| 1 | 1 | . . . | 1 | * | * | . . . | * | * | * | . . . | * |

Таблица 2.

В этой таблице $m' = |GF(q')^4|$, $p' = |SL_4(U')|$. Каждый столбец, соответствующий какой-либо функции, обрабатывается отдельным процессором.

Таким образом последовательность шагов при криптографической защите базы данных следующая:

1) Представить базу данных в виде таблицы 1;

2) Зашифровать все строки таблицы 1;

3) Представить каждый столбец таблицы 1 в виде таблицы 2.

Опишем теперь алгоритм, организуюший доступ к зашифрованной базе данных. Фактически это алгоритм, вычисляющий булеву функцию от $r$ независимых входов и учитывающий ее криптографическую защиту. Назовем этот алгоритм $CDP_{r,f}$.

**Алгоритм** $CDP_{r,f}$

**1.** Применим процедуру, которую обозначим через $\mathcal{D}$, для считывания $r$ строк таблицы 1, соответствующих наборам $B_1$, $B_2$, . . . , $B_r$, из таблиц вида 2.

i) $\forall A \in GF(q)^4$ применим алгоритм $DP_{r,f_A}$ для вычисления каждой функции $f_A$ на входе $\vec{B} = (B_1, B_2, \ldots, B_r)$ и из результатов сформируем булеву функцию $F_A = (f_A(B_1), f_A(B_2), \ldots, f_A(B_r))$.

ii) $\forall l \in SL_4(U)$ применим алгоритм $DP_{r,g_l}$ для вычисления каждой функции $g_l$ на входе $\vec{B} = (B_1, B_2, \ldots, B_r)$ и из результатов сформируем булеву функцию $G_l = (g_l(B_1), g_l(B_2), \ldots, g_l(B_r))$.

Полученные результаты объединим в виде таблицы из $r$ строк, соответствующих наборам $B_1$, $B_2$, . . . , $B_r$ (табл. 3).

| $y_1$ | $y_2$ | . . . | $y_{n-4k}$ | $F_{A_1}$ | $F_{A_2}$ | . . . | $F_{A_m}$ | $G_{l_1}$ | $G_{l_2}$ | . . . | $G_{l_p}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| набор $B_1$ | | | | * | * | . . . | * | * | * | . . . | * |
| набор $B_2$ | | | | * | * | . . . | * | * | * | . . . | * |
| . . . | | | | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| набор $B_r$ | | | | * | * | . . . | * | * | * | . . . | * |

Таблица 3.

Данная таблица представляет собой $r$ зашифрованных строк таблицы 1, соответствующих наборам $B_1$, $B_2$, . . . , $B_r$.

**2.** Применим процедуру, которую обозначим через $\mathcal{C}$, для расшифровки всех строк таблицы 3. Расшифрованные строки опять разместим в виде таблице 3.

**3.** Применим процедуру $\mathcal{A}$ к входному вектору $\vec{A}$, то есть $\vec{u} = \mathcal{A}(\vec{A})$.

**4.** Применим процедуру $\mathcal{B}_5$, для построения правильных входов булевых функций базы данных, то есть $\mathcal{B}_5$:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;
ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

**5.** Вычислим все функции, необходимые для дальнейшей работы алгоритма
i) для любых $A \in GF(q)^4$ вычислим $Y_A = F_A(B_A)$;
ii) для любых $l \in SL_4(U)$ вычислим $Z_l = G_l(B_l)$.

Отметим, что на этом шаге не осуществляется обращение к базе данных. Все необходимые значения выбираются в таблице 3.

**6.** Применим процедуру $\mathcal{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathcal{B}_7$:
для любых $i = 1, 2, ..., r$ вычислим $f(A_i, B_i) = \mathcal{B}_6(A, u, \vec{Z}, \vec{Y})$.

Оценим сложность алгоритма $CDP_{r,f}$ по трем параметрам:
1. Время параллельных вычислений – **ptime**.
2. Количество процессоров – **bp**.
3. Размер базы данных – **mem**.

Учитывая, что $U$ подмножество $GF(q)^4$ размерности $r^3$ и $k = \lceil 3\log r + \log n \rceil$, можно доказать следующую теорему:

**Теорема 1.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f$ от $n$ переменных выполнено:*
$\mathbf{ptime}(CDP_{r,f}) = O(\log r + \log n) + \mathbf{extime} + \mathbf{crypttime}$,
$\mathbf{bp}(CDP_{r,f}) = O((r^{25}n^4)(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2)$,
$\mathbf{mem}(CDP_{r,f}) \leqslant 2^n \left(1 + \frac{1}{n}\right)\left(1 + \frac{1}{n - 12\log r - 4\log n}\right)$.

**Доказательство.** Докажем приведенные оценки, используя оценки для алгоритма $DP_{r,f}$ из [1, 2]. Рассмотрим первый шаг алгоритма. Для каждого конкретного $A \in GF(q)^4$ мы имеем
$\mathbf{bp}(DP_{r,f_A}) = O((r^{13}(n')^5)(\log r + \log n')^2)$,
$\mathbf{ptime}(DP_{r,f_A}) = O(\log r + \log n') + \mathbf{extime}$.
Для каждого конкретного $l \in SL_4(U)$:
$\mathbf{bp}(DP_{r,g_l}) = \mathbf{bp}(DP_{r,f_A}) = O((r^{13}(n')^5)(\log r + \log n')^2)$,
$\mathbf{ptime}(DP_{r,g_l}) = \mathbf{ptime}(DP_{r,f_A}) = O(\log r + \log n') + \mathbf{extime}$.

Учитывая, что $|GF(q)^4| = 2^{4k}$ и $|SL_4(U)| = \frac{|GF(q)^4| \cdot |U|}{|GF(q)|} = 2^{4k} \cdot \frac{|U|}{2^k} \leqslant 2^{4k} \cdot \frac{1}{n}$, получаем

$\mathbf{bp}(\mathcal{D}) = (|GF(q)^4|+|SL_4(U)|)\cdot\mathbf{bp}(DP_{r,f_A}) = O(2^{4k}\left(1+\frac{1}{n}\right)r^{13}(n')^5(\log r+\log n')^2)$, где $k=\lceil 3\log r+\log n\rceil$, $n'=n-4k$. Таким образом, $\mathbf{bp}(\mathcal{D}) = O(2^{4k}\left(1+\frac{1}{n}\right)r^{13}(n')^5(\log r+\log n')^2) = O(r^{12}n^4\left(1+\frac{1}{n}\right)r^{13}(n-12\log r-4\log n)^5(\log r+\log(n-k))^2) = O(r^{25}n^4(n-12\log r-4\log n)^5(\log r+\log(n-12\log r-4\log n))^2)$,
$\mathbf{ptime}(\mathcal{D}) = O(\log r+\log n')+\mathbf{extime} = O(\log r+\log(n-k))+\mathbf{extime} = O(\log r+\log(n-12\log r-4\log n))+\mathbf{extime}$.

На втором шаге мы расшифровываем строки таблицы 3. Будем считать, что для этого нам не требуется новых процессоров и эта процедура выполняется за время **crypttime**. Точная величина **crypttime** зависит от используемой криптосистемы и возможности ее распараллеливания. Получаем, что

$\mathbf{bp}(\mathcal{C}) = \mathbf{bp}(\mathcal{D}), \quad \mathbf{ptime}(\mathcal{C}) = \mathbf{crypttime}$.

Для третьего шага алгоритма, согласно оценкам для $DP_{r,f}$, имеем

$\mathbf{bp}(\mathcal{A}) = O(r^7k^2), \quad \mathbf{ptime}(\mathcal{A}) = O(\log r+\log k)$.

Для четвертого шага из [2] следует

$\mathbf{bp}(\mathcal{B}_5) = O(|GF(q)^4|\cdot(rk^2+rn))+O(|SL_4(U)|\cdot rk+rn))$,

$\mathbf{ptime}(\mathcal{B}_5) = O(\log k+\log r)$.

Учитывая, что $|GF(q)^4| = 2^{4k}$ и $|SL_4(U)| \leqslant 2^{4k}\cdot\frac{1}{n}$, получаем

$\mathbf{bp}(\mathcal{B}_5) = O(r2^{4k}\cdot(k^2+n))+O(r2^{4k}\cdot\left(1+\frac{k}{n}\right))$.

