Прикладная гомоморфная криптография: примеры

Г. Г. АРАКЕЛОВ, А. В. ГРИБОВ, А. В. МИХАЛЁВ
Московский государственный университет им. М. В. Ломоносова
e-mail: g.g.arakelov@gmail.com

УДК 004.056.55

Ключевые слова: гомоморфная криптография, матричные полиномы, криптографические вычисления, функциональное шифрование, шифрование с возможностью поиска.

Аннотация

Рассматриваются прикладные аспекты гомоморфной криптографии. Приводится описание полностью гомоморфной схемы шифрования на основе матричных полиномов. Приводятся результаты практического сравнения полностью гомоморфных схем шифрования. Рассматриваются частные случаи гомоморфного шифрования, допускающие вычисления ограниченного набора функций.

Abstract

G. G. Arakelov, A. V. Gribov, and A. V. Mikhalev, Applied homomorphic cryptography: examples, Fundamentalnaya i prikladnaya matematika, vol. 21 (2016), no. 3, pp. 25—38.

The paper is devoted to the application aspects of homomorphic cryptography. It provides a description of a fully homomorphic matrix polynomial-based encryption scheme. It also gives the results of practical comparison of fully homomorphic encryption schemes. We consider some special cases of homomorphic encryption allowing computations of a limited number of functions.

Введение

Одна из наиболее интересных и важных задач, стоящих перед современной криптографией, — это проведение вычислений над зашифрованными данными без их предварительной расшифровки. Вопрос о принципиальной возможности таких вычислений долгое время оставался открытым, и надо сказать, что авторы схемы шифрования RSA полагали, что такие вычисления в принципе невозможны. Раздел криптографии, посвящённый схемам, допускающим вычисления над шифротекстами, принято называть гомоморфной криптографией, а соответствующие схемы — гомоморфными. При этом различают полностью гомоморфные и частично гомоморфные схемы шифрования. В полностью гомоморфной схеме

Фундаментальная и прикладная математика, 2016, том 21, № 3, с. 25—38.

шифрования операции сложения и умножения шифротекстов являются гомоморфными. Более точно, выполняются следующие соотношения:

$$D\big(E(m_1) \cdot E(m_2)\big) = m_1 \cdot m_2, \tag{1}$$

$$D\big(E(m_1) + E(m_2)\big) = m_1 + m_2, \tag{2}$$

где $E(\cdot)$ — функция шифрования, а $D(\cdot)$ — функция дешифрования.

Если в некоторой схеме шифрования выполняется хотя бы одно из двух условий, то такая схема называется частично гомоморфной. Примеров частично гомоморфных шифровальных схем достаточно много. Например, сама схема RSA является гомоморфной относительно операции умножения, так же как и схема Эль-Гамаля. Схема RSA и схема Эль-Гамаля, к примеру, являются частично гомоморфными, а именно гомоморфными относительно операции умножения шифротекстов.

Первую модель полностью гомоморфной системы шифрования предложил К. Джантри в 2009 году. Работа К. Джантри доказала принципиальную возможность построения полностью гомоморфных систем шифрования. Предложенная К. Джантри модель требует существенных вычислительных затрат для практической реализации. И несмотря на то что после выхода работы К. Джантри появилось достаточное количество работ, в которых предлагались различные улучшения модели Джантри, до сих пор не существует полностью гомоморфной системы шифрования, которая может быть широко использована на практике. Кроме полностью гомоморфных систем шифрования, появились схемы, допускающие гомоморфные вычисления в некоторых частных случаях.

Относительно новым направлением в криптографии является построение криптографических схем на основе неассоциативных структур. Схема шифрования, сохраняющая гомоморфизм, предложена в [3], где построение криптографической системы ведётся на основе квазигруппового кольца. В [4] авторы используют неассоциативные группоиды для реализации процедуры открытого распределения ключей.

В данной статье рассматриваются некоторые модели гомоморфных схем шифрования, полезные с практической точки зрения. Одной из интересных и практически ценных схем шифрования является предложенная в [1] схема, основанная на матричных полиномах. Преимущество этой модели в том, что все вычисления, проводимые над зашифрованными данными, сводятся к сложению и перемножению матриц, а эти операции, как известно, допускают широкое распараллеливание. Возможность распараллеливания вычислений повышает практическую значимость схемы шифрования.

В первой части статьи рассматривается криптографическая схема, основанная на матричных полиномах. Приводятся результаты экспериментов по сравнению данной схемы с улучшенной версией криптосистемы Джантри, описанной в [14]. Во второй и третьей частях статьи рассматриваются частные случаи гомоморфного шифрования, допускающие вычисление ограниченного ряда функ-

ций. В четвёртой части приводится обзор модели защищённой реляционной базы данных, основанной на полностью гомоморфной схеме шифрования.

