А. В. Афанасьева, В. Б. Балакирский, С. В. Беззатеев, Протокол конфиденциального получения информации, *Матем. вопр. криптогр.*, 2015, том 6, выпуск 4, 5–21

DOI: https://doi.org/10.4213/mvk165

МАТЕМАТИЧЕСКИЕ ВОПРОСЫ КРИПТОГРАФИИ
2015 Т. 6 № 4 С. 5–21

УДК: 004.738+004.724/.728+004.7.057.4

Протокол конфиденциального получения информации

А. В. Афанасьева, В. Б. Балакирский, С. В. Беззатеев

Санкт-Петербургский государственный университет аэрокосмического приборостроения, Санкт Петербург

Получено 20.V.2015

Предложен новый вычислительно-эффективный протокол конфиденциального получения информации, основанный на свойствах орбит действия групп Галуа конечных расширений поля GF(q). Коммуникационная сложность протокола немного больше сложности лучших известных схем, основанных на локально-декодируемых кодах, но он может быть построен для любых параметров системы (в отличие от кодовых конструкций). Вычислительная сложность протокола меньше, чем у протоколов, основанных на арифметике полиномов, что важно для серверов, обслуживающих запросы от множества пользователей.

Ключевые слова: конфиденциальная передача информации, протоколы связи, интерполяция полиномов, группы Галуа, конечные поля

Private information retrieval protocol

A. V. Afanasieva, V. B. Balakirskii, S. V. Bezzateev

Saint-Petersburg State University of Aerospace Instrumentation, Saint-Petersburg

Abstract. A new computationally efficient private information retrieval protocol is proposed. It is based on coset properties of Galois groups of the field GF(q) finite extensions. The proposed protocol has communication complexity slightly worse than the best known schemes based on locally decodable codes and it may be constructed for any system parameters (as opposed to codes). In comparison with similar solutions based on polynomials the computational complexity of our method is smaller which is important especially for servers processing multiple requests from multiple users.

Key words: private information retrieval protocol, polynomial interpolation, coset, Galois groups, finite fields

Citation: *Mathematical Aspects of Cryptography*, 2015, vol. 6, no. 4, pp. 5–21 (Russian).

1. Введение

Задача конфиденциального получения информации из удаленного хранилища возникла и приобрела большое значение с распространением «облачных» хранилищ данных. Формально данная задача может быть описана следующим образом: существует база данных $X = (x_1, x_2, \ldots, x_n)$ длиной n бит, расположенная на удаленном сервере; владелец данных хочет получить один бит информации x_i из своей базы данных X так, чтобы сервер не узнал ничего о том, с какой позиции i был запрошен бит.

Впервые данная задача была рассмотрена Шором, Голдрайхом, Кушелевицем и Суданом в 1995 г. [2], в той же работе авторами был описан первый протокол конфиденциального получения информации. Данный протокол требовал хранить несколько копий базы данных на разных серверах, которые не могут общаться друг с другом. При выполнении этих условий протокол гарантировал теоретико-информационную стойкость системы, то есть по полученному запросу ни один сервер, даже вычислительно неограниченный, не мог получить никакой информации о запрашиваемой позиции. В этой же работе было доказано, что не существует коммуникационно-эффективного решения без дублирования базы данных на нескольких серверах. В более общем виде эта задача была переформулирована следующим образом: в ходе предобработки n-битная строка X увеличивается по длине до l бит и копируется на r серверов. Пользователь запрашивает у каждого сервера значение ровно с одной позиции и вычисляет запрашиваемое значение. Ни один сервер не может получить никакой информации ни о полученном значении, ни о его истинной позиции.

Вторая модель атакующего и соответственно обеспечиваемый уровень стойкости системы были сформулированы в работе Шора и Гилбоа [3]. Эта модель предполагает наличие вычислительно ограниченного атакующего, способного на реализацию лишь полиномиальных алгоритмов. Криптографическая стойкость схем в данной модели обеспечивается только тем фактом, что запрашиваемая позиция i вычислительно скрыта от атакующего. В данной модели безопасности протокол для одного сервера предложили Эйал Кушелевиц и Рафаил Островский [4] в 1997 г. Основная идея данного и всех последующих решений заключалась в том, что пользователь предлагал хранителю базы данных вычислить некоторую функцию от всей базы данных. Чтобы вычислить запрашиваемую позицию или ее значение по полученному результату, необходимо знать некоторый секрет или решить сложную задачу. Для односерверных систем было доказано, что их криптографическая стойкость может основываться только на вычислительной сложности, то есть не

существует решений в рамках теоретико-информационной модели безопасности.

Любой протокол конфиденциального получения информации включает в себя следующие этапы.