На пятом шаге потребуется обратиться к каждому столбцу таблицы 3, хранящейся в оперативной памяти. Так как каждый столбец обрабатывается своим процессором, то можно оценить время как $O(1)$. Количество процессоров не увеличится.

На шестом шаге алгоритма $CDP_{r,f}$ имеем

$\mathbf{bp}(\mathcal{B}_7) = r\cdot\mathbf{bp}(\mathcal{B}_6), \quad \mathbf{ptime}(\mathcal{B}_7) = \mathbf{ptime}(\mathcal{B}_6)$, где

$\mathbf{bp}(\mathcal{B}_6) = O(|SL_4(U)|\cdot k+|GF(q)^4|\cdot k^2) = O(k2^{4k}\cdot\left(k+\frac{1}{n}\right))$,

$\mathbf{ptime}(\mathcal{B}_6) = O(k)$.

Из всего вышеизложенного следуют оценки для алгоритма $CDP_{r,f}$.

$\mathbf{bp}(CDP_{r,f}) = \max(\mathbf{bp}(\mathcal{D}),\mathbf{bp}(\mathcal{C}),\mathbf{bp}(\mathcal{A}),\mathbf{bp}(\mathcal{B}_5),\mathbf{bp}(\mathcal{B}_7)) = O((r^{25}n^4)(n-12\log r-4\log n)^5(\log r+\log(n-12\log r-4\log n))^2)$.

$\mathbf{ptime}(CDP_{r,f}) = \mathbf{ptime}(\mathcal{D})+\mathbf{ptime}(\mathcal{C})+\mathbf{ptime}(\mathcal{A})+\mathbf{ptime}(\mathcal{B}_5)+\mathbf{ptime}(\mathcal{B}_7) = O(\log r+\log(n-12\log r-4\log n))+\mathbf{extime}+\mathbf{crypttime}+O(\log r+\log k)+O(k) = O(\log r+\log n)+\mathbf{extime}+\mathbf{crypttime}$.

Для оценки величины $\mathbf{mem}(CDP_{r,f})$ вспомним, что каждый столбец таблицы 1 разбивается по алгоритму $DP_{r,f}$. Учитывая, что

$m = |GF(q)^4|$, $p = |SL_4(U)|$ и $\mathbf{mem}(DP_{r,f_A}) \leqslant 2^{n'} \cdot \left(1 + \frac{1}{n'}\right)$, получаем

$\mathbf{mem}(CDP_{r,f}) \leqslant (|GF(q)^4| + |SL_4(U)|) \cdot 2^{n'} \cdot \left(1 + \frac{1}{n'}\right) \leqslant 2^{4k}\left(1 + \frac{1}{n}\right) \cdot 2^{n-4k}\left(1 + \frac{1}{n-4k}\right) = 2^n\left(1 + \frac{1}{n}\right) \cdot \left(1 + \frac{1}{n-4k}\right) = 2^n\left(1 + \frac{1}{n}\right) \cdot \left(1 + \frac{1}{n - 12\log r - 4\log n}\right)$.

Что и требовалось доказать.

## 4. Параллельный алгоритм одновременного вычисления $s$ булевых функций и его сложность

Рассмотрим задачу о вычислении $s$ булевых функций $f_1, f_2, \ldots, f_s$, где $f_i : \{0,1\}^n \to \{0,1\}$, $i = 1, 2, ..., s$, от $r$ независимых входов $X_1$, $X_2$, . . . , $X_r$. Все функции реализованы таблицами и представляют собой базу данных. Последовательность бит ($f_1$, $f_2$, . . . , $f_s$) представляет собой слово (фразу), хранимую в базе данных. Таким образом, мы имеем задачу организации доступа к базе данных по нескольким независимым адресам. Первоначальную базу данных можно представить в виде следующей таблицы.

| $x_1$ | $x_2$ | . . . | $x_{n-1}$ | $x_n$ | $f_1$ | $f_2$ | . . . | $f_s$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | . . . | 0 | 0 | * | * | . . . | * |
| 0 | 0 | . . . | 0 | 1 | * | * | . . . | * |
| 0 | 0 | . . . | 1 | 0 | * | * | . . . | * |
| . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . | . . . |
| 1 | 1 | . . . | 1 | 1 | * | * | . . . | * |

Таблица 4.

Здесь символ «*» заменяет 0 или 1.

Отметим, что некоторые функции из $f_1$, $f_2$, . . . , $f_s$ являются частичными, так как длина слов, хранимых в базе данных, может быть различна.

Предварительная обработка базы данных аналогична алгоритму $DP_{r,f}$. Разбиваем исходное пространство на два подпространства:

$\{0,1\}^n = \{0,1\}^{4k} \times \{0,1\}^{n-4k}$, где $k = \lceil 3\log r + \log n \rceil$. Первое подпространство рассмотрим как четырехмерное линейное пространство $GF(q)^4$, где $q = 2^k$.

Для всех $A \in GF(q)^4$ рассмотрим булевы функции $f_A^{(j)} : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $f_A^{(j)}(B) = f_j(A,B)$, $j = 1,2,...,s$.

Для всех $l \in SL_4(U)$ рассмотрим булевы функции $g_l^{(j)} : \{0,1\}^{n-4k} \to \{0,1\}$ такие, что $g_l^{(j)}(B) = \bigoplus\limits_{A \in l} f_A^{(j)}$, $j = 1,2,...,s$.

Таким образом, таблица 4 разбивается на таблицы для функций $f_A^{(j)}$ и $g_l^{(j)}$. Если за одно обращение к базе данных каждый процессор может считывать не менее $s$ бит информации, то все таблицы функций $f_A^{(j)}$, $j = 1,2,...,s$ для конкретного $A \in GF(q)^4$ объединяются в один блок и представляют собой отдельную часть базы данных, обслуживаемую одним процессором. Для каждого конкретного $l \in SL_4(U)$ мы также получаем отдельную часть базы данных, состоящую из таблиц функций $g_l^{(j)}$, $j = 1,2,...,s$, объединенных в один блок. Для наглядности можно объединить все эти таблицы в две больших таблицы из блоков функций.

| $y_1$ | $y_2$ | ... | $y_{n-4k}$ | $f_{A_1}^{(1)} \ldots f_{A_1}^{(s)}$ | ... | $f_{A_m}^{(1)} \ldots f_{A_m}^{(s)}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | ... | 0 | * ... * | ... | * ... * |
| 0 | 0 | ... | 1 | * ... * | ... | * ... * |
| ... | ... | ... | ... | ... | ... | ... |
| 1 | 1 | ... | 1 | * ... * | ... | * ... * |

Таблица 5.

В данных таблицах $m = |GF(q)^4| = 2^{4k}$, а $p = |SL_4(U)|$.

Возможны два варианта обращения к базе данных: первый – за одно обращение каждый процессор считывает $s$ бит информации и второй – за одно обращение считывается только один бит. В первом случае каждому блоку функций $f_{A_i}^{(1)} \ldots f_{A_i}^{(s)}$, $i = 1,2,...,m$ и каждому блоку $g_{l_i}^{(1)} \ldots g_{l_i}^{(s)}$, $i = 1,2,...,p$ должен соответствовать свой процес-

| $y_1$ | $y_2$ | ... | $y_{n-4k}$ | $g_{l_1}^{(1)} \dots g_{l_1}^{(s)}$ | ... | $g_{l_p}^{(1)} \dots g_{l_p}^{(s)}$ |
|---|---|---|---|---|---|---|
| 0 | 0 | ... | 0 | * ... * | ... | * ... * |
| 0 | 0 | ... | 1 | * ... * | ... | * ... * |
| ... | ... | ... | ... | ... | ... | ... |
| 1 | 1 | ... | 1 | * ... * | ... | * ... * |

Таблица 6.

сор. Если же за одно обращение к базе данных каждый процессор может считывать только один бит информации, то каждой булевой функции в последних двух таблицах должен соответствовать свой процессор.

Отметим, что для обоих вариантов алгоритмы практически не отличаются друг от друга. Поэтому будет предложен только один алгоритм, который мы назовем $DP_{r,f,s}$.