1. Криптосистема на основе матричных полиномов

1.1. Основные понятия и определения

Интересная модель полностью гомоморфного шифрования предложена в [1]. В основе модели лежит кольцо матричных полиномов. Открытыми текстами являются элементы поля вычетов $\mathbb{Z}_p$. Хотя по асимптотическим оценкам вычислительной сложности она незначительно уступает модели, предложенной в [4], проведённый анализ показывает её практическую ценность.

Напомним основные определения, которые будут использоваться.

Матричным полиномом над кольцом квадратных матриц $\mathbb{Z}_p^{N\times N}$ называется последовательность матриц

$$F = \{A_0, A_1, A_2, \ldots\}, \quad A_i \in \mathbb{Z}_p^{N\times N},$$

такая что среди A_i лишь конечное число ненулевых матриц. Пусть

$$F = \{F_1, F_2, \ldots\}, \quad G = \{G_1, G_2, \ldots\} -$$

матричные полиномы над кольцом $\mathbb{Z}_p$. Операции сложения и умножения матричных полиномов определяются следующим образом:

$$F + G = \{F_0 + G_0, F_1 + G_1, \ldots\},$$
$$F \cdot G = \{F_0 \cdot G_0, F_0 \cdot G_1 + F_1 \cdot G_0, F_0 \cdot G_2 + F_1 \cdot G_1 + F_2 \cdot G_0 \ldots\},$$
$$\text{т. е. } [F \cdot G]_k = \sum_{i+j=k} F_i \cdot G_j.$$

Множество матричных полиномов над кольцом $\mathbb{Z}_p^{N\times N}$ с операциями сложения и умножения само является кольцом. Будем обозначать это кольцо $\mathbb{Z}_p^{N\times N}[X]$.

Приведённым матричным полиномом называется матричный полином, старший коэффициент которого — единичная матрица.

Пусть

$$F = A_n X^n + A_{n-1} X^{n-1} + \ldots + F_0 -$$

некоторый матричный полином. Можно задаться поиском такой матрицы K, $K \in \mathbb{Z}_{\text{ı}}^{N\times N}$, что после подстановки матрицы K вместо X мы получим

$$F(K) = F_n \cdot K^n + F^{n-1} \cdot K^{n-1} + \ldots + F_0 = 0,$$

где под 0 подразумевается нулевая матрица.

Отметим, что уравнение $F(X) = 0$ может иметь корней больше, чем его степень, но может и вовсе не иметь корней.

Для любого матричного полинома $C(X)$ и произвольного приведённого матричного полинома $K(X)$, таких что $\deg\big(C(X)\big) > \deg\big(K(X)\big)$ существует и

единственно представление в виде $C(X) = K(X)Q(X)+R(X)$, где $\deg\big(R(X)\big) < \deg\big(K(X)\big)$ (см., например, [2]). Для полноты изложения приведём алгоритм деления на приведённый матричный полином.

Алгоритм деления на приведённый матричный полином

Пусть требуется разделить произвольный матричный полином $C(X) \in \mathbb{Z}_p^{N\times N}$ на приведённый матричный полином $K(X) \in \mathbb{Z}_p^{N\times N}$. В этом случае необходимо выполнить следующие действия.

1. Присвоить $Q(X)$ значение 0, т. е. все коэффициенты полинома равны нулю: $Q(X) := 0$.
2. Домножить $K(X)$ на $X^{\deg(C(X))-\deg(K(X))}$ и на некоторую матрицу $A \in \mathbb{Z}_p^{N\times N}$ так, чтобы старшие коэффициенты полиномов $C(X)$ и $A \cdot K(X) \cdot X^{\deg(C(X))-\deg(K(X))}$ стали равными. Присвоить $Q(X)$ новое значение:
$$Q(X) := Q(X) + A \cdot X^{\deg(C(X))-\deg(K(X))}.$$
3. Присвоить $C(X)$ новое значение:
$$C(X) := C(X) - A \cdot K(X) \cdot X^{\deg(C(X))-\deg(K(X))}.$$
4. Если $\deg\big(K(X)\big) > \deg\big(C(X)\big)$, то возвращаем в качестве результата полином $Q(X) \cdot K(X) + C(X)$, иначе переходим к шагу 2.

Ввиду того что матричный полином $K(X)$ является приведённым, т. е. его старший коэффициент — единичная матрица, шаг 2 будет выполним всегда, и притом однозначно, до тех пор пока $\deg\big(K(X)\big) < \deg\big(C(X)\big)$.

