ПРОГРАММИРОВАНИЕ, 2024, № 6, с. 12–23

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

УДК 004.056.5

АЛГОРИТМЫ РАЗМЕЩЕНИЯ И ЗАПРОСА К КОНФИДЕНЦИАЛЬНЫМ ДАННЫМ НА ОБЛАКЕ

© 2024 г. Н. П. Варновский[a], *, С. А. Мартишин[b], **, М. В. Храпченко[b], *, А. В. Шокуров[b], ****

[a] *Институт проблем информационной безопасности МГУ им. М.В. Ломоносова 119192 Москва, Мичуринский проспект, 1, офис 10, Россия*

[b] *Институт системного программирования имени В.П. Иванникова РАН 109004 Москва, ул. А. Солженицына, д. 25, Россия*

* *E-mail: otd13isp@gmail.com*

** *E-mail: mart@ispras.ru*

*** *E-mail: khrap@ispras.ru*

**** *E-mail: shok@ispras.ru*

Поступила в редакцию 10.07.2024 г.
После доработки 17.07.2024 г.
Принята к публикации 17.07.2024 г.

Авторами рассматривается задача PIR (Private Information Retrieval) обеспечения безопасных запросов к базе данных. Ранее авторы рассматривали задачу для базы данных, размещенной на облаке при наличии активного противника, который не вмешивается в выполнение протокола, но может производить атаку с известными открытыми запросами. В предложенных алгоритмах номер бита i представляется в системе счисления по основанию l с числом разрядов d. Предложен алгоритм размещения базы данных на облаке и алгоритм запроса требуемого бита с использованием перестановок в цифрах разряда номера бита, при задании номера бита i в системе счисления по основанию l. Перестановки рассматриваются как секретные ключи шифрования. Приведены оценка коммуникационной сложности и оценки вероятности угадывания номера бита при однократной атаке с известным открытым запросом номера бита i и при атаке с неограниченным числом известных открытых запросов.

Ключевые слова: база данных, облачные вычисления, PIR

DOI: 10.31857/S0132347424060027, **EDN:** DZDHBD

1. ВВЕДЕНИЕ

В работах [1, 2] авторы исследовали задачу организации конфиденциальных запросов к базе данных (PIR – Private Information Retrieval) в случае размещения реплицированной базы данных на облаке и наличия активного противника, работающего по протоколу, но имеющему возможность производить атаку с известными открытыми запросами.

Задача PIR в информационно-теоретической постановке была сформулирована в 1995 г. Шором, Голдрайхом, Кушелевицем и Суданом [3].

Классическая постановка задачи PIR:

• имеется база данных – бинарная строка $X = (x_1, ..., x_n)$ длины n, хранящаяся на сервере в облаке;

• клиент хочет получить один бит информации x_i из базы данных X так, чтобы никто, кроме клиента, обращающегося с запросом, не смог определить, с какой позиции i был запрошен бит.

В работах [3, 4] было показано, что если база данных размещена на единственном сервере, который полностью контролируется противником, то теоретико-информационное условие конфиденциальности корректного протокола PIR может быть выполнено только том случае, когда клиент запрашивает базу данных целиком. Однако в этом случае коммуникационная сложность протокола, т. е. общее количество бит, которыми обмениваются участники протокола за время его работы, будет не меньше размера базы данных, что делает его практически не применимым.

Для уменьшения коммуникационной сложности в работах [3, 4] была предложена модель реплицированной базы данных, в которой несколько одинаковых копий строки $X = (x_1, ..., x_n)$

размещалось на разных серверах. Предполагалось, что клиент имеет связь с каждым из этих серверов, но сами серверы не имеют связи друг с другом. Также предполагалось, что противник может наблюдать любой из серверов, на которых размещена база данных, причем, возможно, разные серверы во время выполнения разных сеансов протокола.

Очевидно, что в облачных информационных системах нельзя обеспечить изолированность копий распределенной базы данных друг от друга. Поскольку ни пользователь, ни клиент не могут контролировать облако, то всегда предполагается наличие на облаке противника. С учетом этого выбирается модель для дальнейших исследований. Важное отличие от классической модели, в рассматриваемой модели противники на разных серверах могут общаться.

В работах [1, 2] предлагалась модель, включающая в себя облако, состоящее из нескольких серверов, на каждом из которых хранится копия одной и той же базы данных, дилера, центра аутентификации, пользователей, клиентов, противника.

Серверы в облаке соединены посредством незащищенных каналов связи друг с другом и дилером. По этим каналам серверы обмениваются информацией между собой и дилером. Облачные серверы и незащищенные каналы связи доступны для стороннего наблюдателя. Дилер находится вне облака, противнику недоступен. Основной функцией дилера являлось шифрование и дешифрование информации при работе с облаком. Предполагалось, что противник не только пассивно наблюдает, но производит атаку с известными открытыми запросами.

В работах [1, 2] была получена оценка коммуникационной сложности работы протокола, а также оценки вероятности угадывания противником, работающим по протоколу, номера бита i при однократной атаке и при атаке с неограниченным числом известных открытых запросов.

В отличие от работ [1, 2] в данной статье предлагается вместо шифрования использовать перестановки номеров битов.

Предложены алгоритмы, основанные на перестановках, для представления номера бита в новой системе счисления, формирования запроса к облаку, обработки ответа облака и получения значения искомого бита. Даны новые оценки коммуникационной сложности алгоритмов и вероятности угадывания противником номера запрошенного бита.

В отличие от рассматриваемых в [1, 2] алгоритмах предлагается: использовать перестановки цифр номера бита в выбранной системе счисления вместо шифрования. В новом алгоритме поиск противником рядом стоящих запрашиваемых битов затруднен, и замена сдвига в интервале множества битов на перестановку не позволяет противнику сделать вывод по одному запрашиваемому множеству об остальных не пересекающихся с этим множеством множествах.

Получены оценки вероятности угадывания номера бита при однократной атаке с известным открытым запросом номера бита i и при атаке с неограниченным числом известных открытых запросов при наличии противника на облаке. Приведена оценка коммуникационной сложности алгоритма.