**Алгоритм $DP_{r,f,s}$:**

**1.** Применим процедуру $\mathcal{A}$ к входному вектору $\vec{A}$, то есть $\vec{u} = \mathcal{A}(\vec{A})$.

**2.** Применим процедуру $\mathcal{B}_5$, для построения правильных входов булевых функций базы данных, то есть $\mathcal{B}_5$:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;

ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

**3.** Вычислим все функции базы данных:

i) $\forall A \in GF(q)^4$ и $\forall j = 1, 2, ..., s$ вычислим $Y_A^{(j)} = f_A^{(j)}(B_A)$;

ii) $\forall l \in SL_4(U)$ и $\forall j = 1, 2, ..., s$ вычислим $Z_l^{(j)} = g_l^{(j)}(B_l)$.

Заметим, что это единственный шаг на котором происходит обращение к базе данных и все время вычислений можно отнести к **extime**, так как все функции должны считаться за одно обращение к базе данных независимо от того, сколько бит считывает каждый процессор за один раз. Разумеется это будет выполнено при определенном количестве процессоров в каждом случае. Оба возможных варианта были описаны выше.

Обозначим через $\vec{Y}^{(j)}$ набор переменных $Y_A^{(j)}$, где $A \in GF(q)^4$, а через $\vec{Z}^{(j)}$ набор переменных $Z_l^{(j)}$, где $l \in SL_4(U)$.

**4.** Применим процедуру $\mathcal{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathcal{B}_7$:

$\forall i = 1, 2, ..., r$ и $\forall j = 1, 2, ..., s$ вычислим

$$f_j(A_i, B_i) = \mathcal{B}_6(A_i, u_i, \vec{Z}^{(j)}, \vec{Y}^{(j)}) =$$
$$= \left( \bigvee_{l \in SL_4(U)} Z_l^{(j)} \cdot \mathcal{B}_2(A, u, l) \right) \oplus \left( \bigoplus_{A^* \in GF(q)^4} Y_{A^*}^{(j)} \cdot \mathcal{B}_1(A, u, A^*) \right).$$

Оценим сложность полученного алгоритма. Видно, что первые два шага ничем не отличаются от шагов алгоритма $DP_{r,f}$ [2] мы сможем использовать соответствующие результаты из [1, 2].

**Теорема 2.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f_1$, $f_2$, . . . , $f_s$ от $n$ переменных выполнено:*
$\mathbf{ptime}(DP_{r,f,s}) = O(\log r + \log n) + \mathbf{extime}$,
$\mathbf{bp}(DP_{r,f,s}) = O((sr^{13}n^5)(\log r + \log n)^2)$,
$\mathbf{mem}(DP_{r,f,s}) \leqslant s \cdot 2^n \left(1 + \frac{1}{n}\right)$.

**Доказательство.** Первый шаг алгоритма можно оценить, используя результаты из второго раздела
$\mathbf{bp}(\mathcal{A}) = O(r^7k^2)$,
$\mathbf{ptime}(\mathcal{A}) = O(\log r + \log k)$.
Для второго шага мы имеем
$\mathbf{bp}(\mathcal{B}_5) = O(|GF(q)^4| \cdot \mathbf{bp}(\mathcal{B}_3)) + O(|SL_4(U)| \cdot \mathbf{bp}(\mathcal{B}_4))$,
$\mathbf{ptime}(\mathcal{B}_5) = \mathbf{ptime}(\mathcal{B}_3) = \mathbf{ptime}(\mathcal{B}_4)$.
Воспользуемся далее оценками соответствующих процедур из [1] и [2], учитывая, что $|GF(q)^4| = 2^{4k}$ и $|SL_4(U)| \leqslant 2^{4k} \cdot \frac{1}{n}$.
Получаем
$\mathbf{bp}(\mathcal{B}_5) = O(r2^{4k} \cdot (k^2 + n)) + O(r2^{4k} \cdot \left(1 + \frac{k}{n}\right)) = O(r2^{4k} \cdot (k^2 + n))$ и
$\mathbf{ptime}(\mathcal{B}_5) = O(\log k + \log r)$.

На третьем шаге происходит обращение к базе данных, время которого можно оценить через величину **extime**. Количество процессоров необходимое на этом шаге алгоритма зависит от того, сколько бит считывает каждый процессор при одном обращении к базе данных. У нас $|GF(q)^4|$ блоков функций $f_A^{(j)}$ и $|SL_4(U)|$ блоков функций $g_l^{(j)}$, которые содержат по $s$ функций. Следовательно, если каждый процессор может считывать $s$ бит за один раз, то

нам необходимо $|GF(q)^4| + |SL_4(U)|$ процессоров. Если же за одно обращение процессора считывается только один бит, то это количество необходимо увеличить в $s$ раз и, следовательно, потребуется $s \cdot (|GF(q)^4| + |SL_4(U)|)$ процессоров. Таким образом, на третьем шаге алгоритма необходимо $O(2^{4k})$ или $O(s \cdot 2^{4k})$ процессоров.

Легко видеть, что четвертый шаг алгоритма может быть оценен следующим образом:

$\mathbf{bp}(\mathcal{B}_7) = r \cdot s \cdot \mathbf{bp}(\mathcal{B}_6), \quad \mathbf{ptime}(\mathcal{B}_7) = \mathbf{ptime}(\mathcal{B}_6).$

Воспользуемся оценками для процедуры $\mathcal{B}_6$ из [2].

$\mathbf{bp}(\mathcal{B}_6) = O(k2^{4k} \cdot \left(k + \frac{1}{n}\right)), \quad \mathbf{ptime}(\mathcal{B}_6) = O(k).$

Следовательно,

$\mathbf{bp}(\mathcal{B}_7) = O(r \cdot s \cdot k \cdot 2^{4k} \cdot \left(k + \frac{1}{n}\right)) = O(r \cdot s \cdot k^2 \cdot 2^{4k}), \mathbf{ptime}(\mathcal{B}_7) = O(k).$

Как было определено ранее $k = \lceil 3 \log r + \log n \rceil$. Подставляя эту величину в ранее полученные оценки, имеем:

$\mathbf{bp}(\mathcal{B}_5) \leqslant O(r2^{4k}(3 \log r + \log n + 1) \cdot (n + (3 \log r + \log n + 1)^2)) = O(r^{13}n^4(n + (\log r + \log n)^2)) = O(r^{13}n^5) + O(r^{13}n^4(\log r + \log n)^2).$

На третьем шаге нам требуется $O(r^{12}n^4)$ или $O(s \cdot r^{12}n^4)$ процессоров.

Оценки четвертого шага примут следующий вид

$\mathbf{bp}(\mathcal{B}_7) = O(r \cdot s \cdot k^2 \cdot 2^{4k}) \leqslant O(r \cdot s \cdot (3 \log r + \log n + 1)^2 \cdot 2^{4(3 \log r + \log n + 1)}) = O(s \cdot r^{13}n^4(\log r + \log n)^2).$

Заметим, что четвертый шаг алгоритма требует больше процессоров, чем в любом случае будет необходимо на третьем шаге и, следовательно, для оценок алгоритма не важно, сколько бит считывается каждым процессором за одно обращение к базе данных.

Окончательно имеем следующее:

$\mathbf{bp}(DP_{r,f,s}) = \max(\mathbf{bp}(\mathcal{A}), \mathbf{bp}(\mathcal{B}_5), \mathbf{bp}(\mathcal{B}_7)) = \max(O(r^7k^2), O(r^{13}n^5) + O(r^{13}n^4(\log r + \log n)^2), O(sr^{13}n^4(\log r + \log n)^2)) \leqslant O(sr^{13}n^5(\log r + \log n)^2),$

$\mathbf{ptime}(DP_{r,f,s}) = \mathbf{ptime}(\mathcal{A}) + \mathbf{ptime}(\mathcal{B}_5) + \mathbf{extime} + \mathbf{ptime}(\mathcal{B}_7) = O(\log r + \log k) + O(k) + \mathbf{extime} = O(\log r + \log(\log r + \log n)) + O(\log r + \log n) + \mathbf{extime} = O(\log r + \log n) + \mathbf{extime}.$

Оценка размера базы данных, полученной при построении алгоритма легко получается из аналогичной оценки для алгоритма $DP_{r,f}$. Таблицы 5 и 6 вместе содержат $s \cdot (|GF(q)^4| + |SL_4(U)|)$ столбцов по $2^{n-4k}$ бит в каждом. Следовательно,

$\mathbf{mem}(DP_{r,f,s}) = s \cdot 2^{n-4k}(|GF(q)^4| + |SL_4(U)|) \leqslant s \cdot 2^{n-4k}(2^{4k} + 2^{4k} \cdot \frac{1}{n}) = s \cdot 2^n \cdot \left(1 + \frac{1}{n}\right)$.