- Этап 1: инициализация. Функция, которая устанавливает все публичные параметры системы и производит все предвычисления с базой данных. Функцию выполняет владелец базы данных.
- Этап 2: генерация запроса. Функция, которая формирует запрос к серверу, хранящему базу данных. Функцию выполняет владелец базы данных для каждого запрашиваемого значения.
- Этап 3: вычисление отклика. Функция, которая вычисляет отклик сервера на запрос, опираясь на строку запроса и хранимую базу данных. Функцию выполняет сервер, хранящий базу данных, для каждого присланного запроса.
- Этап 4: вычисление результата запроса. Функция, которая вычисляет запрошенное значение по отклику сервера. Функцию выполняет владелец базы данных для каждого запрашиваемого значения.

Для оценки сложности протоколов конфиденциального получения информации используются две составляющие: вычислительная сложность — это количество вычислений, необходимых серверу и пользователю для извлечения одного бита информации; коммуникационная сложность — это количество данных, которое должно быть передано между пользователем и сервером (серверами) для извлечения одного бита информации.

2. Существующие решения

Как было отмечено во введении, все существующие на данный момент протоколы конфиденциального извлечения информации делятся на две большие группы по модели безопасности: схемы, обладающие теоретико-информационной и вычислительной стойкостью. Так как нас интересуют только теоретико-информационные модели безопасности, то в данном разделе будут рассмотрены решения, предлагаемые для двух и более серверов. Необходимо отметить, что в последнее время в рамках ослабленной модели безопасности было предложено множество решений для одного сервера с очень низкой коммуникационной сложностью, но с очень высокой вычислительной сложностью, в первую очередь на стороне клиента. Эти решения можно разделить на группы по типу сложной задачи, на которой они базируются. Первые протоколы были предложены на основе задачи о квадратичных

вычетах [4, 5]; следующий тип задач — скрывающие Ф-функции [6]; использование гладких подгрупп позволило авторам извлекать информацию блоками, вместо бит [7–9]; последние результаты были получены с использованием решеток [10].

Многосерверные протоколы извлечения информации имеют ряд неоспоримых достоинств: для них может быть доказана теоретико-информационная стойкость, они имеют низкую сложность вычислений (как на клиентской, так и на серверной стороне), что является существенным фактором для практической реализации. Минимальную коммуникационную сложность дает подход, основанный на локально-декодируемых кодах (детальный обзор кодов можно найти в [13]), однако на данный момент не существует конструктивных алгоритмов построения таких кодов с произвольными параметрами.

На данный момент известно три класса локально-декодируемых кодов [13]. Для них были оценены коммуникационные сложности протоколов для сравнения с предлагаемым в данной работе решением. Полученные оценки приведены в табл. 1. В данной таблице, а также во всей последующей работе логарифмы имеют основание 2 и обозначаются $\log(x)$ для уменьшения громоздкости формул.

Таблица 1. Коммуникационная сложность протоколов извлечения информации для различных классов кодов

Класс кодов	Коммуникационная сложность
Коды Рида–Малера	$O(k^{1/(r-1)})$
Множественные коды	$O(\sqrt{k})$
Совпадающие векторы	$2^{O(\log(k)^{1/t}(\log\log(k))^{1-1/t})},\ r=2^t$

С точки зрения асимптотического анализа наилучшие результаты получены для кодов, построенных с использованием совпадающих векторов.

Помимо решений на базе кодовых методов [11] можно выделить также решения, использующие комбинаторные подходы [2] и арифметику полиномов [1, 12]. При этом целью работы [2] является минимизация количества использованных в системе серверов, а работы [1] — приблизиться по коммуникационной сложности к кодовым методам, однако это требует от серверов значительных вычислений. Цель данной работы — модификация схемы, предложенной в [1], для получения решения, минимизирующего вычислительную сложность системы и сохраняющего коммуникационную сложность, близкую к кодовым методам.

3. Описание новой схемы

Как было указано во введении, в стандартной постановке задачи база данных рассматривается как битовая строка, из которой должен быть извлечен один бит. Однако необходимо отметить, что в более поздних работах данная постановка была обобщена для произвольного q-ичного алфавита и извлечения блока заданной длины. В этой статье мы опишем решение и для работы с блоками заданной длины.

3.1. Конфиденциальное извлечение q-ичного символа

Предлагаемое решение базируется на свойствах многочленов в поле характеристики q и будет описано согласно стандартным этапам работы протоколов конфиденциального получения информации.

3.1.1. Инициализация

Используется предложенное в предыдущих работах [1, 12] инъективное отображение множества индексов $j \in \{1, 2, \ldots, n\}$ базы данных в множество двоичных векторов $\boldsymbol{u}_j$ размерности l и веса w:

$$\begin{gathered} j \to \boldsymbol{u}_j = (u_j^{(0)}, u_j^{(1)}, \ldots, u_j^{(l-1)}),\ u_j^{(i)} \in \{0,1\} \subseteq GF(q), \\ wt(\boldsymbol{u}_j) = w, \binom{l}{w} \geq n. \end{gathered} \tag{1}$$

При этом количество серверов в системе, хранящих базу данных, должно быть не менее $w + 1$.

Аналогично некоторым описанным ранее схемам каждому вектору $\boldsymbol{u}_j$ ставится в соответствие моном $m_j = z_0^{u_j^{(0)}} z_1^{u_j^{(1)}} \cdots z_{l-1}^{u_j^{(l-1)}}$. Пример такого отображения приведен в разделе 3.4.