2. МОДЕЛЬ ВЫЧИСЛЕНИЙ, ОСНОВНЫЕ ОПРЕДЕЛЕНИЯ И ПОСТАНОВКА ЗАДАЧИ

2.1. Состав модели

Состав модели вычислений, для которой в данной статье исследуется облачный вариант задачи PIR, аналогичен составу модели, рассмотренной в [2].

Модель включает в себя:

Облако. Состоит из нескольких серверов, хранящих копии одной и той же базы данных. Копии на серверах имеют разные перестановки номеров битов. Серверы соединены посредством незащищенных каналов связи друг с другом и дилером.

Пользователь. Хранит данные на облаке. Предполагается, что на облаке хранится k копий баз данных. Для загрузки данных пользователь обращается к дилеру по каналу связи, использующему стандартный криптографический протокол для обмена зашифрованными данными.

Клиенты. Запрашивают некоторую информацию из базы данных. Обращаются для выполнения запроса к дилеру. С дилером соединены каналами связи, использующими стандартный криптографический протокол для обмена зашифрованными данными.

Центр аутентификации. Аутентифицирует пользователя и клиентов.

Дилер. Находится вне облака, противнику недоступен. Канал связи с облаком является незащищенным. Объем памяти дилера для постоянного хранения данных пренебрежимо мал по сравнению с n. Получив от клиента данные для размещения на облаке, дилер выполняет перестановку уникальную для каждой копии базы данных, размещает данные на облаке. При запросе клиента дилер возвращает клиенту значение.

В дальнейшем для простоты будем считать, что база данных загружается одним пользователем.

Предполагается, что после загрузки база данных не может быть изменена. Либо изменена полностью. Также будем рассматривать случай запроса одного бита.

2.2. Основные определения

Противник. Не вмешивается в выполнение криптографического протокола. Имеет доступ к базе данных на облаке, к каждой ее копии, к каналам связи на облаке. Может создавать фальшивых клиентов, работающих по протоколу. То есть противник не только пассивно наблюдает, но и сам может производить атаку с известными открытыми запросами. Противник может заставить дилера отправить запрос на облако. При этом противник знает алгоритм формирования запроса к облаку, но не знает конкретных параметров. Противник также знает алгоритм формирования ответа клиенту дилером на основании ответа, полученного дилером от облака.

Угроза: угадывание исходного номера бита, запрашиваемого клиентом в базе данных.

Атака:

• Атака пассивного противника на облаке заключается в наблюдении за запросом от дилера и ответом облака дилеру. Эта информация используется для сбора статистики и анализа.

• Атака активного противника (атака с известными открытыми запросами) заключается в создании фальшивых клиентов и управление ими. При помощи фальшивых клиентов активный противник может сформировать произвольное число запросов, что позволит в коалиции с пассивным противником на облаке собрать и проанализировать информацию для всей базы данных.

2.3. Постановка задачи и основные обозначения

В отличие от классической постановки задачи PIR будем рассматривать базу данных (бинарная строка) $X = (x_0, ..., x_{n-1})$ длины n, где элементы X нумеруются с нуля. Это делает белее удобным работу с различными системами счисления.

Клиент хочет получить один бит информации x_i с номером i из базы данных X так, чтобы противник не узнал ничего о том, с какой позиции i был запрошен бит.

Будем размещать на облаке k копий баз данных. Пусть $k = 2^d$, где $d \geq 2$. На практике $4 \leq k \leq 32$. В дальнейшем будем предполагать, что на k наложены эти ограничения.

Без потери общности предполагается, что $n = l^d$, т. е. $d = \log_2 k$ и $l = \sqrt[d]{n}$. Заметим, что поскольку l-основание системы счисления, то $l \geq 2$ и d подбирается так, чтобы $l \ll n$. Пусть $L_p = \dfrac{n}{l}$.

Так как $l = \sqrt[d]{n}$, округлим l до целого числа в большую сторону.

Пусть x – элемент группы Z_2, обозначим через $h\left(h \in \{1, \ldots, k\}\right)$ – номер копии базы данных.

3. ПРЕДСТАВЛЕНИЕ НОМЕРА БИТА l-ИЧНОЙ СИСТЕМЕ СЧИСЛЕНИЯ

Представим номер бита как d разрядов в l-ичной системе счисления. Выбор l и d основывается на известной модели [1, 2], где номер i бита x_i представлен в l-ичной записи: $i = a_0^{(i)} + a_1^{(i)} l + \ldots + a_{d-1}^{(i)} l^{d-1}$. Цифры l-ичной записи представляют собой элементы кортежа $\left(a_0^{(i)}, \ldots, a_{d-1}^{(i)}\right)$ длины d, т. е. на j-м месте кортежа стоит l-ичная цифра.

Эти d элементов кортежа можно интерпретировать как точку с целочисленными координатами в гиперкубе размерности d и длиной стороны l, где $l = \sqrt[d]{n}$. Такое множество точек куба размерности d можно рассматривать как множество слов длины d с алфавитом $(0, ..., l-1)$.

4. ЗАГРУЗКА ДАННЫХ НА ОБЛАКО

4.1. Инициализация массива X^c

Напомним, что $n = l^d$ и $L_p = \dfrac{n}{l}$. Выберем целые числа $d_u \geq 2$ и $l_u \geq 2$ такие, чтобы $l_u^{d_u} \leq L_p$ и обозначим $L_p^- = l_u^{d_u}$ $(L_p^- \leq L_p)$. Пусть $l^* = \left\lceil \dfrac{n}{L_p^-} \right\rceil$, где $l^* \geq l$ и $n_u = l^* L_p^-$. Ниже будет показано, что коммуникационная сложность зависит от l^*. Для того чтобы коммуникационная сложность не возрастала, нужно подобрать значения d_u и l_u так, чтобы минимизировать разность $L_p - L_p^-$. Конкретный выбор значений d_u и l_u, который не только минимизирует разность $L_p - L_p^-$, но и уменьшает количество вычислительных операций, производимых дилером, будет описан ниже.

Поскольку $n_u \geq n$, добавим в базу данных $n_u - n$ битов. Добавленные биты могут быть заполнены произвольными значениями. Биты будут добавлены в конец базы данных.

Пусть l_c и d_c целые числа, такие, что $n_c = l_c^{d_c}$ и $n_c \geq n_u$, тогда $n_c \geq n$.