Таким образом, мы получили все оценки для алгоритма $DP_{r,f,s}$ и теорема доказана полностью.

## 5. Параллельный алгоритм доступа к криптографически защищенной базе данных из $s$ булевых функций и его сложность

Рассмотрим возможность криптографической защиты базы данных из $s$ булевых функций при использовании алгоритма $DP_{r,f,s}$ из предыдущего раздела для доступа к информации. Отметим, что для работы алгоритма $DP_{r,f,s}$ необходимо специальным образом преобразовать базу данных. Результат такого преобразования был представлен в таблицах 5 и 6.

В зависимости от того, сколько бит может считывать процессор за одно обращение к базе данных, каждый блок булевых функций либо каждая булева функция из указанных таблиц хранится на отдельном носителе и обслуживается своим процессором. Оба возможных варианта были рассмотрены в предыдущем разделе. Там же было доказано. что в обоих случаях оценки сложности алгоритма одинаковые.

Таким образом, перед нами стоит задача криптографической защиты таблиц 5 и 6. Причем главным ограничением является требование обеспечения доступа к необходимым булевым функциям за одно обращение к базе данных. С подобной ситуацией мы уже встречались при разработке алгоритма $CDP_{r,f}$ и здесь воспользуемся аналогичными методами.

Сразу же отметим, что при решении рассматриваемой задачи можно вести речь только о шифровании строк указанных таблиц, так как шифрование столбцов приведет к нарушению главного ограничения алгоритма $DP_{r,f,s}$. Действительно, при работе алгоритма потребуется расшифровать используемые столбцы, а для этого необходимо сначала считать их с внешнего носителя. Так как каждый столбец хранится на отдельном носителе, то полностью считать его

за одно обращение к базе данных невозможно.

Предположим сначала, что $s$ удовлетворяет параметрам криптосистемы, то есть длина шифруемой последовательности должна быть меньше либо равна $s$. В этом случае в базе данных можно хранить зашифрованным каждый блок булевых функций $f_{A_i}^{(1)} \ldots f_{A_i}^{(s)}$ и $g_{l_i}^{(1)} \ldots g_{l_i}^{(s)}$, где $i$= 1, 2, . . . , $p$. Тогда новый алгоритм практически ничем не будет отличается от $DP_{r,f,s}$, за исключением процедуры расшифровки каждого считанного блока, которую необходимо добавить после третьего шага алгоритма $DP_{r,f,s}$. Следовательно, последний (четвертый) шаг алгоритма станет пятым и добавится новый четвертый шаг:

**4.** Применим процедуру $\mathcal{C}$ для расшифровки всех извлеченных блоков:

i) $\forall A \in GF(q)^4$ расшифруем блоки $(Y_A^{(1)}, \ldots, Y_A^{(s)})$;

ii) $\forall l \in SL_4(U)$ расшифруем блоки $(Z_l^{(1)}, \ldots, Z_l^{(s)})$.

Сложность этой процедуры полностью зависит от используемой криптосистемы и возможности ее распараллеливания. Так как в данной работе не используется никакая конкретная криптосистема, то предположим, что нам будет достаточно количества процессоров используемых при работе алгоритма, а время работы этой процедуры обозначим через величину **crypttime**. Ясно, что последняя величина это время необходимое для расшифровки $(|GF(q)^4| + |SL_4(U)|)$ блоков по $s$ бит в каждом при использовании параллельных процессоров.

Легко видеть, что сложность полученного нового алгоритма отличается от сложности алгоритма $DP_{r,f,s}$ только величиной **ptime**, так как к ней добавится **crypttime**.

Рассмотрим теперь ситуацию, когда параметры криптосистемы таковы, что нужно зашифровывать последовательности, состоящие из нескольких блоков таблиц базы данных 5 и 6. То есть каждая строка указанных таблиц разбивается на части, содержащие по несколько блоков булевых функций $f_{A_i}^{(1)} \ldots f_{A_i}^{(s)}$ или $g_{l_i}^{(1)} \ldots g_{l_i}^{(s)}$, где $i = 1, 2, ..., p$. Полученные части шифруются по отдельности и хранятся в зашифрованном виде. Очевидно, что перед применением алгоритма $DP_{r,f,s}$ необходимо расшифровать все части строк таблиц

базы данных, используемые при его работе. Однако, может случиться так, что для разных строк потребуются одинаковые блоки булевых функций, которые были зашифрованы в различных частях и, следовательно, необходимо считать и расшифровать каждую такую часть. В этом случае из одного столбца таблицы базы данных будет считываться несколько бит, то есть произойдет нарушение ограничения алгоритма $DP_{r,f,s}$. Рассмотрим описанную ситуацию на простом примере.

Приведем фрагмент одной из таблиц базы данных (таблицы 5).

| $y_1$ | $y_2$ | $\ldots$ | $y_{n-4k}$ | $f_{A_1}^{(1)} \ldots f_{A_1}^{(s)}$ | $f_{A_2}^{(1)} \ldots f_{A_2}^{(s)}$ | $\ldots$ |
|---|---|---|---|---|---|---|
| 0 | 0 | $\ldots$ | 0 | $* \ldots *$ | $* \ldots *$ | $\ldots$ |
| 0 | 0 | $\ldots$ | 1 | $* \ldots *$ | $* \ldots *$ | $\ldots$ |
| $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ | $\ldots$ |
| 1 | 1 | $\ldots$ | 1 | $* \ldots *$ | $* \ldots *$ | $\ldots$ |

Таблица 7.

Предположим, что зашифровывались последовательности, содержащие по два блока булевых функций, то есть длина шифруемой последовательности $2s$ бит. Пусть при работе алгоритма выяснилось, что из первой строки нужно считать блок функций $f_{A_1}^{(1)} \ldots f_{A_1}^{(s)}$, а из второй строки блок $f_{A_2}^{(1)} \ldots f_{A_2}^{(s)}$. Однако в каждой строке нашей таблицы оба эти блока зашифрованы вместе и чтобы извлечь блок функций $f_{A_1}^{(1)} \ldots f_{A_1}^{(s)}$ для первой строки необходимо считать из этой строки оба блока вместе, а затем расшифровать их. Аналогично, для второй строки требуется считать одновременно оба блока для их расшифрования. Таким образом, при одном обращении к базе данных необходимо одновременно считать блоки $f_{A_1}^{(1)} \ldots f_{A_1}^{(s)}$ и $f_{A_2}^{(1)} \ldots f_{A_2}^{(s)}$ из первой и второй строки таблицы. Очевидно, что обычными методами это сделать невозможно. Следовательно, мы имеем ситуацию подобную той, которая была решена при построении алгоритма $CDP_{r,f}$ и мы можем воспользоваться теми же методами.

Во втором разделе было отмечено, что при своей работе алгоритм $DP_{r,f}$ обращается только к строкам, соответствующим набо-

рам $B_1, \ldots, B_r$. Это же утверждение выполнено и для алгоритма $DP_{r,f,s}$. Поэтому одно из возможных решений проблемы состоит в расшифровке всех соответствующих $r$ строк таблиц базы данных с применением методов алгоритма $CDP_{r,f}$ для считывания $r$ бит из каждого столбца.