Теперь может быть задано описание базы данных в следующем виде:

$$X = (x_1, \ldots, x_n) \to F(z_0, z_1, \ldots, z_{l-1}) = \sum_{j=1}^{n} x_j m_j = \sum_{j=1}^{n} x_j \prod_{i=0}^{l-1} z_i^{u_j^{(i)}}. \tag{2}$$

Последней выбираемой характеристикой на данном этапе является степень расширения m поля $GF(q)$. Значение m непосредственно определяется количеством серверов в системе и должно обеспечивать выполнение неравенства

$$(w+1)m \leq |\{\alpha \in GF(q^m) | GF(q)(\alpha) = GF(q^m)\}|. \tag{3}$$

Для построения схемы разобьем множество $P_m = \{\alpha \in GF(q^m) | GF(q)(\alpha) = GF(q^m)\}$ на орбиты O_i действия группы Галуа расширения $GF(q^m)$ поля $GF(q)$. Каждая из этих орбит содержит в точности m элементов. Так как в соответствии с неравенством (3) должно выполняться соотношение $|P_m| \geq (w+1)m$, то число орбит должно быть не меньше величины $w+1$, определяющей минимальное число серверов в системе, и $\bigcup_{i=1}^{w+1} O_i \subseteq P_m$.

Для каждого сервера S_i необходимо выбрать элемент $\alpha_i \in GF(q^m)$ таким образом, чтобы каждому серверу соответствовала своя орбита $O_i = \{\alpha_i^{q^k} | k = 0, \ldots, m-1\}$.

Утверждение 3.1. *Для любого числа серверов $w+1$ всегда найдется такое m, что $|P_m| \geq (w+1)m$.*

Доказательство. Действительно, множество P_m содержит все примитивные элементы поля $GF(q^m)$, число которых определяется функцией Эйлера $\varphi(q^m - 1)$ [14]. Используя теорему Ландау [15] и результат из [16] (теорема 15, с. 71), получаем следующую оценку снизу для функции Эйлера:

$$\varphi(q^m - 1) > \frac{q^m - 1}{e^c \log\log(q^m - 1) + \frac{2{,}50637}{\log\log(q^m - 1)}},$$

где c — положительная константа. Отсюда получим неравенство для m:

$$\frac{q^m - 1}{m(e^c \log\log(q^m - 1) + \frac{2{,}50637}{\log\log(q^m - 1)})} \geq w + 1. \qquad \square$$

3.1.2. Генерация запроса

Для извлечения j-го q-ичного блока x_j (2) необходимо сгенерировать случайную q-ичную матрицу C размера $m \times l$, общую для всех серверов. При помощи этой матрицы для каждого сервера S_i следует сформировать запрос. Для этого потребуется выполнить следующие шаги.

1. Построить базис B_i для сервера S_i при помощи выбранного на этапе инициализации элемента α_i:

$$B_i = \left[\alpha_i,\ \alpha_i^2,\ \alpha_i^3, \ldots, \alpha_i^m\right].$$

2. Для двоичного вектора $\boldsymbol{u}_j = \left(u_j^{(0)},\ u_j^{(1)}, \ldots, u_j^{(l-1)}\right)$, полученного отображением индекса j, построить матрицу

$$U_j = \left[u_j^{(0)} \cdot \alpha^0, \quad u_j^{(1)} \cdot \alpha^0, \ldots u_j^{(l-1)} \cdot \alpha^0\right].$$

Чтобы построить эту матрицу, необходимо элемент поля $\alpha^0 \in GF(q^m)$ представить в виде столбца из m элементов поля $GF(q)$. При этом будем использовать базис поля $GF(q^m)$ над $GF(q)$, у которого первым элементом является 1. Таким образом, будет получена матрица

$$U_j = \begin{bmatrix} u_j^{(0)} & u_j^{(1)} & \ldots & u_j^{(l-1)} \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

3. Вычислить матрицу запроса по формуле

$$R_i = U_j + B_i C.$$

При этом базис B_i можно представить в виде матрицы из элементов поля $GF(q)$ аналогично U_j.

Полученная в результате матрица запроса R_i может быть переписана в виде вектора размерности l над полем $GF(q^m)$ следующим образом:

$$R_i = \left[u_j^{(0)}\alpha^0 + \sum_{k=1}^{m} c_{k1}\alpha_i^k, \quad u_j^{(1)}\alpha^0 + \sum_{k=1}^{m} c_{k2}\alpha_i^k, \quad \ldots, \quad u_j^{(l-1)}\alpha^0 + \sum_{k=1}^{m} c_{kl}\alpha_i^k\right],$$

где c_{kj} — это элемент k-й строки и j-го столбца в матрице C.

3.1.3. Вычисление отклика

Каждый сервер S_i поэлементно подставляет полученный вектор запроса R_i в функцию базы данных $F(z_0, z_1, \ldots, z_{l-1})$ и вычисляет значение $F(R_i)$ в поле $GF(q^m)$.