На облако будет загружаться массив $X^c = \left(x_0^c, \ldots, x_{n_c-1}^c\right)$, где $n_c \geq n$, т. е. на облако будет

загружено n_c битов. Нумерация массива X^c начинается с нуля. Увеличение числа элементов в массиве необходимо, чтобы вместо шифрования битов использовать их секретную перестановку. Это позволяет уменьшить объем памяти, необходимой дилеру для хранения данных и снизить коммуникационную сложность.

4.2. Построение матрицы перестановок

Номер запрашиваемого бита в l_c-ичной системе счисления можно представить как целочисленную точку гиперкуба размерности d_c с длиной стороны l_c. При этом j-я ($j \in \{1,\dots,d_c\}$) координата точки гиперкуба является j-м разрядом представления числа в l_c-ичной системе счисления.

Изменим номера битов путем изменения цифр в разрядах номеров в l_c-ичном представлении. Цифры в каждом разряде изменяются с помощью перестановок цифр от 0 до $l_c - 1$. Для каждого разряда вычисляется своя перестановка. Поскольку перестановки являются случайными, то для разных разрядов возможно, что перестановки совпадают.

Выберем копию базы данных $h \in \{1,\dots,k\}$. Пусть β_{hj} – случайная перестановка из l_c чисел от 0 до $l_c - 1$, где $j \in \{1,\dots,d_c\}$ – номер разряда числа в l_c-ичной системе счисления. Представим перестановку чисел в виде таблицы:

$$\begin{pmatrix} 0 & \dots & l_c - 1 \\ \beta_{hj}(0) & \dots & \beta_{hj}(l_c - 1) \end{pmatrix}.$$

Тогда $\beta_h = \{\beta_{h1},\dots,\beta_{hd_c}\}$ – множество "поразрядных" перестановок числа в l_c-ичной системе счисления.

Построим матрицу M_β размера $k \times d_c$, где k – число копий базы данных, а d_c – целое число, такое, что $n_c = l_c^{d_c}$. Элементом β_{ij} матрицы M_β является случайная перестановка целых чисел от 0 до $l_c - 1$. Таким образом, для каждой копии базы данных будет построено d_c случайных перестановок.

Для генерации перестановок цифр номера бита l_c-ичной системе счисления используется датчик псевдослучайных чисел $PRNG(db, m)$. На вход $PRNG(db, m)$ подается уникальное значение db для конкретной базы данных и номер копии базы данных m. Датчик псевдослучайных чисел при одинаковой инициализации генерирует одинаковую перестановку.

Поскольку матрица M_β требует для хранения значительного объема памяти, матрица M_β при необходимости восстанавливается с помощью датчика псевдослучайных чисел.

4.3. Отображение $i \xrightarrow{\beta_h} i_c$

Номер бита i представляем в системе счисления по основанию l_c с числом разрядов d_c:

$$i = \sum_{j=1}^{d_c} a_j \cdot l_c^{j-1}, \text{ где } a_j \in \{0,\dots,l_c - 1\}.$$

Для копии базы данных с номером h выполним перестановки для разрядов числа i:

$$i_c = \sum_{j=1}^{d_c} \beta_{hj}(a_j) \cdot l_c^{j-1}.$$

Обозначим такое преобразование через $i \xrightarrow{\beta_h} i_c$, где $\beta_h = \{\beta_{h1},\dots,\beta_{hd_c}\}$ – строка матрицы M_β. Необходимые для работы алгоритма элементы матрицы M_β генерируются с помощью датчика псевдослучайных чисел.

4.4. Вспомогательные построения, необходимые для работы алгоритма запроса бита – алгоритм построения матрицы G

Построим матрицу G размера $n_c \times 2$:

$$G = \begin{pmatrix} 0 & x_0^c \\ \vdots & \vdots \\ n_c - 1 & x_{n_c-1}^c \end{pmatrix}.$$

Первый столбец этой матрицы – последовательность n_c целых чисел $[0, n_c - 1]$. Второй столбец матрицы G – элементы $X^c = (x_0^c,\dots,x_{n_c-1}^c)$.

4.5. Формирование матриц G_h для каждой копии базы данных из матрицы G

Алгоритм формирования матриц G_h

Вход: матрица G.

Выход: матрицы G_h для всех $h \in \{1,\dots,k\}$.

Шаг 1. Выберем копию базы данных h.

Шаг 2. Скопируем матрицу G в матрицу G_{aux}.

Шаг 3. Для каждого элемента i, где $i = 0,\dots,n_c - 1$ первого столбца матрицы G_{aux} выполним преобразование $i \xrightarrow{\beta_h} i_c$.

Шаг 4. Отсортируем строки матрицы G_{aux} по первому столбцу. Получим матрицу G_h.

Шаг 5. Выполним Шаги 1–4 для всех $h \in \{1,\dots,k\}$ ■.

В результате работы алгоритма получим G_h $(h \in \{1,\dots,k\})$ матриц для каждой копии базы данных.

4.6. Загрузка данных на облако

Алгоритм загрузки данных на облако аналогичен алгоритму, приведенному в [1, 2].

На вход алгоритма подается бинарный массив $X = (x_1, ..., x_n)$. Для удобства будем предполагать, что $X = (x_0, ..., x_{n-1})$. В результате работы алгоритма на облако загружается k преобразованных с помощью перестановки копий баз данных.

Алгоритм загрузки данных на облако

Вход: X.

Выход: загрузка на облако G_h, где $h \in \{1, \ldots, k\}$.

Шаг 1. Дилер аутентифицирует пользователя. Пользователь передает дилеру массив значений битов $X = (x_0, ..., x_{n-1})$. Дилер преобразует массив $X = (x_0, ..., x_{n-1})$ в массив $X^c = \left(x_0^c, \ldots, x_{n_c-1}^c\right)$.

Шаг 2. Дилер формирует матрицу $G = \|g_{i,j}\|$, где $i = 0, \ldots, n_c - 1,\ j = 1, 2$.

Шаг 3. Дилер генерирует элементы матрицы M_β в процессе работы.

Шаг 4. Дилер создает матрицу G_h, где $h \in \{1, \ldots, k\}$.

Шаг 5. Дилер загружает матрицу G_h, где $h \in \{1, \ldots, k\}$ на облако.

Шаг 6. Шаги 4, 5 дилер выполняет в цикле для каждой копии базы данных. По окончании перебора k копий базы данных происходит выход из алгоритма ■.