Заметим, однако, что не требуется считывать и расшифровывать $r$ строк таблиц целиком. Действительно, для каждого конкретного $A \in GF(q)^4$ существует единственный набор $B_A$, определяемый в алгоритме $DP_{r,f,s}$, а для каждого конкретного $l \in SL_4(U)$ единственный набор $B_l$. Следовательно, из каждого столбца таблиц 5 и 6 считывается только один бит. В этом, собственно, и состоит главное ограничения алгоритма. Пусть параметры криптосистемы таковы, что должны шифроваться последовательности длиной $bs$ бит, где $b$ – некоторое натуральное число, то есть шифруются каждые $b$ блоков булевых функций. При работе алгоритма нужно вычислить каждый блок булевых функций на определенном наборе, то есть считать его с внешнего носителя по определенному адресу. Но при этом необходимо считать и всю зашифрованную последовательность, содержащую этот блок, а это означает дополнительное вычисление $b - 1$ блоков булевых функций на этом наборе. Так как любая зашифрованная последовательность содержит по $b$ блоков, то в худшем случае потребуется вычислить каждый блок булевых функций на $b$ независимых адресах. Следовательно, необходимо будет считать максимум по $b$ значений из каждого столбца таблиц базы данных 5 и 6.

Таким образом, нам необходимо модифицировать алгоритм $DP_{r,f,s}$, по аналогии с алгоритмом $CDP_{r,f}$ для извлечения $b$ значений из каждого столбца таблиц базы данных за одно к ней обращение. Новый алгоритм обозначим через $CDP_{r,f,s}$.

Воспользуемся изложенным при построении алгоритма $CDP_{r,f}$ методом разбиения каждого столбца таблиц базы данных и разобьем их так, чтобы стало возможным применение алгоритма $DP_{b,f}$. Опишем алгоритм $CDP_{r,f,s}$ в случае, когда шифруются последовательности из $bs$ булевых функций ($b$ блоков по $s$ функций в таблицах базы данных). При этом будем считать, что за одно обращение к базе данных каждый процессор может считать только один бит. (Случай, когда можно считывать по $s$ бит аналогичен.)

**Алгоритм** $CDP_{r,f,s}$**:**

**1.** Применим процедуру $\mathcal{A}$ к входному вектору $\vec{A}$, то есть $\vec{u} = \mathcal{A}(\vec{A})$.

**2.** Применим процедуру $\mathcal{B}_5$, для построения правильных входов булевых функций базы данных, то есть $\mathcal{B}_5$:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;

ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

**3.** Применим процедуру $\mathcal{D}$ для вычисления всех необходимых внешних функций. Для этого с помощью алгоритма $DP_{b,f}$ вычислим по $b$ соответствующих значений каждой булевой функции $f_{A_i}^{(j)}$ и $g_{l_i}^{(j)}$. Для того, чтобы сформировать правильные зашифрованные последовательности будем отдельно обрабатывать по $b$ блоков булевых функций. Количество последовательностей из $b$ блоков булевых функций $f_{A_i}^{(1)} \ldots f_{A_i}^{(s)}$, где $i = 1, 2, ..., m$ можно оценить как

$$z = \left\lceil \frac{m}{b} \right\rceil = \left\lceil \frac{|GF(q)^4|}{b} \right\rceil,$$

а количество последовательностей функций $g_{l_i}^{(1)} \ldots g_{l_i}^{(s)}$, где $i = 1, 2, ..., p$ равно

$$v = \left\lceil \frac{p}{b} \right\rceil = \left\lceil \frac{|SL_4(U)|}{b} \right\rceil.$$

Таким образом, получаем следующую процедуру:

i) $\forall i = 1, 2, ..., b$ и $\forall j = 1, 2, ..., s$ применим алгоритм $DP(b, f)_{A_i}^{(j)}$ ко входу $(B_{A_1}, B_{A_2}, \ldots, B_{A_b})$ и из результатов сформируем таблицу $F_1$:

| | | | |
|---|---|---|---|
| $f_{A_1}^{(1)}(B_{A_1}) \ldots f_{A_1}^{(s)}(B_{A_1})$ | $f_{A_2}^{(1)}(B_{A_1}) \ldots f_{A_2}^{(s)}(B_{A_1})$ | ... | $f_{A_b}^{(1)}(B_{A_1}) \ldots f_{A_b}^{(s)}(B_{A_1})$ |
| $f_{A_1}^{(1)}(B_{A_2}) \ldots f_{A_1}^{(s)}(B_{A_2})$ | $f_{A_2}^{(1)}(B_{A_2}) \ldots f_{A_2}^{(s)}(B_{A_2})$ | ... | $f_{A_b}^{(1)}(B_{A_2}) \ldots f_{A_b}^{(s)}(B_{A_2})$ |
| ... | ... | ... | ... |
| $f_{A_1}^{(1)}(B_{A_b}) \ldots f_{A_1}^{(s)}(B_{A_b})$ | $f_{A_2}^{(1)}(B_{A_b}) \ldots f_{A_2}^{(s)}(B_{A_b})$ | ... | $f_{A_b}^{(1)}(B_{A_b}) \ldots f_{A_b}^{(s)}(B_{A_b})$ |

Таблица 8.

$\forall i = b+1, b+2, ..., 2b$ и $\forall j = 1, 2, ..., s$ применим алгоритм

$DP(b,f)^{(j)}_{A_i}$ ко входу $(B_{A_b+1},\, B_{A_b+2}, ..., B_{A_{2b}})$ и из результатов сформируем таблицу $F_2$:

| | | |
|---|---|---|
| $f^{(1)}_{A_{b+1}}(B_{A_{b+1}})$ ... $f^{(s)}_{A_{b+1}}(B_{A_{b+1}})$ | ... | $f^{(1)}_{A_{2b}}(B_{A_{b+1}})$ ... $f^{(s)}_{A_{2b}}(B_{A_{b+1}})$ |
| $f^{(1)}_{A_{b+1}}(B_{A_{b+2}})$ ... $f^{(s)}_{A_{b+1}}(B_{A_{b+2}})$ | ... | $f^{(1)}_{A_{2b}}(B_{A_{b+2}})$ ... $f^{(s)}_{A_{2b}}(B_{A_{b+2}})$ |
| ... | ... | ... |
| $f^{(1)}_{A_{b+1}}(B_{A_{2b}})$ ... $f^{(s)}_{A_{b+1}}(B_{A_{2b}})$ | ... | $f^{(1)}_{A_{2b}}(B_{A_{2b}})$ ... $f^{(s)}_{A_{2b}}(B_{A_{2b}})$ |

Таблица 9.

Продолжаем далее аналогично.

. . . . . .

$\forall i = (z-1)b+1, (z-1)b+2, ..., m$ и $\forall j = 1, 2, ..., s$ применим алгоритм $DP(b,f)^{(j)}_{A_i}$ ко входу $(B_{A_{(z-1)b+1}},\, B_{A_{(z-1)b+2}},\, \ldots,\, B_{A_m})$ и из результатов сформируем таблицу $F_z$ по аналогии с таблицами, описанными выше.

ii) $\forall i = 1, 2, ..., b$ и $\forall j = 1, 2, ..., s$ применим алгоритм $DP(b,g)^{(j)}_{l_i}$ ко входу $(B_{l_1},\, B_{l_2},\, \ldots,\, B_{l_b})$ и из результатов сформируем таблицу $G_1$ по аналогии с таблицей $F_1$:

| | | | |
|---|---|---|---|
| $g^{(1)}_{l_1}(B_{l_1})$ ... $g^{(s)}_{l_1}(B_{l_1})$ | $g^{(1)}_{l_2}(B_{l_1})$ ... $g^{(s)}_{l_2}(B_{l_1})$ | ... | $g^{(1)}_{l_b}(B_{l_1})$ ... $g^{(s)}_{l_b}(B_{l_1})$ |
| $g^{(1)}_{l_1}(B_{l_2})$ ... $g^{(s)}_{l_1}(B_{l_2})$ | $g^{(1)}_{l_2}(B_{l_2})$ ... $g^{(s)}_{l_2}(B_{l_2})$ | ... | $g^{(1)}_{l_b}(B_{l_2})$ ... $g^{(s)}_{l_b}(B_{l_2})$ |
| ... | ... | ... | ... |
| $g^{(1)}_{l_1}(B_{l_b})$ ... $g^{(s)}_{l_1}(B_{l_b})$ | $g^{(1)}_{l_2}(B_{l_b})$ ... $g^{(s)}_{l_2}(B_{l_b})$ | ... | $g^{(1)}_{l_b}(B_{l_b})$ ... $g^{(s)}_{l_b}(B_{l_b})$ |

Таблица 10.