Полученный результат вычисления значения функции в виде элемента поля $GF(q^m)$ возвращается клиенту.

3.1.4. Вычисление результата запроса

Клиент получает $w+1$ значений функции $F(R_i), i = 1, \ldots, w+1$, от $w+$ $+1$ серверов. Так как каждый элемент вектора запроса R_i является функцией от α_i, то клиент рассматривает полученные значения не как значения функции l переменных $F(z_0, z_1, \ldots, z_{l-1})$, а как значения в точке $x = \alpha_i$ функции одной переменной $\widehat{F}(x)$, определенной над полем $GF(q)$.

Основываясь на свойствах выбранных орбит $O_i, i = 1, \ldots, w+1$, клиент может вычислить значения функции $\widehat{F}(x)$ в $(m-1)(w+1)$ различных дополнительных точках. Для этого он вычисляет:

$$\widehat{F}(\alpha_i^q) = (\widehat{F}(\alpha_i))^q,\ \widehat{F}(\alpha_i^{q^2}) = (\widehat{F}(\alpha_i))^{q^2}, \ldots, \widehat{F}(\alpha_i^{q^{m-1}}) = (\widehat{F}(\alpha_i))^{q^{m-1}},$$

$i = 1, \ldots, w+1$. Таким образом может быть получено $m(w+1)$ значений полинома $\widehat{F}(x)$ в различных точках.

По полученным значениям при помощи процедуры интерполяции Лагранжа может быть вычислено значение $\widehat{F}(0)$, равное свободному члену полинома $\widehat{F}(x)$. Полученное значение и есть результат запроса x_j.

3.2. Доказательство корректности протокола

Покажем, что в результате работы протокола интерполяция выполняется однозначно и что получаемое значение равно искомому блоку x_j.

Теорема 3.1. *В результате выполнения протокола пользователь системы единственным образом восстанавливает значение $\hat{F}(0)$, и это значение равно запрашиваемому блоку.*

Доказательство. Рассмотрим подробно выполняемую сервером S_s процедуру поэлементной подстановки вектора запроса

$$R_s = \left[u_j^{(0)}\alpha^0 + \sum_{k=1}^{m} c_{k1}\alpha_s^k,\quad u_j^{(1)}\alpha^0 + \sum_{k=1}^{m} c_{k2}\alpha_s^k,\quad \ldots,\quad u_j^{(l-1)}\alpha^0 + \sum_{k=1}^{m} c_{kl}\alpha_s^k \right]$$

в функцию базы данных

$$F(z_0, z_1, \ldots, z_{l-1}) = \sum_{i=1}^{n} x_i m_i = \sum_{i=1}^{n} x_i \prod_{k=0}^{l-1} z_k^{u_i^{(k)}} =$$
$$= x_1 z_0^{u_1^{(0)}} z_1^{u_1^{(1)}} \cdots z_{l-1}^{u_1^{(l-1)}} + x_2 z_0^{u_2^{(0)}} z_1^{u_2^{(1)}} \cdots z_{l-1}^{u_2^{(l-1)}} + \cdots + x_n z_0^{u_n^{(0)}} z_1^{u_n^{(1)}} \cdots z_{l-1}^{u_n^{(l-1)}}. \tag{4}$$

При выполнении подстановки сервером S_s каждая переменная z_i в соответствии с вектором запроса представляется следующим образом:

$$z_i = u_j^{(i)}\alpha^0 + \sum_{k=1}^{m} c_{k,i+1}\alpha_s^k,\ i = 0, \ldots, l-1.$$

При этом каждый моном m_i при x_i будет представлен полиномом от α_s:

$$m_1 = \prod_{i=0}^{l-1}\left(u_j^{(i)}\alpha^0 + \sum_{k=1}^{m} c_{k,(i+1)}\alpha_s^k\right)^{u_1^{(i)}},$$
$$m_2 = \prod_{i=0}^{l-1}\left(u_j^{(i)}\alpha^0 + \sum_{k=1}^{m} c_{k,(i+1)}\alpha_s^k\right)^{u_2^{(i)}},$$
$$\ldots$$

После выполнения умножения и приведения всех подобных слагаемых каждый моном можно будет представить полиномом от элемента α_s степени, не большей mw (так как мы перемножаем w многочленов от элемента α_s степени, не превышающей m; остальные $l-w$ многочленов возводятся в нулевую степень и равны 1), например:

$$\begin{aligned} m_1 &= \prod_{i=0}^{l-1}\left(u_j^{(i)}\alpha^0 + \sum_{k=1}^{m} c_{k,(i+1)}\alpha_s^k\right)^{u_1^{(i)}} = \\ &= g_{mw}^{(1)}\alpha_s^{mw} + \ldots + g_1^{(1)}\alpha_s + \prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_1^{(i)})}, \quad g_i^{(k)} \in GF(q). \end{aligned} \tag{5}$$