Число элементов, переданных каждой из k копий базы данных равно $2n_c$.

Заметим, что поскольку в дальнейшем восстановление матрицы G не предполагается, матрицу G дилер не хранит.

Для выполнения перестановки дилеру требуется иметь объем памяти, сравнимый с объемом, необходимым для хранения одной копии базы данных. При этом предполагается, что дилер может работать с несколькими базами данных и нет необходимости их одновременного хранения. После загрузки базы данных на облако память дилера очищается для работы со следующей базой данных.

В приведенном ниже алгоритме дилер должен выполнять Шаги 4 и 5 для каждой копии базы данных последовательно. Иначе ему придется хранить в памяти все k копий базы данных, что приводит к большим расходам памяти.

5. ВЫПОЛНЕНИЕ ДИЛЕРОМ ЗАПРОСА КЛИЕНТА

Для сокрытия при запросе искомого номера бита, вместо одного запрашиваемого номера бита дилер будет запрашивать у облака множество номеров битов мощностью l^*. Причем только один из этих номеров битов был запрошен клиентом. Каждый элемент из множества номеров битов лежит только в одном из интервалов мощности L_p^- разбиения этого множества.

5.1. Использование множества номеров битов в запросе для сокрытия номера бита

Пусть клиент интересуется элементом $x_i \in X$. Этому элементу соответствует i-я строка матрицы G, причем $0 \le i < n$. Отрезок целых чисел от 0 до $n_c - 1$ содержит l^* интервалов по L_p^- элементов в каждом из них.

Номер i лежит ровно в одном из этих интервалов. Далее для этого элемента i выберем по одному элементу в каждом из оставшихся $l^* - 1$ интервалов (алгоритм выбора элементов описан в п. 5.2). Таким образом, каждому номеру i поставим в соответствие другие $l^* - 1$ чисел, лежащих в различных интервалах и не попадающих в интервал, где лежит само число i.

Как было сказано выше, каждый интервал можно интерпретировать как дискретный гиперкуб размерности d_u с длиной стороны l_u.

Выберем интервал $m \in \{1, \ldots, l^*\}$. Пусть σ_{mj} – случайная перестановка из l_u чисел от 0 до $l_u - 1$, где $j \in \{1, \ldots, d_u\}$ – разряд числа в l_u-ичной системе счисления. Как и ранее, запишем перестановку чисел в виде

$$\begin{pmatrix} 0 & \ldots & l_u - 1 \\ \sigma_{mj}(0) & \ldots & \sigma_{mj}(l_u - 1) \end{pmatrix}.$$

Тогда $\sigma_m = \left\{\sigma_{m1}, \ldots, \sigma_{md_u}\right\}$ – множество “поразрядных” перестановок числа в l_u-ичной системе счисления d_u-разрядного числа.

Построим матрицу M_σ (по аналогии с матрицей M_β) размера $l^* \times d_u$, где l^* – число интервалов, а d_u – целое число, такое, что $L_p^- = l_u^{d_u}$. Элементом σ_{ij} матрицы M_σ является случайная перестановка целых чисел от 0 до $l_u - 1$. Таким образом, для каждого интервала на которые разбивается база данных будет построено d_u случайных перестановок.

Для генерации перестановок цифр номера бита матрицы M_σ используется датчик псевдослучайных чисел $PRNG(db, m, j)$. На вход $PRNG(db, m, j)$ подается уникальное значение db для конкретной базы данных, номер интервала m (всего l^* штук) и $j \in \{1, \ldots, d_u\}$ – разряд числа в l_u-ичной системе счисления. Датчик псевдослучайных чисел при одинаковой инициализации генерирует одинаковую перестановку.

Поскольку матрица M_σ требует для хранения значительного объема памяти, элементы матрицы M_σ восстанавливаются с помощью датчика псевдослучайных чисел.

Поскольку отрезок целых чисел от 0 до $n_c - 1$ разбивается на l^* интервалов по L_p^- элементов в каждом интервале, то любое число из этого множества попадет в один из таких интервалов.

Найдем номер интервала w, в который попало число i при разбиении отрезка целых чисел от 0 до $n_c - 1$ на l^* интервалов:

$$w = \left\lfloor \frac{i}{L_p^-} \right\rfloor + 1.$$

Заметим, что $w \in \{1, \ldots, l^*\}$.

Приведем i по модулю L_p^-. Обозначим через $i' = i \bmod L_p^-$ результат приведения. Тогда i' является порядковым номером в интервале с номером w.

Представим порядковый номер i' в интервале с номером w в l_u-ичной системе счисления:

$$i' = \sum_{j=1}^{d_u} a_j \cdot l_u^{j-1}, \text{ где } a_j \in \{0, \ldots, l_u - 1\}.$$

Рассмотрим сумму $\sum_{j=1}^{d_u} \left(\sigma_{wj}\right)^{-1}\left(a_j\right) \cdot l_u^{j-1}$, где $\left(\sigma_{wj}\right)^{-1}\left(a_j\right)$ – обратная перестановка, такая, что $\sigma_{wj}\left(\left(\sigma_{wj}\right)^{-1}\left(a_j\right)\right) = a_j$.

Обозначим через $D_j = \left(\sigma_{wj}\right)^{-1}\left(a_j\right)$, величина D_j – секрет, известный только дилеру.

В каждом интервале $m\left(m \in \left[1, l^*\right]\right)$ найдем число, такое, что $i'(m) = \sum_{j=1}^{d_u} \sigma_{mj}\left(D_j\right) \cdot l_u^{j-1}$, где $j = 1, \ldots, d_u$. Для всех интервалов таких чисел $i'(m)$ будет l^*. Заметим, что для интервала w это будет число i'.

Обозначим через Set_i полученное по алгоритму 5.2 (Алгоритм построения множества Set_i дилером) множество из l^* чисел.