1.2. Построение криптосистемы

Через $\lambda \in \mathbb{N}$ обозначим параметр, задающий уровень криптостойкости. Пространством открытых текстов является $\mathbb{Z}_p$, где p — простое число. Пространство шифротекстов — это $\mathbb{Z}_p^{N\times N}[X]$, где $N = O(\lambda)$, т. е. шифротекстами являются матричные полиномы. Пространство секретных ключей — это $\mathbb{Z}_p^{N\times N}[X] \times \mathbb{Z}_p^N$, т. е. секретный ключ — это пара $(K(X), k)$, где k — N-мерный вектор целых чисел по модулю p. В данной схеме, помимо секретного ключа, используется ключ пере шифрования Это некоторой матричный полином $\mathrm{rk} \in \mathbb{Z}_p^{N\times N}[X]$, который передаётся на сторону вычислений для сокращения размеров шифротекстов. Теперь можно перейти к описанию алгоритмов схемы шифрования.

1.2.1. Генерация секретного ключа

1. Генерируется приведённый полином $K(X) \in \mathbb{Z}_p^{N\times N}[X]$, не имеющий корней, такой что $\deg\big(K(X)\big) = O(\lambda)$.

2. Генерируется вектор $k \in \mathbb{Z}_p^N$. Так как $\mathbb{Z}_p$, где p — простое число, является полем, то каждая координата вектора k обратима.
3. Пара $(K(X), k)$ сохраняется в качестве секретного ключа.

1.2.2. Генерация ключа перешифрования

Ключ перешифрования используется для предотвращения роста размера шифротекстов. После перемножения шифротекстов результат приводится по модулю ключа перешифрования.

1. Генерируется приведённый матричный полином $R'(X) \in \mathbb{Z}_p^{N\times N}[X]$, такой что $\deg\big(R'(X)\big) = O(\lambda)$.
2. Полином $\mathrm{rk}(X) = R'(X) \cdot K(X)$ сохраняется в качестве ключа перешифрования.

1.2.3. Шифрование

1. Открытому тексту $m \in \mathbb{Z}_p$ ставится в соответствие случайная матрица $M \in \mathbb{Z}_p^{N\times N}$, такая что $M \cdot \vec{k} = m \cdot \vec{k}$ и $M \cdot K(X) = K(X) \cdot M$, т. е. матрица M имеет собственный вектор $\vec{k}$ при собственном значении m и коммутирует с матричным полиномом $K(X)$.
2. Генерируется матричный полином $R(X) \in \mathbb{Z}_p^{N\times N}[X]$, для которого $\deg\big(R(X)\big) = O(\lambda)$ и $\deg\big(R(X)\big) < \deg\big(R'(X)\big)$.
3. Вычисляется шифротекст $C(X) = R(X)K(X) + M$.

1.2.4. Дешифрование

1. По шифротексту $C(X)$ вычисляется матрица $M = C(X) \mod K(X)$.
2. Выбирается любая обратимая координата вектора $\vec{k}$, пусть это будет k_i, и вычисляется $m = \Big(k_i^{-1}\big(M \cdot \vec{k}\big)\Big)_i$.

Мы не будем здесь приводить доказательство того, что данная схема является корректной. Это доказательство в развернутом виде приведёно в [1]. Нам интересны гомоморфные свойства описанной криптосистемы. Убедимся в том, что мы действительно имеем дело с полным гомоморфизмом, а именно что гомоморфизм выполняется в отношении сложения и умножения шифротекстов.

1.3. Аддитивный и мультипликативный гомоморфизм

Лемма 1. *Шифрование на основе матричных полиномов является полностью гомоморфным, т. е. выполняются соотношения*

$$D\big(E(m_1) + E(m_2)\big) = m_1 + m_2, \tag{3}$$

$$D\big(E(m_1) \cdot E(m_2)\big) = m_1 \cdot m_2, \tag{4}$$

где m_1, m_2 — открытые тексты, $E(\cdot)$ — функция шифрования, $D(\cdot)$ — функция дешифрования.

Доказательство. Покажем, что имеет место аддитивный и мультипликативный гомоморфизмы. Пусть

$$C_1(X) = R_1(X) \cdot K(X) + M_1, \quad C_2(X) = R_2(X) \cdot K(X) + M_2 -$$

два шифротекста, которые шифруют тексты m_1 и m_2 соответственно.