При этом по построению (1) вектора $\boldsymbol{u_j}$

$$\prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_k^{(i)})} = \begin{cases} 1, & \text{если } k = j, \\ 0 & \text{в противном случае.} \end{cases}$$

Принимая во внимание (5) и определение $\boldsymbol{u}_j$, функцию l переменных (4) можно переписать в виде многочлена $\widehat{F}(\alpha_s)$ от одной переменной:

$$\begin{aligned} \widehat{F}(\alpha_s) &= x_1\left(g_{mw}^{(1)}\alpha_s^{mw} + \ldots + g_1^{(1)}\alpha_s + \prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_1^{(i)})}\right) + \ldots + \\ &+ x_n\left(g_{mw}^{(n)}\alpha_s^{mw} + \ldots + g_1^{(n)}\alpha_s + \prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_n^{(i)})}\right) = \\ &= \hat{f}_{mw}\alpha_s^{mw} + \ldots + \hat{f}_1\alpha_s + x_1\prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_1^{(i)})} + \ldots + x_n\prod_{i=0}^{l-1}\left(u_j^{(i)}\right)^{(u_n^{(i)})} = \\ &= \hat{f}_{mw}\alpha_s^{mw} + \ldots + \hat{f}_1\alpha_s + x_j, \hat{f}_i \in GF(q). \end{aligned} \tag{6}$$

Из (6) следует, что каждый сервер S_s на своей стороне вычисляет значение многочлена $\widehat{F}(x)$ в точке α_s, при этом $\deg(\widehat{F}(x)) \leq mw$, а свободный член $\hat{\hat{f}}_0 = x_j$, так как произведения $\prod_{i=0}^{l-1} \left(u_j^{(i)}\right)^{(u_k^{(i)})}$ при остальных x_k, $k \neq j$, обращаются в 0 по свойствам вектора $\boldsymbol{u}_j$ (1). Таким образом, для интерполяции данного многочлена необходимо иметь его значения в $mw+1$ различных точках. От серверов будут получены значения в $w+1$ точках, а недостающие значения будут вычислены на стороне клиента путем возведения имеющихся значений в степень q:

$$\widehat{F}(\alpha_s^q) = (\widehat{F}(\alpha_s))^q, \widehat{F}(\alpha_s^{q^2}) = (\widehat{F}(\alpha_s))^{q^2}, \ldots, \widehat{F}(\alpha_s^{q^{m-1}}) = (\widehat{F}(\alpha_s))^{q^{m-1}}.$$

Равенства будут верны, так как полином определен над полем $GF(q)$, при этом значения будут получены в различных точках поля $GF(q^m)$ благодаря свойствам орбит $O_i, i = 1, \ldots, w+1$, выбранных для каждого сервера S_i.

Получив значения многочлена $\widehat{F}(x)$ в $mw+1$ различных точках, пользователь может однозначно восстановить значение $\widehat{F}(x)$ в точке $x = 0$ при помощи интерполяционной формулы Лагранжа. □

3.3. Доказательство теоретико-информационной стойкости

В данном разделе рассматриваются вопросы криптографической стойкости в рамках модели отдельных невзаимодействующих серверов.

Как было показано для других многосерверных протоколов, в рамках модели невзаимодействующих серверов может быть доказана теоретико-информационная стойкость подобных систем. Докажем, что предлагаемый протокол конфиденциального извлечения информации является таким же стойким.

В статье К. Шеннона [17] было введено понятие совершенной секретности, которое в дальнейшем получило название «теоретико-информационная стойкость». К. Шеннон определил это понятие с помощью следующего условия: для всех перехваченных криптограмм апостериорные вероятности различных сообщений равны априорным вероятностям независимо от величины этих последних. В этом случае перехват зашифрованного сообщения не дает противнику никакой информации.

Таким образом, для предлагаемого протокола может быть сформулирована следующая теорема.

Теорема 3.2. *Для каждого сервера S_i априорное распределение вероятностей на множестве запрашиваемых позиций базы данных $X = (x_1, x_2, \ldots, x_n)$ остается неизменным после получения им запроса R_i на значение $x_j \in X$.*

Доказательство. По протоколу в качестве запроса каждый сервер S_i получает матрицу

$$R_i = U_j + B_i C. \tag{7}$$

При этом матрица C выбирается случайным образом из равномерного распределения на множестве $GF(q)^{m \times l}$, а B_i — это базис конечного поля $GF(q^m)$, представленный в матричном виде над полем $GF(q)$.

До получения запроса все позиции базы данных для сервера равновероятны, так как пользователь может запросить любую.

После получения вектора запроса R_i сервер S_i может для любого U_j найти уникальное решение:

$$C = B_i^{-1}(R_i - U_j).$$

Это решение удовлетворяет равенству (7) и принадлежит множеству допустимых значений C. А каждый вектор U_j однозначно сопоставлен позиции x_j. Таким образом, после каждого отдельного запроса R_i распределение вероятностей на X для сервера остается прежним. □

В связи с этим ни один сервер после получения любого количества запросов не может отличить запрашиваемые у него позиции между собой. Таким образом, даже вычислительно неограниченный сервер по множеству запросов не может получить никакой информации о запрашиваемой позиции.