5.2. Построение множеств Set_i дилером

Алгоритм построения множества Set_i дилером

Вход: X^c, i, l^*, d_u, L_p^-.
Выход: множество Set_i.
$Set_i = \varnothing$
// найдем номер интервала

$$w \leftarrow \left\lfloor \frac{i}{L_p^-} \right\rfloor + 1$$

// найдем порядковый номер числа i
// в соответствующем интервале разбиения
$i' = i \bmod L_p^-$
// найдем представление числа i' в l_u-ичной записи

$$i' = \sum_{j=1}^{d_u} a_j \cdot l_u^{j-1},\ a_j \in \{0, \ldots, l_u - 1\}$$

// найдем координаты $D_j \in \{0, \ldots, l_u - 1\}$
for $j \leftarrow 1$ to d_u do

$$D_j = \left(\sigma_{wj}\right)^{-1}\left(a_j\right)$$

for $m \leftarrow 1$ to l^* do

$$i'(m) = \sum_{j=1}^{d_u} \sigma_{mj}\left(D_j\right) \cdot l_u^{j-1}$$

$$Set_i = Set_i \cup \left(L_p^- m + i'(m)\right)$$

Листинг 1. Алгоритм построения множества Set_i

5.3. Запрос клиентом бита с номером i. Предварительные замечания

Пусть клиент запрашивает бит с номером i.

После подтверждения центром аутентификации полномочий клиента дилер разрешает клиенту отправить запрос.

Напомним, что физически в облаке хранятся матрицы G_h, $h \in \{1, \ldots, k\}$. Строка матрицы G_h состоит из номера строки и значения бита. Без потери общности (как было сказано выше) будем индексировать элементы базы данных с 0.

Так же как и в [1, 2] для каждого элемента множества Set_i случайно и равномерно выбирается номер копии базы данных. Все элементы множества Set_i для которых выбрана копия базы данных h, обозначим через множества Set_i^h, где $h \in \{1, \ldots, k\}$. Очевидно, что множества Set_i^h не пересекаются между собой, поскольку все элементы множества Set_i различны и каждому элементу ставится в соответствие ровно один номер h (номер копии базы данных). Объединение множеств Set_i^h содержит все элементы множества Set_i, т. е. $Set_i = \cup Set_i^h$, где $h \in \{1, \ldots, k\}$.

Пусть для запрашиваемого номера i в множестве Set_i была выбрана копия базы данных h.

Для каждого элемента множества Set_i^h выполняется перестановка β^h, пусть $y = \beta^h(i)$. Полученные после перестановки элементы сортируются. Это множество обозначим через $Set_i^{\beta^h}$. Позиция i_{pos} элемента $y = \beta^h(i)$ в множестве $Set_i^{\beta^h}$ запоминается.

Дилер на время выполнения запроса формирует вектор-строку $\boldsymbol{s}_{num}$ из трех элементов $\boldsymbol{s}_{num} = (s_1, s_2, s_3)$:

• Первый элемент вектора-строки $\boldsymbol{s}_{num}$ содержит номер элемента i.

• Второй элемент вектора-строки $\boldsymbol{s}_{num}$ содержит номер копии h базы данных, в которую попал номер s_1.

• Третий элемент вектора-строки $\boldsymbol{s}_{num}$ содержит позицию i_{pos} номера y в множестве $Set_i^{\beta^h}$.

Вектор-строка $\boldsymbol{s}_{num}$ позволяет дилеру после получения ответа от облака сразу отбросить данные, кроме данных, необходимых для выполнения запроса.

Вектор-строка $\boldsymbol{s}_{num}$ формируется у дилера и известна только дилеру. Вектор-строка $\boldsymbol{s}_{num}$ является секретной.

На k копий базы данных дилер отправляет в общей сложности l^* номеров.

После выполнения запроса дилер с помощью вектора-строки $\boldsymbol{s}_{num}$ получает значение искомого элемента и отправляет его клиенту.

Алгоритм подготовки запроса к облаку

Вход: X^c, i, l^*, d_u, L_p^-.

Выход: $Set_i^{\beta^h}$, $\boldsymbol{s}_{num}$.

Шаг 1. Клиент обращается к дилеру и запрашивает значение бита с номером i.

Шаг 2. Дилер генерирует множество Set_i.

Шаг 3. Дилер на время выполнения запроса i-го бита резервирует память для вектора-строки $\boldsymbol{s}_{num}$ трех элементов.

Шаг 4. Дилер заносит i в первый элемент вектора-строки $\boldsymbol{s}_{num}$.

Шаг 5. Дилер выполняет разбиение множества Set_i на непересекающиеся подмножества Set_i^h, где $h \in \{1,\ldots,k\}$, путем случайного и равномерного выбора номера копии базы данных для каждого элемента множества Set_i.

Дилер заносит во второй элемент вектора-строки $\boldsymbol{s}_{num}$ номер копии базы данных s_2, соответствующей элементу s_1.

Шаг 6. Дилер выполняет перестановку β^h для элементов множества Set_i^h после чего полученные после перестановки новые номера сортируются. Получаем множество $Set_i^{\beta^h}$. Этот шаг выполняется для всех $h \in \{1,\ldots,k\}$.

Шаг 7. Дилер заносит в третий элемент вектора-строки $\boldsymbol{s}_{num}$ позицию i_{pos} номера $y = \beta^{s_2}(i)$ в множестве $Set_i^{\beta^h}$ ■.

5.4. Выполнение запроса к облаку и ответ облака

Алгоритм обмена информацией с облаком

Вход: $Set_i^{\beta^h}\left(h \in \{1,\ldots,k\}\right)$.

Выход: значения битов в том же порядке, в котором передавались номера битов множества $Set_i^{\beta^h}$ для каждой копии базы данных $\left(h \in \{1,\ldots,k\}\right)$.

Шаг 1. Для каждой копии базы данных дилер отправляет на облако множества $Set_i^{\beta^h}$.

Шаг 2. Для запрошенных номеров битов облако возвращает дилеру значения битов из второго столбца матрицы G_{β^h}, где $h \in \{1,\ldots,k\}$). Порядок возвращаемых значений битов соответствует исходному порядку элементов множества $Set_i^{\beta^h}$ ■.

5.5. Обработка ответа облака

Алгоритм обработки ответа облака

Вход: $\boldsymbol{s}_{num}$, значения битов в том же порядке, в котором передавались номера битов множества $Set_i^{\beta^h}$ для каждой копии базы данных $\left(h \in \{1,\ldots,k\}\right)$.

Выход: значение бита i.

Шаг 1. Дилер с помощью вектора-строки $\boldsymbol{s}_{num}$ выбирает нужное значения бита. Остальные значения битов отбрасываются.