$\forall i = b+1, b+2, ..., 2b$ и $\forall j = 1, 2, ..., s$ применим алгоритм $DP(b,g)^{(j)}_{l_i}$ ко входу $(B_{l_{b+1}},\, B_{l_{b+2}},\, \ldots,\, B_{l_{2b}})$ и из результатов сформируем таблицу $G_2$ по аналогии с таблицей $F_2$.

Продолжаем далее таким же образом.

. . . . . .

$\forall i = (v-1)b+1, (v-1)b+2, ..., p$ и $\forall j = 1, 2, ..., s$ применим алгоритм $DP(b,g)^{(j)}_{l_i}$ ко входу $(B_{l_{(v-1)b+1}}, B_{l_{(v-1)b+2}}, \ldots, B_{lp})$ и из результатов сформируем таблицу $G_v$.

Это единственный шаг алгоритма, на котором происходит обращение к базе данных. Отметим, что все таблицы $F_i$ и $G_i$ вместе занимают $bs(|GF(q)^4| + |SL_4(U)|)$ бит и должны храниться в оперативной памяти.

**4.** Применим процедуру $\mathcal{C}$ для расшифровки всех строк каждой таблицы $F_i$, где $i = 1, 2, ..., z$ и каждой таблицы $G_j$, где $j = 1, 2, ..., v$. Расшифрованные таблицы по прежнему должны храниться в оперативной памяти.

**5.** Выберем необходимые нам значения функций в расшифрованных таблицах $F$ и $G$:

i) $\forall A \in GF(q)^4$ и $\forall j = 1, 2, ..., s$ выберем $Y^{(j)}_A = f^{(j)}_A(B_A)$;

ii) $\forall l \in SL_4(U)$ и $\forall j = 1, 2, ..., s$ выберем $Z^{(j)}_l = g^{(j)}_l(B_l)$.

Обозначим через $\vec{Y}^{(j)}$ набор переменных $Y^{(j)}_A$, где $A \in GF(q)^4$, а через $\vec{Z}^{(j)}$ набор переменных $Z^{(j)}_l$, где $l \in SL_4(U)$.

**6.** Применим процедуру $\mathcal{B}_7$ для вычисления выходной последовательности алгоритма, то есть $\mathcal{B}_7$:

$\forall i = 1, 2, ..., r$ и $\forall j = 1, 2, ..., s$ вычислим

$$f_j(A_i, B_i) = \mathcal{B}_6(A_i, u_i, \vec{Z}^{(j)}, \vec{Y}^{(j)}) =$$
$$= \left( \bigvee_{l \in SL_4(U)} Z^{(j)}_l \cdot \mathcal{B}_2(A, u, l) \right) \oplus \left( \bigoplus_{A^* \in GF(q)^4} Y^{(j)}_{A^*} \cdot \mathcal{B}_1(A, u, A^*) \right).$$

Оценим сложность полученного алгоритма по трем параметрам, опираясь на доказательство оценок для алгоритмов $DP_{r,f}$ и $DP_{r,f,s}$.

**Теорема 3.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f_1$, $f_2$, . . . , $f_s$ от $n$ переменных выполнено:*

$\mathbf{bp}(CDP_{r,f,s}) = O((sb^{13}r^{13}n^4)(n - 12\log r - 4\log n)^5(\log b + \log(n - 12\log r - 4\log n))^2)$,

$\mathbf{ptime}(CDP_{r,f,s}) = O(\log r + \log n) + \mathbf{extime} + \mathbf{crypttime}$,

$\mathbf{mem}(CDP_{r,f,s}) \leqslant s2^n \left(1 + \frac{1}{n}\right) \left(1 + \frac{1}{n - 12\log r - 4\log n}\right)$.

**Доказательство.** Первый шаг алгоритма оценим, используя ранее описанные результаты $\mathbf{bp}(\mathcal{A}) = O(r^7k^2)$, $\mathbf{ptime}(\mathcal{A}) = O(\log r + \log k)$.

Для второго шага, согласно доказательству оценок для алгоритма $DP_{r,f,s}$ (Теорема 2) мы имеем $\mathbf{bp}(\mathcal{B}_5) = O(r^{13}n^5) + O(r^{13}n^4(\log r + \log n)^2)$ и $\mathbf{ptime}(\mathcal{B}_5) = O(\log k + \log r)$.

Рассмотрим третий шаг алгоритма. На этом шаге из базы данных за одно к ней обращение считывается по $b$ соответствующих значений каждой булевой функции $f_{A_i}^{(j)}$ и $g_{l_i}^{(j)}$. Для этого используется алгоритм $DP_{b,f}$. Указанных булевых функций у нас всего $s \cdot (|GF(q)^4| + |SL_4(U)|)$.

Следовательно, получаем:

$\mathbf{bp}(\mathcal{D}) = s \cdot (|GF(q)^4| + |SL_4(U)|) \cdot \mathbf{bp}(DP_{b,f_A})$.

Воспользуемся оценками сложности для алгоритма $DP_{r,f}$ из [1, 2]. Учтем, что в нашем случае $r = b$, а $f = f_A$. Причем функция $f_A$ – это булева функция $n - 4k$ переменных. Имеем

$\mathbf{bp}(DP_{b,f_A}) = O(b^{13}(n-4k)^5(\log b + \log(n-4k))^2)$.

Таким образом, зная, что $k = \lceil 3\log r + \log n \rceil$, $|GF(q)^4| = 2^{4k}$ и $|SL_4(U)| \leqslant 2^{4k} \cdot \frac{1}{n}$, получаем оценку количества процессоров на четвертом шаге

$\mathbf{bp}(\mathcal{D}) = O(s \cdot 2^{4k} \cdot \left(1 + \frac{1}{n}\right) \cdot b^{13}(n-4k)^5(\log b + \log(n-4k))^2) = O(sr^{12}n^4b^{13}(n - 12\log r - 4\log n)^5(\log b + \log(n - 12\log r - 4\log n))^2)$.

Время вычислений оценивается следующим образом

$\mathbf{ptime}(\mathcal{D}) = \mathbf{ptime}(DP_{b,f_A}) = O(\log b + \log(n - 12\log r - 4\log n)) +$ **extime**, где величина **extime** характеризует время считывания информации с внешних носителей.

На четвертом шаге происходит расшифрование строк таблиц. Будем считать, что нам не требуется для этого новых процессоров, так как мы не используем никакой конкретной криптосистемы и не знаем оценок ее сложности. Заметим, что на этом шаге необходимо расшифровать

$$b \cdot (z + v) = b \cdot \left( \left\lceil \frac{|GF(q)^4|}{b} \right\rceil + \left\lceil \frac{|SL_4(U)|}{b} \right\rceil \right) \leqslant 16r^{12}n^4\left(1 + \frac{1}{n}\right) + 2b$$

последовательностей по $bs$ бит в каждой. Оценим время этой процедуры через **crypttime**.

Пятый шаг не требует новых процессоров, а время его работы можно оценить как O(1).

Оценки для шестого шага полностью соответствуют соответствующим оценкам для алгоритма $DP_{r,f,s}$. Следовательно,

$\mathbf{bp}(\mathcal{B}_7) = O(r \cdot s \cdot k^2 \cdot 2^{4k}) = O(s \cdot r^{13} n^4 (3\log r + \log n)^2)$,

$\mathbf{ptime}(\mathcal{B}_7) = O(k) = O(\log r + \log n)$.