3.4. Численный пример работы протокола

Проиллюстрируем работу предложенного протокола на простом числовом примере.

Пусть база данных представлена строкой

$$X = (x_1, x_2, \ldots, x_6) = (1, 0, 1, 1, 0, 0)$$

и хранится на $r = 3$ серверах.

1. **Начальная инициализация**

   Отображение множества позиций может иметь следующий вид: $j \to \to \boldsymbol{u}_j = (u_j^{(0)}, \ldots, u_j^{(3)})$,

$$X \to F(z_0, z_1, z_2, z_3) = \sum_{i=1}^{6} x_i \cdot m_i = z_0 z_1 + z_1 z_2 + z_0 z_3.$$

В качестве расширения двоичного поля выберем $GF(2^4)$ с примитивным полиномом $g(x) = x^4 + x + 1$.

Таблица 2. Таблица отображения

j	$\boldsymbol{u}_j$	Моном (m_j)
1	1100	$z_0 z_1$
2	1010	$z_0 z_2$
3	0110	$z_1 z_2$
4	1001	$z_0 z_3$
5	0101	$z_1 z_3$
6	0011	$z_2 z_3$

Таблица 3. Таблица элементов $GF(2^4)$

i	α^i	i	α^i	i	α^i
0	0001	5	0110	10	0111
1	0010	6	1100	11	1110
2	0100	7	1011	12	1111
3	1000	8	0101	13	1101
4	0011	9	1010	14	1001

В выбранном расширении поля для $GF(2^4)$ каждого сервера назначим элементы α_i. Для S_1 установим $\alpha_1 = \alpha$ и, соответственно, $\alpha_2 = \alpha^3$ и $\alpha_3 = \alpha^7$.

2. **Генерация запроса для получения второго бита** ($j = 2$)

Согласно выбранному отображению (табл. 2) $u_2^{(0)} = 1, u_2^{(1)} = 0, u_2^{(2)} = = 1, u_2^{(3)} = 0$ и, следовательно, вектор запроса равен $\boldsymbol{u}_2 = (1, 0, 1, 0)$.

В поле $GF(2^4)$ вектор отображения позиции $\boldsymbol{u}_2$ может быть переписан в двоичном виде как матрица U_2 размера 4×4:

$$U_2 = [u_2^{(0)}\alpha^0,\ \ u_2^{(1)}\alpha^0,\ \ u_2^{(2)}\alpha^0,\ \ u_2^{(3)}\alpha^0] = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Для представления U_2 в двоичном виде элемент поля $\alpha^0 \in GF(2^4)$ был представлен в виде столбца из m элементов $GF(2)$.

Для каждого сервера S_i вычисляется замаскированный запрос:

$$R_i = U_2 + B_i C.$$

Случайная матрица $C = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$.

Таким образом, для первого сервера S_1 запрос имеет вид

$$R_1 = U_2 + B_1 C = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} + B_1 C,$$

где $B_1 = [\alpha, \alpha^2, \alpha^3, \alpha^4] = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ и $O_1 = \{\alpha, \alpha^2, \alpha^4, \alpha^8\}$.

В результате получаем $R_1 = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix} = [\alpha^7, \alpha^{12}, \alpha^8, \alpha^5]$.

Аналогичным образом для остальных серверов S_2 и S_3 получаем

$$B_2 = [\alpha^3, \alpha^6, \alpha^9, \alpha^{12}], \quad O_2 = \{\alpha^3, \alpha^6, \alpha^{12}, \alpha^9\},$$
$$B_3 = [\alpha^7, \alpha^{14}, \alpha^6, \alpha^{13}], \quad O_3 = \{\alpha^7, \alpha^{14}, \alpha^{13}, \alpha^{11}\}.$$

Таким образом,

$$R_2 = [\alpha^4, \alpha^{14}, \alpha^{13}, \alpha^2],$$
$$R_3 = [\alpha^5, \alpha^3, \alpha^3, \alpha].$$

Полученные в результате вычислений запросы отправляются соответствующим серверам.

3. **Отклик сервера**

Каждый сервер при получении запроса на своей стороне вычисляет значение функции $F(R_i)$ в поле $GF(2^4)$.

Тогда получается, что S_1 вычисляет: $F(R_1) = F(\alpha^7, \alpha^{12}, \alpha^8, \alpha^5) = \alpha^7\alpha^{12} + \alpha^{12}\alpha^8 + \alpha^7\alpha^5 = \alpha^9$.

S_2 вычисляет: $F(R_2) = F(\alpha^4, \alpha^{14}, \alpha^{13}, \alpha^2) = \alpha^3\alpha^6 + \alpha^6\alpha^9 + \alpha^3\alpha^{12} = \alpha^7$.

S_3 вычисляет: $F(R_3) = F(\alpha^5, \alpha^3, \alpha^3, \alpha) = \alpha^5\alpha^3 + \alpha^3\alpha^3 + \alpha^5\alpha = \alpha^8$.