Шаг 2. Дилер получает значение бита для первого элемента вектора-строки $\boldsymbol{s}_{num}$. По построению вектора-строки $\boldsymbol{s}_{num}$ запрашиваемый номер является элементом s_1.

Шаг 3. Дилер отправляет значение i бита клиенту ■.

6. ОЦЕНКА ТРЕБУЕМОЙ ПАМЯТИ И СЛОЖНОСТИ АЛГОРИТМА

6.1. Объем информации, который необходимо хранить дилеру

На протяжении всего времени существования базы данных дилер хранит информацию для базы данных объема n_c:

• значение db для инициализации датчиков псевдослучайных чисел $PRNG(db, m)$ и $PRNG(db, m, j)$;

• значения k, d_c и l_c для датчика псевдослучайных чисел $PRNG(db, m)$, генерирующего матрицу M_β размера $k \times d_c$. Элементом матрицы M_β является случайная перестановка целых чисел от 0 до $l_c - 1$;

• значения k, d_u, l_u и l^* для датчика псевдослучайных чисел $PRNG(db, m, j)$, генерирующего матрицу M_σ размера $l^* \times d_u$. Элементом матрицы M_σ является случайная перестановка целых чисел от 0 до $l_u - 1$.

Объем хранимой дилером информации много меньше n.

6.2. Характеристики предложенной схемы

В данной схеме рассматривается активный противник, работающий по протоколу. Противник имеет доступ к базе данных на облаке, к каждой ее копии, к каналам связи на облаке, может создавать фальшивых клиентов, работающих по протоколу. То есть противник не только пассивно наблюдает, но и сам может производить запросы с помощью созданных фальшивых клиентов, т. е. производит атаку с известными открытыми запросами. Противник не имеет доступа к дилеру.

6.3. Основные результаты

Под коммуникационной сложностью для предложенной схемы будем понимать общее количество пересылаемых битов, необходимых для обмена информацией между дилером и облаком. То есть сумму числа битов, отправляемых дилером на облако и числа битов, получаемых от облака дилером, необходимых для нахождения дилером значения бита, запрашиваемого у дилера. Пусть s – число битов для представления номера бита. Напомним, что l^* является мощностью множества Set_i.

Утверждение 1. Коммуникационная сложность схемы получения значения номера бита дилером без раскрытия его номера равна $l^*(s+1)$ битов.

Доказательство. Для запроса значения бита дилер посылает копиям базы данных l^* номеров битов длиной s каждый. Облако отвечает l^* битами ■.

Проанализируем вероятность угадывания противником номера запрашиваемого бита.

Если запросы выполнены к одному множеству, то они неразличимы. Таким образом, если противник совершает n или более запросов, он не узнает номер конкретного бита. Максимум, какую информацию противник может получить – такое подмножество множества Set_i, что при запросе к каждому элементу из подмножества, на облаке осуществляется доступ ко всем битам из множества Set_i и только к ним. Что будет означать, что противник не знает, а только угадывает, какой именно бит из Set_i был выбран.

Утверждение 2. При однократной атаке с известным открытым запросом номера бита i и предположении о наличии пассивного противника на облаке вероятность угадывания противником номера бита не более $\frac{1}{l^*}$.

Доказательство. Множество Set_i состоит из l^* элементов и для любого $jSet_i$ выполняется $Set_i = Set_j$. То есть существует одинаковое множество для l^* различных номеров битов. Таким образом вероятность угадывания номера из множества Set_i равна $\frac{1}{l^*}$, так как Set_i имеет мощность l^* ■.

Сделав n или более запросов, противник получит информацию о числе реальных битов в каждом множестве Set_i. Число этих битов будет меньше или равно l^*. Таким образом, противник понимает, что запрошенный клиентом номер находится среди истинных номеров битов. И вероятность того, что клиент запрашивал конкретный номер реального бита для всех реальных номеров битов из одинаковая.

Для того чтобы оценить вероятность угадывания конкретного номера бита из множества Set_i, необходимо посчитать среднее число истинных битов, попадающих в множество Set_i.

Мощность множества X^c равна n_c, где элементам множества X соответствует n ($n < n_c$) элементов. Из множества X^c выбирается l^* элементов. В этом случае возникает вопрос: сколько элементов N из случайной выборки мощности l^* в среднем соответствует элементам множества X?

Пусть P_{r_*} – вероятность получения в выборке мощности l^* ровно r элементов, соответствующих элементам множества X. Тогда P_r находится по формуле

$$P_r = \frac{C_n^r C_{n_c-n}^{l^*-r}}{C_{n_c}^{l^*}}.$$

Тогда среднее число истинных битов, попадающих в множество Set_i,

$$N = \sum_{r=1}^{l^*} rP_r.$$

Утверждение 3. При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником в среднем не более $\frac{1}{N}$.

Доказательство. Выполнив n_c запросов, противник определяет элементы, входящие в каждое множество Set_i. Если фиктивные клиенты будут выполнять любое количество запросов большее n_c, противник все равно не получит никакой дополнительной информации.

Количество реальных бит, которые попали в каждую выборку мощности l^* в среднем равно N.

Тогда вероятность угадывания номера i в среднем будет не более $\frac{1}{N}$ ■.

Утверждение 4. При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$.

Доказательство. По построению гиперкубы, соответствующие интервалам, заполняются таким образом, что добавленные фиктивные номера битов добавляются только в интервал с номером l^*. Предположим, что $l^* - 1$ гиперкубов, соответствующих интервалам заполнены номерами битов, а интервал с номером l^* заполнен только фиктивными битами.

В этом случае противник, выполнив любое количество запросов, определит номер фиктивного бита из интервала с номером l^*, поскольку из каждого интервала выбирается только один бит для построения множества Set_i. Следовательно, при любом количестве запросов вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$ ■.

Значение k по сравнению с n мало. Докажем, что предложенный протокол удовлетворяет условию лаконичности.

Как было сказано ранее, для того чтобы коммуникационная сложность не возрастала, нужно подобрать значения d_u и l_u так, чтобы минимизировать разность $L_p - L_p^-$. Это следует из того, что $l = \frac{n}{L_p}$ и $l^* = \left\lceil \frac{n}{L_p^-} \right\rceil$, тогда $l^* \approx l$, следовательно $l^* \ll n$.