Окончательно имеем следующее:

$\mathbf{bp}(CDP_{r,f,s}) = \max(\mathbf{bp}(\mathcal{A}), \mathbf{bp}(\mathcal{B}_5), \mathbf{bp}(\mathcal{D}), \mathbf{bp}(\mathcal{B}_7)) = \max(O(r^7 k^2), O(r^{13} n^5) + O(r^{13} n^4 (\log r + \log n)^2), O(sr^{12} n^4 b^{13} (n - 12\log r - 4\log n)^5 (\log b + \log(n - 12\log r - 4\log n))^2), O(sr^{13} n^4 (\log r + \log n)^2)) \leqslant O(sr^{13} n^4 b^{13} (n - 12\log r - 4\log n)^5 (\log b + \log(n - 12\log r - 4\log n))^2)$;

$\mathbf{ptime}(CDP_{r,f,s}) = \mathbf{ptime}(\mathcal{A}) + \mathbf{ptime}(\mathcal{B}_5) + \mathbf{ptime}(\mathcal{D}) + \mathbf{extime} + \mathbf{crypttime} + O(1) + \mathbf{ptime}(\mathcal{B}_7) = O(\log r + \log(\log r + \log n)) + O(\log r + \log n) + O(\log b + \log(n - 12\log r - 4\log n)) + \mathbf{extime} + \mathbf{crypttime} + O(1) = O(\log r + \log n) + O(\log b + \log n) + \mathbf{extime} + \mathbf{crypttime}$.

Заметим, что теоретически (на практике это нереально) максимальное значение $b$ равняется $|GF(q)^4| + |SL_4(U)| \leqslant 2^{4k} \cdot \left(1 + \frac{1}{n}\right) \leqslant 16 r^{12} n^4 \left(1 + \frac{1}{n}\right)$, и, следовательно, $\log b = O(\log r + \log n)$.

Таким образом,

$\mathbf{ptime}(CDP_{r,f,s}) = O(\log r + \log n) + \mathbf{extime} + \mathbf{crypttime}$.

Оценим размер базы данных, полученной при построении алгоритма. Для алгоритма $DP_{r,f,s}$ количество столбцов таблиц 5 и 6 определяется как

$s \cdot (|GF(q)^4| + |SL_4(U)|)$.

Каждый столбец должен быть разбит на части для применения алгоритма $DP_{b,f}$, где в качестве функции $f$ выступают булевы функции $f_{A_i}^{(j)}$ и $g_{l_i}^{(j)}$, определенные на $n - 4k$ переменных. Согласно оценкам для алгоритма $DP_{r,f}$, получаем

$\mathbf{mem}(DP_{b,f}) \leqslant 2^{n-4k} \cdot \left(1 + \frac{1}{n-4k}\right)$.

Следовательно,

$\mathbf{mem}(CDP_{r,f,s}) = s \cdot (|GF(q)^4| + |SL_4(U)|) \cdot \mathbf{bp}(DP_{b,f}) \leqslant s \cdot (2^{4k} + 2^{4k} \cdot \frac{1}{n}) \cdot 2^{n-4k} \cdot \left(1 + \frac{1}{n-4k}\right) \leqslant s \cdot 2^n \cdot \left(1 + \frac{1}{n}\right) \cdot \left(1 + \frac{1}{n - 12\log r - 4\log n}\right)$.

Таким образом, мы получили все оценки для алгоритма $DP_{r,f,s}$ и теорема доказана полностью.

Заметим, однако, что если $b > r$, то выгоднее считывать и расшифровывать все $r$ строк таблиц базы данных, соответствующих наборам $B_1, \ldots, B_r$. Действительно, в этом случае у каждого столбца

требуется считать $r$ бит, но так как $b > r$, то это выгоднее, чем считывать $b$ бит. Тогда на третьем шаге $DP_{r,f,s}$ необходимо применять алгоритм $DP_{r,f}$. Следовательно,

$\mathbf{bp}(\mathcal{D}) = s \cdot (|GF(q)^4| + |SL_4(U)|) \cdot \mathbf{bp}(DP_{r,f}) = s \cdot 2^{4k} \cdot \left(1 + \frac{1}{n}\right) \cdot O(r^{13}(n - 4k)^5(\log r + \log(n - 4k))^2) \leqslant O(sr^{12}n^4r^{13}(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2) = O(sr^{25}n^4(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2)$.

Далее, следуя доказательству предыдущей теоремы, нетрудно показать, что

$\mathbf{bp}(CDP_{r,f,s}) = O(sr^{25}n^4(n - 12\log r - 4\log n)^5(\log r + \log(n - 12\log r - 4\log n))^2)$.

Остальные оценки алгоритма останутся без изменений.

## 6. Сложность алгоритма $DP_{r,f}$ при ограниченном числе процессоров

Отметим, что все задачи, рассмотренные в предыдущих разделах данной работы, решались при условии максимального распараллеливания алгоритмов, что существенно увеличивало количество процессоров в оценках их сложности. Минимизируем количество процессоров в алгоритме $DP_{r,f}$ при соблюдении главного ограничения задачи, а именно однократного обращения к базе данных. Ясно, что нет необходимости в модификации алгоритма. Изменятся только его оценки сложности.

Рассмотрим задачу параллельного извлечения информации из базы данных по $r$ независимым адресам за одно обращение к внешним носителям при минимально возможном количестве процессоров.

Алгоритм $DP_{r,f}$ разбивает базу данных на определенное количество частей, каждая из которых обслуживается своим процессором. При обращении к базе данным по любым $r$ адресам допускается только одно обращение каждого процессора к соответствующей ему части и вся информация должна быть считана за один шаг. Поэтому для выполнения главного ограничения алгоритма необходимо минимум столько процессоров, на сколько частей разбита база данных. Заметим, что для работы алгоритма $DP_{r,f}$ необходимо

представить булеву функцию $f$, представляющую собой базу данных, через булевы функции $f_A^{(j)}$ и $g_l^{(j)} : \{0,1\}^{n-4k} \to \{0,1\}$, где $A \in GF(q)^4$, $l \in SL_4(U)$. Каждая такая функция это часть разбиения базы данных, обслуживаемая отдельным процессором. Следовательно, количество частей в $DP_{r,f}$ не может превышать величину $|GF(q)^4| + |SL_4(U)| \leqslant (2^{4k} + 2^{4k} \cdot \frac{1}{n}) = 2^{4k} \cdot \left(1 + \frac{1}{n}\right) \leqslant 16r^{12}n^4\left(1 + \frac{1}{n}\right)$.

Таким образом, количество процессоров алгоритма можно ограничить величиной $O(r^{12}n^4)$.

Оценим время работы алгоритма при таком ограниченном количестве процессоров. Для этого нам потребуется рассмотреть все шаги алгоритма $DP_{r,f}$, описанного во втором разделе данной работы.

**Теорема 4.** *Для любых положительных целых $n$ и $r$, $r \leqslant 2^{\frac{n}{16}}$ и для любых булевых функций $f$ от $n$ переменных время работы алгоритма $DP_{r,f}$ при количестве процессоров, ограниченном величиной $O(r^{12}n^4)$ можно оценить как:*

$$\mathbf{ptime}(DP_{r,f}) = O(r(n + \log^2 r + \log^2 n + \log r \log n)(\log r + \log n)) + \mathbf{extime}.$$

**Доказательство.** Рассмотрим первый шаг алгоритма $DP_{r,f}$. Его сложность оценивалась следующим образом.

$\mathbf{bp}(\mathcal{A}) = O(r^7(\log r + \log n)^2)$,

$\mathbf{ptime}(\mathcal{A}) = O(\log r + \log(\log r + \log n))$.

Количество процессоров, необходимое на данном шаге, удовлетворяет нашему ограничению и, следовательно, оценка времени не изменится.

Для второго шага алгоритма $DP_{r,f}$ без ограничения количества процессоров мы имели

$\mathbf{bp}(\mathcal{B}_5) \leqslant O(r^{13}n^5) + O(r^{13}n^4(\log r + \log n)^2)$,

$\mathbf{ptime}(\mathcal{B}_5) = O(\log r + \log(\log r + \log n))$.

Указанное количество процессоров превышает ограничение теоремы, а значит необходимо заново оценить время этой процедуры при ограниченном числе процессоров.

Рассмотрим этот шаг подробнее. Процедура $\mathcal{B}_5$ состоит из двух подпунктов:

i) для любых $A \in GF(q)^4$ вычислим $B_A = \mathcal{B}_3(A, \vec{A}, \vec{B}, \vec{u})$;
ii) для любых $l \in SL_4(U)$ вычислим $B_l = \mathcal{B}_4(l, \vec{A}, \vec{B}, \vec{u})$.