Каждый сервер S_i отправляет свой ответ клиенту.

4. **Извлечение второго бита на стороне клиента**

Пользователь системы хранит у себя предвычисленные значения множителей Лагранжа для точек α_i и их орбит O_i:

$$\{\lambda_{0,1}, \lambda_{0,2}, \lambda_{0,3}, \lambda_{0,4}, \lambda_{0,6}, \lambda_{0,7}, \lambda_{0,8}, \lambda_{0,9}, \lambda_{0,12}\} = \\ = \{\alpha, \alpha^{11}, \alpha^{12}, \alpha^6, \alpha^0, \alpha^4, \alpha^0, \alpha^4, \alpha^{11}\}.$$

На первом шаге вычислений клиент из полученных от серверов значений многочлена от l переменных $F(z_0, z_1, \ldots, z_{l-1})$, рассматривая их как точки многочлена одной переменной $\widehat{F}(x)$, рассчитывает недостающие значения в дополнительных точках.

Из значения $\widehat{F}(\alpha) = F(R_1) = \alpha^9$, полученного от сервера S_1, клиент может вычислить еще 3 значения в 3 различных точках $\left(\widehat{F}(\alpha)\right)^2 = \widehat{F}(\alpha^2) = \alpha^3$, $\left(\widehat{F}(\alpha)\right)^4 = \widehat{F}(\alpha^4) = \alpha^6$, $\left(\widehat{F}(\alpha)\right)^8 = \widehat{F}(\alpha^8) = \alpha^{12}$.

Аналогичным образом для сервера S_2 из полученного значения $\widehat{F}(\alpha^3) = F(R_2) = \alpha^7$ можно вычислить $\left(\widehat{F}(\alpha^3)\right)^2 = \widehat{F}(\alpha^6) = \alpha^{14}$, $\left(\widehat{F}(\alpha^3)\right)^4 = \widehat{F}(\alpha^{12}) = \alpha^{13}$, $\left(\widehat{F}(\alpha^3)\right)^8 = \widehat{F}(\alpha^9) = \alpha^{11}$.

Для третьего сервера S_3 дополнительные вычисления не требуются, так как полученное от него значение $\widehat{F}(\alpha^7) = F(R_3) = \alpha^8$ оказывается девятым значением функции $\widehat{F}(x)$ и этого количества достаточно для восстановления полинома.

Таким образом, было получено 9 значений $\widehat{F}(x)$ в 9 различных точках, и этого достаточно для интерполяции $\widehat{F}(0) = x_2$ при помощи предвычисленных множителей Лагранжа:

$$x_2 = \widehat{F}(0) = \sum_{i \in \{1,2,3,4,6,7,8,9,12\}} \lambda_{0,i} \cdot \widehat{F}(\alpha^i) = 0.$$

Данный пример наглядно демонстрирует технику применения предлагаемого решения. Параметры схемы выбраны с единственной целью — простого и наглядного вычисления. Поэтому можно заметить, что в данном случае по коммуникационной сложности схема проигрывает полной передаче базы данных. Это является следствием особенности большинства рассматриваемых схем, которые дают выигрыш только при больших размерах базы

данных N (в нашем случае когда выполняется условие $\sqrt{N} > wm$), в противном случае для маленьких длин наиболее простое решение — возвратить всю базу данных.

4. Сравнительный анализ

Для проведения сравнительного анализа предложенной схемы с существующими решениями наиболее значимые параметры приведены в табл. 4. Согласно приведенной сравнительной таблице можно сделать следующие выводы: предложенный протокол имеет меньшую вычислительную сложность, чем схема Woodruff – Yekhanin, при сравнимой коммуникационной сложности. Следовательно, данный алгоритм позволяет снизить вычислительную нагрузку как на клиента, так и на сервер.

Таблица 4. Сравнительная таблица

Параметры	Схема Woodruff – Yekhanin [1]	Схема на совпадающих векторах [13] при $r = 3 \cdot 2^{t-2}$	Предлагаемое решение
Коммуникационная сложность	$O(r^2 \log(r) N^{1/(2r-1)})$	$2^{(\log(N))^{1/t}(\log\log(N))^{1-1/t}}$	$O(N^{\frac{1}{r}})$
Сложность хранения	N	$2^{2^{(\log(N))^{1/t}(\log\log(N))^{1-1/t}}}$	N
Вычислительная сложность:			
на стороне сервера	$O(r^2 N^{2r/(2r-1)})$	$O(1)$	$O(rN)$
на стороне клиента	$O(r^2 N^{1/(2r-1)})$	$O(r^3)$	$O(r^2)$

Наилучшую коммуникационную сложность из всех приведенных решений имеет подход, основанный на кодах, построенных с использованием совпадающих векторов.