Теорема 1. Для предложенной схемы организации базы данных размера n:

• Коммуникационная сложность запроса равна $l^*(s+1)$ битов.

• При однократной атаке с известным открытым запросом номера бита i и предположении о наличии пассивного противника на облаке вероятность угадывания противником номера бита не более $\frac{1}{l^*}$.

• При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником в среднем не более $\frac{1}{N}$.

• При атаке с неограниченным числом известных открытых запросов и предположении о наличии пассивного противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$.

Доказательство. Из Утверждения 1 следует, что коммуникационная сложность запроса равна $l^*(s+1)$ битов.

Из Утверждения 2 следует, что при однократном запросе номера бита вероятность угадывания номера равна $\frac{1}{l^*}$.

Из Утверждения 3 следует, что при любом числе запросов номеров битов фиктивными клиентами и наличии противника на облаке вероятность угадывания номера бита в среднем не более $\frac{1}{N}$.

Из Утверждения 4 следует, что при любом числе запросов номеров битов фиктивными клиентами и наличии противника на облаке вероятность угадывания номера бита противником будет не более $\frac{1}{l^* - 1}$ ■.

7. ОЦЕНКА ГЕНЕРИРУЕМОЙ ДИЛЕРОМ ИНФОРМАЦИИ

Напомним, что $n = l^d$, $l = \sqrt[d]{n}$. По приведенным выше построениям d – размер гиперкуба, а l – длина его стороны. Для минимизации вычислений дилером необходимо минимизировать размер вычисляемых матриц M_β и M_σ. Для этого надо минимизировать размер строки каждой из матриц, т. е. величины $d_c l_c \left(l_c = \sqrt[d_c]{n_c}\right)$ и $d_u l_u \left(l_u = \sqrt[d_u]{n_u}\right)$.

Рассмотрим функцию $f(x) = \sqrt[x]{n} \cdot x$ описывающую соотношение между длиной стороны гиперкуба и его размерностью. Исследуем функцию $f(x)$ для того, чтобы минимизировать размер генерируемой дилером информации. Заметим, что $x \geq 1$.

По известной формуле $\left(a^{\frac{1}{x}}\right)' = -\frac{a^{\frac{1}{x}}}{x^2} \ln a$ найдем производную функции $f(x)$:

$$f'(x) = \left(n^{\frac{1}{x}} x\right)' = \left(n^{\frac{1}{x}}\right)' x + n^{\frac{1}{x}} x' =$$

$$= n^{\frac{1}{x}} \ln n \left(-\frac{1}{x^2}\right) x + n^{\frac{1}{x}} = n^{\frac{1}{x}} \left(1 - \frac{\ln n}{x}\right).$$

Производная равна нулю в точках экстремума. Найдем координату x точек экстремума. Поскольку $n^{\frac{1}{x}} > 0$, приравняем к нулю второй сомножитель:

$$1 - \frac{\ln n}{x} = 0.$$

Тогда

$$\frac{\ln n}{x} = 1,$$

$$x = \ln n.$$

Теперь выясним, эта точка экстремума минимум или максимум:

$$f''(x) = \left(n^{\frac{1}{x}} \left(1 - \frac{\ln n}{x} \right) \right)' =$$

$$= \left(n^{\frac{1}{x}} \right) \left(1 - \frac{\ln n}{x} \right) + n^{\frac{1}{x}} \left(1 - \frac{\ln n}{x} \right)' =$$

$$= -n^{\frac{1}{x}} \frac{\ln n}{x^2} \left(1 - \frac{\ln n}{x} \right) + n^{\frac{1}{x}} \frac{\ln n}{x^2} = n^{\frac{1}{x}} \frac{\ln^2 n}{x^3} > 0.$$

Таким образом, минимум функции $f(x) = \sqrt[x]{n} \cdot x$ находится в точке $x = \ln n$. Следовательно, минимальное количество элементов матриц M_β и M_σ, которое должен генерировать дилер, достигается при $x = \ln n$. Величина x должна быть целым числом, поскольку число разрядов и основание системы счисления должны быть целыми числами.

Выбор $\lceil x \rceil$ или $\lfloor x \rfloor$ влияет на число интервалов $l^* = \left\lceil \frac{n}{L_p^-} \right\rceil \left(L_p^- = l_u^{d_u} \right)$. Если интервал содержит большее число элементов, то l^* уменьшается, в противном случае l^* увеличивается. Величина l^* влияет на коммуникационную сложность.

8. СРАВНЕНИЕ С РАНЕЕ ПРЕДЛОЖЕННЫМИ АЛГОРИТМАМИ

Коммуникационная сложность предложенного алгоритма равна $l^*(s+1)$.

На практике для рассматриваемой базы данных размера 2^{40} число бит для представления номера s – не более 40 битов $\le 2^6$ (биты нумеруются с нуля).

В [2] коммуникационная сложность равна $\hat{l}\,d!\,Len(K, K_{enc})$, где $Len(K, K_{enc})$ – сумма длин шифротекстов. На практике $d \le 4$.

По построению интервалов:

$l^* \le 2^{10}$;

$\hat{l} \le 2^{10}$.

Для широко используемых алгоритмов вероятностного шифрования (AES, RSA) $Len(K, K_{enc})$ не менее $128 + 128 = 256$ бит (2^8).

Сравним l^* и $\hat{l} \cdot d!\cdot$, $l^* \le 2^{10}$, $\hat{l}\,d! \le 2^{10} \cdot 2^5$.

На практике:

• коммуникационная сложность предложенного алгоритма: $l^*(s+1) = 2^{10} \cdot 2^6$;

• коммуникационная сложность алгоритма из [2]: $2^{10} \cdot 2^4 \cdot 2^8 < \hat{l}\,d!\,Len(K, K_{enc}) < 2^{10} \cdot 2^5 \cdot 2^8$.