Для первого подпункта необходимо $|GF(q)^4| \cdot \mathbf{bp}(\mathcal{B}_3)$, процессоров, которые выполнят его за время $\mathbf{ptime}(\mathcal{B}_3)$, а для второго подпункта требуется $|SL_4(U)| \cdot \mathbf{bp}(\mathcal{B}_4)$, процессоров и время выполнения равно $\mathbf{ptime}(\mathcal{B}_4)$. Как было отмечено выше, количество процессоров у нас ограничено величиной $|GF(q)^4| + |SL_4(U)| \leqslant O(r^{12}n^4)$. Выделим из этой величины $|GF(q)^4|$ процессоров на первый подпункт и $|SL_4(U)|$ на второй. Так как $|GF(q)^4| \cdot \mathbf{bp}(\mathcal{B}_3)$ процессоров выполняет первый подпункт за время $\mathbf{ptime}(\mathcal{B}_3)$, то $|GF(q)^4|$ процессоров выполнит эту же процедуру за время $\mathbf{ptime}(\mathcal{B}_3) \cdot \mathbf{bp}(\mathcal{B}_3)$. Аналогично для второго подпункта получаем время $\mathbf{ptime}(\mathcal{B}_4) \cdot \mathbf{bp}(\mathcal{B}_4)$. Воспользуемся для процедур $\mathcal{B}_3$ и $\mathcal{B}_4$ оценками из [1] и [2].

$\mathbf{bp}(\mathcal{B}_3) = O(r \cdot (k^2 + n))$ и $\mathbf{ptime}(\mathcal{B}_3) = O(\log k + \log r)$,

$\mathbf{bp}(\mathcal{B}_4) = O(r \cdot (k + n))$ и $\mathbf{ptime}(\mathcal{B}_4) = O(\log k + \log r)$.

Таким образом, при ограниченном количестве процессоров

$\mathbf{ptime}(\mathcal{B}_5) = O(r \cdot (k^2 + n) \cdot (\log k + \log r)) + O(r \cdot (k + n) \cdot (\log k + \log r)) = O(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n)))$.

На третьем шаге алгоритма происходит обращение к базе данных. Наше ограничение на число процессоров устанавливалось исходя из этого шага, поэтому новых процессоров не требуется. Время выполнения оценивается как **extime**.

Рассмотрим четвертый шаг (процедура $\mathcal{B}_7$). Для всех $i = 1, 2, ..., r$ вычисляется

$$f(A_i, B_i) = \mathcal{B}_6(A_i, u_i, \vec{Z}, \vec{Y}) =$$
$$= \left( \bigvee_{l \in SL_4(U)} Z_l \cdot \mathcal{B}_2(A, u, l) \right) \oplus \left( \bigoplus_{A^* \in GF(q)^4} Y_{A^*} \cdot \mathcal{B}_1(A, u, A^*) \right).$$

Оценки сложности этой процедуры до ограничения количества процессоров были таковы

$\mathbf{bp}(\mathcal{B}_7) = O(s \cdot r^{13} n^4 (\log r + \log n)^2)$, $\mathbf{ptime}(\mathcal{B}_7) = O(\log r + \log n)$.

Указанное количество процессоров превышает ограничение теоремы, поэтому необходимо заново определить величину $\mathbf{ptime}(\mathcal{B}_7)$ при ограниченном количестве процессоров.

Для вычисления каждого произведения $Z_l \cdot \mathcal{B}_2(A, u, l)$ в алгоритме $DP_{r,f}$ использовалось $\mathbf{bp}(\mathcal{B}_2)$ процессоров и происходило это за время $\mathbf{ptime}(\mathcal{B}_2)$, а для любого произведения $Y_{A^*} \cdot \mathcal{B}_1(A, u, A^*)$ использовалось $\mathbf{bp}(\mathcal{B}_1)$ процессоров и затрачивалось время $\mathbf{ptime}(\mathcal{B}_1)$. Произведений первого типа должно быть вычислено $|SL_4(U)|$, а произведений второго типа $|GF(q)^4|$. Однако, по ограничению теоремы у нас всего $|GF(q)^4| + |SL_4(U)|$ процессоров, то есть по одному процессору на произведение каждого типа. На вычисление каждого произведения $Z_l \cdot \mathcal{B}_2(A, u, l)$ одним процессором будет затрачено время $\mathbf{bp}(\mathcal{B}_2) \cdot \mathbf{ptime}(\mathcal{B}_2)$, а на каждое $Y_{A^*} \cdot \mathcal{B}_1(A, u, A^*)$ время $\mathbf{bp}(\mathcal{B}_1) \cdot \mathbf{ptime}(\mathcal{B}_1)$. Таким образом, все указанные произведения будут вычислены за время $\max(\mathbf{bp}(\mathcal{B}_2) \cdot \mathbf{ptime}(\mathcal{B}_2), \mathbf{bp}(\mathcal{B}_1) \cdot \mathbf{ptime}(\mathcal{B}_1))$. Далее легко доказать, что время вычисления всех дизъюнкций оценивается как $O(\log |SL_4(U)|)$, а на вычисление всех сумм по модулю два необходимо времени $O(\log |GF(q)^4|)$. Следовательно, время выполнения всей процедуры $\mathcal{B}_6$ можно оценить как

$\mathbf{ptime}(\mathcal{B}_6) = \max(\mathbf{bp}(\mathcal{B}_2) \cdot \mathbf{ptime}(\mathcal{B}_2), \mathbf{bp}(\mathcal{B}_1) \cdot \mathbf{ptime}(\mathcal{B}_1)) + \max(O(\log |SL_4(U)|), O(\log |GF(q)^4|))$.

Из [2] мы имеем следующие оценки

$\mathbf{bp}(\mathcal{B}_1) = O((\log r + \log n)^2)$,

$\mathbf{ptime}(\mathcal{B}_1) = O(\log(\log r + \log n))$,

$\mathbf{bp}(\mathcal{B}_2) = O(\log r + \log n)$,

$\mathbf{ptime}(\mathcal{B}_2) = O(\log(\log r + \log n))$,

$\log |SL_4(U)| = O(\log r + \log n), \log |GF(q)^4| = O(\log r + \log n)$.

Отсюда получаем, что

$\mathbf{ptime}(\mathcal{B}_6) = O((\log r + \log n)^2 \cdot \log(\log r + \log n))$

Окончательно, для четвертого шага алгоритма определяем

$\mathbf{ptime}(\mathcal{B}_7) = O(r \cdot (\log r + \log n)^2 \cdot \log(\log r + \log n))$.

Время работы всего алгоритма при ограниченном количестве процессоров определяется как

$\mathbf{ptime}(DP_{r,f}) = \mathbf{ptime}(\mathcal{A}) + \mathbf{ptime}(\mathcal{B}_5) + \mathbf{extime} + \mathbf{ptime}(\mathcal{B}_7) = O(\log r + \log(\log r + \log n)) + O(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n))) + \mathbf{extime} + O(r \cdot (\log r + \log n)^2 \cdot \log(\log r + \log n)) = O(r \cdot (n + (\log r + \log n)^2) \cdot (\log r + \log(\log r + \log n))) + \mathbf{extime}$.

Что и требовалось доказать.

Очевидно, что размер базы данных не увеличится при ограничении количества процессоров.То есть

$\mathbf{mem}(DP_{r,f}) \leqslant 2^n \left(1 + \frac{1}{n}\right)$.

Следовательно, окончательно получаем следующие новые оценки алгоритма $DP_{r,f}$ при ограниченном количестве процессоров:

$\mathbf{ptime}(DP_{r,f}) = O(r(n + \log^2 r + \log^2 n + \log r \log n)(\log r + \log n))$,

$\mathbf{bp}(DP_{r,f}) = O(r^{12} n^4)$,

$\mathbf{mem}(DP_{r,f}) \leqslant 2^n \left(1 + \frac{1}{n}\right)$.

## Список литературы

[1] Andreev A.E., Clementi A.E.F., Rolim J.D.P. On the parallel computation of Boolean functions on unrelated inputs // Proc. of 4-th Israeli Symposium on Theory of Computing and Systems (ISTCS'96). 1996.

[2] Andreev A.E., Clementi A.E.F., Rolim J.D.P. A parallel algorithm for the Direct sum problem with optimal use of external memory, and its consequences in circuit complexity.