Для сравнения этих двух схем оценим отношение их коммуникационных сложностей для случая $t = 2, r = 3$:

$$\lim_{N\to\infty} \frac{N^{\frac{1}{3}}}{\exp\left((\log(N))^{1/2}(\log\log(N))^{1-1/2}\right)}.$$

Этот предел равен $+\infty$, что свидетельствует о том, что в асимптотике числитель растет быстрее знаменателя. Следовательно, предлагаемое в данной работе решение проигрывает в коммуникационной сложности. Однако в реальных условиях длина строки X никогда не достигает бесконечности.

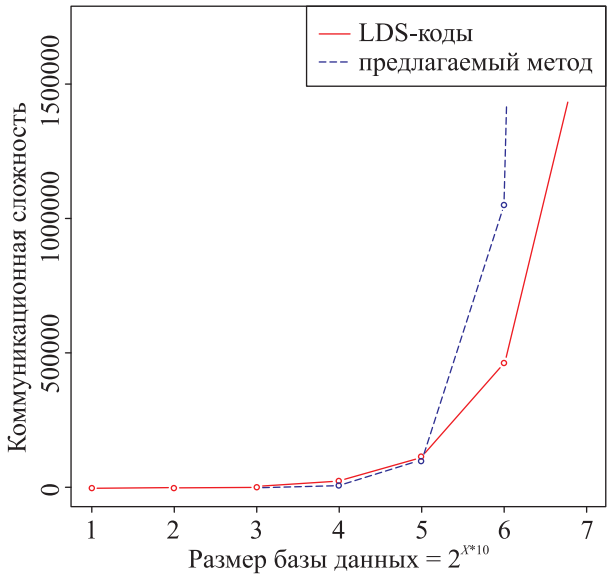

Рис. 1. Зависимость коммуникационной сложности от размера базы данных

Если построить графики сравниваемых величин (рис. 1), то можно заметить, что выигрыш кодового подхода начинается с размера базы данных в 2^{50} бит.

5. Заключение

Авторы выражают искреннюю признательность рецензенту за полезные замечания и рекомендации, позволившие существенно улучшить представление материалов данной работы.

Список литературы

[1] Woodruff D., Yekhanin S., “A geometric approach to information-theoretic private information retrieval”, *SIAM J. Comput.*, **37**:4 (2007), 1046–1056.

[2] Chor B., Kushilevitz E., Goldreich O., Sudan M., “Private information retrieval”, In Proc. 36th Annu. IEEE Symp. on Foundations of Computer Science, 1995, 41–50.

[3] Chor B., Gilboa N., “Computationally private information retrieval”, 29th STOC, 1997, 304–313.

[4] Kushilevitz E., Ostrovsky R., “Replication is not needed: Single database, computationally-private information retrieval”, In Proc. of the 38th Annu. IEEE Symp. on Foundations of Computer Science, 1997, 364–373.

[5] Ostrovsky R. et al., *Method and apparatus for private information retrieval from a single electronic storage device*, US Patent 6167392.

[6] Cachin C., Micali S., Stadler M., "Computationally Private Information Retrieval with Polylogarithmic Communication", In Proceedings of EUROCRYPT'99, *Lect. Notes Comput. Sci*, **1592** (1999), 402–414.

[7] Gentry C., Ramzan Z., "Single-database private information retrieval with constant communication rate", In ICALP: Annu. Int. Colloq. on Automata, Languages and Programming, 2005, 803–815.

[8] Ramzan et al., *Method and apparatus for communication efficient private information retrieval and oblivious transfer*, US Patent 8065332 B2.

[9] Groth J., Kiayias A., and Lipmaa H., "Multi-query computationally-private information retrieval with constant communication rate. Practice and Theory in Public Key Cryptography — PKC 2010", *Lect. Notes Comput. Sci.*, **6056** (2010), 107–123.

[10] Melchor C. A., Gaborit P., *A lattice-based computationally-efficient private information retrieval protocol*, 2007, arXiv: http://eprint.iacr.org/2007/446.pdf.

[11] Beimel A., Ishai Y., Kushilevitz E., "General constructions for information-theoretic private information retrieval", *J. Comput. Syst. Sci.*, **71** (2005), 213–247.

[12] Beimel A., Ishai Y., Kushilevitz E., Raymond J., "Breaking the $O(n^{1/(2k-1)})$ barrier for information-theoretic private information retrieval", In 43rd IEEE Symposium on Foundations of Computer Science (FOCS), 2002, 261–270.

[13] Yekhanin S., "Locally decodable codes", *Foundations and Trends in Theoretical Computer Science*, **7**:1 (2011), 1–117.

[14] Мак-Вильямс Ф. Дж., Слоэн Н. Дж., *Теория кодов, исправляющих ошибки*, М.: Связь, 1979.

[15] Landau E., *Handbuch der Lehre yon der Verteilung der Primzahlen*, Reprinted in 1953 by Chelsea Publishing Co., New York, **1**, Teubner, Leipzig, 1909.

[16] Rosser J. B., Schoenfeld L., "Approximate formulas for some functions of prime numbers", *Illinois J. Math.*, **6**:1 (1962), 64–94.

[17] Шеннон К., *Работы по теории информации и кибернетике*, М.: Иностр. лит., 1963.