Напомним, что вероятность угадывания при однократном запросе с использованием перестановок:

$$\frac{1}{l^*} = \frac{1}{2^{10}}.$$

С использованием шифрования при $d = 4$:

$$\frac{1}{\hat{l}d!} \ge \frac{1}{2^{10} \cdot 2^5}.$$

Увеличим l^* (пусть $l^* = 2^{15}$), чтобы вероятность угадывания при однократном запросе была одинаковая в ранее рассмотренном и предложенном алгоритме. В этом случае коммуникационная сложность предложенного алгоритма равна:

$$l^*(s+1) = 2^{15} \cdot 2^6.$$

Таким образом, при заданных параметрах на практике при аналогичной вероятности угадывания коммуникационная сложность предложенного алгоритма по крайней мере в 2 раза лучше.

Оценим объем памяти, которую необходимо генерировать дилеру на практике. Поскольку минимум функции $f(x) = \sqrt[x]{n} \cdot x$ находится в точке $x = \ln n$, определим целочисленное значение x для $n = 2^{40}$. Если выбрать $x = 28$ то основание системы счисления $\sqrt[28]{n} \approx 3$, если выбрать $x = 27$, то основание системы счисления $\sqrt[27]{n} \approx 3$. В обоих случаях перестановка выполняется для последовательности чисел из 3 элементов и от выбора целой части x не зависит. Но размерность гиперкуба зависит от выбора целой части x. Если размерность гиперкуба больше, то l^* будет меньше, если размерность гиперкуба меньше, то l^* будет больше. Как было показано выше, если l^* увеличивается, то уменьшается вероятность угадывания, но увеличивается коммуникационная сложность и наоборот. Выбор целой части x зависит от цели: уменьшить

вероятность угадывания или уменьшить коммуникационную сложность.

Таким образом, для рассматриваемых параметров число перестановок будет 28 и длина перестановки равна 3. Для $n = 2^{40}$ нужно генерировать $28 \cdot 3 = 84$ элемента для вычисления нового номера при запросе к облаку.

При генерации множества Set_i после получения нового номера в интервале память дилера освобождается.

Генерацию перестановок как для всей базы данных и интервалов можно выполнять в разных потоках. Для каждого из l^* интервалов перестановки цифр числа при формировании множества Set_i могут выполняться параллельно, что увеличивает скорость работы алгоритма.

9. ЗАКЛЮЧЕНИЕ

Предложенный алгоритм обладает значительными преимуществами по сравнению с алгоритмами, описанными в [1, 2].

Алгоритм обладает следующими преимуществами:

• Вместо шифрования номера бита используются перестановки цифр в выбранной системе счисления. Шифрование и расшифрование номера бита на практике требует большего числа операций, чем перестановка разрядов в выбранной системе счисления.

• Так как при загрузке БД на облако выполняется перестановка, то на практике запрос рядом стоящих битов чаще всего выполняется для битов, которые не являются соседними на облаке. Это затрудняет противнику поиск стоящих рядом запрашиваемых битов.

• Все сдвиги в интервалах разные для любого множества Set_i. Это не позволяет сделать вывод о Set_j по Set_i. Ранее, в статье [2], использовался вектор $\boldsymbol{b}$, который содержит величину сдвига в интервале для каждого элемента любого множества Set_i. Если противник угадает вектор $\boldsymbol{b}$, это позволит ему сделать вывод о Set_j по Set_i.

Заметим также, что на практике при реализации вычислений легко организовать параллельные процессы, что позволяет увеличить скорость выполнения алгоритма.

10. БЛАГОДАРНОСТИ

Авторы выражают особую благодарность Лазареву Д.О. за ценные замечания в процессе работы над статьей.

СПИСОК ЛИТЕРАТУРЫ

1. *Мартишин С.А., Храпченко М.В., Шокуров А.В.* Организация безопасного запроса к базе данных на облаке // Труды Института системного программирования РАН. 2022. Т. 34. № 3. С. 173–188. ISSN 2079-8156 (Print), ISSN 2220-6426 (Online).
2. *Варновский Н.П., Мартишин С.А., Храпченко М.В., Шокуров А.В.* Организация конфиденциальных запросов к облаку // Труды Института системного программирования РАН. 2023. Т. 35. № 5. С. 37–54. ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).
3. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, in IEEE Annual Symposium on Foundations of Computer Science, 1995. P. 41–50.
4. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, Journal of the ACM, November 1998. V. 45. № 6. P. 965–982.

CLOUD DATA PLACING AND PRIVATE INFORMATION RETRIEVAL ALGORITHMS

© 2024 N. P. Varnovskiy[a], S. A. Martishin[b], M. V. Khrapchenko[b], A. V. Shokurov[b]

[a] *Information Security Section of Information Security Institute of Moscow State Lomonosov University*
Office 10, 1, Michurinskiy prospect, 119192, Moscow, Russia
[b] *Ivannikov Institute for System Programming of the RAS*
25, Alexander Solzhenitsyn st. 109004, Moscow, Russia

The authors consider the problem of ensuring secure queries to the database PIR (Private Information Retrieval) problem. Previously, the authors considered the problem for a database hosted in the cloud in the presence of an active adversary who does not interfere with the execution of the protocol, but can carry out an attack with known open queries. In algorithms, bit number i is represented as the l-ary number with a number of digits d. An algorithm for placing a database in the cloud and an algorithm for querying the required bit using permutations in the digits of the bit number, using the specification of the bit number i in the base l numerical system, were proposed. Permutations are treated as secret encryption keys. Communication complexity and probability of guessing the bit number for a one-time attack with a known open request for bit number i and for an attack with unlimited number of known open requests were estimated.

Keywords: database, cloud computing, PIR

REFERENCES

1. *Martishin S.A.., Khrapchenko M.V., Shokurov A.V.* Organization of a secure query to a database in the cloud. Trudy ISP RAN/Proc. ISP RAS. Vol. 34. Issue 3, 2022. P. 173–188 (in Russian). ISSN 2079-8156 (Print), ISSN 2220-6426 (Online).
2. *Varnovskiy N.P., Martishin S.A., Khrapchenko M.V., Shokurov A.V.* About cloud request protection. Trudy ISP RAN/Proc. ISP RAS. Vol. 35. Issue 5, 2023. P. 37–54 (in Russian). ISSN 2220-6426 (Online), ISSN 2079-8156 (Print).
3. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, in IEEE Annual Symposium on Foundations of Computer Science, 1995. P. 41–50.
4. *Chor B., Goldreich O., Kushilevitz E., Sudan M.* Private Information Retrieval, Journal of the ACM. Vol. 45. № 6. November 1998. P. 965–982.