Рассмотрим сумму шифротекстов:

$$C_1(X) + C_2(X) = R_1(X) \cdot K(X) + M_1 + R_2(X) \cdot K(X) + M_2. \tag{5}$$

Из (5) получаем, что

$$C_1(X) + C_2(X) = \big(R_1(X) + R_2(X)\big) \cdot K(X) + (M_1 + M_2). \tag{6}$$

Посмотрим, что получится после применения алгоритма расшифровки к правой части равенства (6). Произведя деление суммы шифротекстов $C_1(X) + C_2(X)$ на $K(X)$, в остатке получаем $M_1 + M_2$. После умножения на вектор $\vec{k}$ получаем

$$(M_1 + M_2) \cdot \vec{k} = M_1 \cdot \vec{k} + M_2 \cdot \vec{k} = m_1\vec{k} + m_2 \cdot \vec{k}. \tag{7}$$

Затем, умножая вектор, полученный в (7), на k_i^{-1}, где i выбирается произвольно, получаем

$$\big(m_1\vec{k} + m_2 \cdot \vec{k}\big) \cdot k_i^{-1} = m_1\vec{k} \cdot k_i^{-1} + m_2 \cdot \vec{k} \cdot k_i^{-1}. \tag{8}$$

В результате в i-й позиции полученного в выражении (8) вектора будет стоять сумма $m_1 + m_2$, что и требовалось показать. Отсюда следует, что правая часть равенства (6) является шифротекстом для суммы открытых текстов $m_1 + m_2$ и после расшифровки даст значение $m_1 + m_2$, что доказывает аддитивность описываемой схемы шифрования.

Рассмотрим теперь произведение шифротекстов:

$$C_1(X) \cdot C_2(X) = (R_1(X) \cdot K(X) + M_1) \cdot (R_2(X) \cdot K(X) + M_2). \tag{9}$$

В равенстве (9) $R'(X)K(X)$ — это ключ перешифрования. Из равенства (9) получаем

$$R_1(X)\cdot K(X)\cdot R_2(X)\cdot K(X)+R_1(X)\cdot K(X)\cdot M_2+M_1\cdot R_2(X)\cdot K(X)+M_1\cdot M_2. \tag{10}$$

В правой части равенства (10) вынесем за скобки полином $K(X)$:

$$C_1(X) \cdot C_2(X) = \\ = (R_1(X) \cdot K(X) \cdot R_2(X) + R_1(X) \cdot M_2 + R_2(X) \cdot M_1) \cdot K(X) + M_1 \cdot M_2. \tag{11}$$

Посмотрим, что получится после расшифровки полинома из равенства (11). На первом шаге дешифрования после деления $C_1(X) \cdot C_2(X)$ на полином $K(X)$ получаем в остатке произведение двух матриц $M_1 \cdot M_2$. После применения следующего шага алгоритма дешифрования имеем

$$(M_1 \cdot M_2) \cdot \vec{k} = M_1 \cdot (M_2 \cdot \vec{k}) = M_1 \cdot \big(m_2\vec{k}\big) = m_2\big(M_1 \cdot \vec{k}\big) = m_1 \cdot m_2 \cdot \vec{k}. \tag{12}$$

Из (12) следует, что

$$D\big(E(m_1)\cdot E(m_2)\big) = m_1 \cdot m_2.$$

Мы получили, что сложение и умножение шифротекстов обладают свойством гомоморфизма, т. е. описанная схема шифрования является полностью гомоморфной. □

1.4. Вычислительные издержки

Наиболее затратная операция в данной схеме шифрования — это умножение шифротекстов, являющихся матричными полиномами, поэтому вычислительная сложность всей схемы будет напрямую зависеть от этой операции. В свою очередь, умножение матричных полиномов зависит от двух алгоритмов:

1) алгоритма умножения матриц;
2) алгоритма умножения полиномов.

Алгоритм умножения полиномов, пригодный для описанной схемы, имеет асимптотическую сложность $O(d^{\log_2 3}) = O(d^{1,5849\ldots})$ операций над коэффициентами полиномов, где d — наибольшая из степеней полиномов. Алгоритм умножения двух $(N \times N)$-матриц имеет асимптотическую сложность $O(N^{2,373\ldots})$ элементарных операций.

Если $N = O(\lambda)$ и степени полиномов шифротекстов равны $O(\lambda)$, получаем, что общее число операций над элементами $\mathbb{Z}_p$ имеет асимптотическую сложность $\approx O(\lambda^{3,76})$.

В настоящий момент наилучшая оценка на вычислительные издержки при гомоморфном вычислении составляет $g(\lambda) = O(\lambda^{3,5})$. Такой вычислительной сложностью обладает схема, описанная в [14].

Здесь стоит заметить, что при оценке вычислительной сложности схемы шифрования, основанной на матричных полиномах, считалось, что умножение матриц размерности $N \times N$ требует $O(N^{2,373\ldots})$ операций. Действительно, существует алгоритм, способный произвести умножение двух матриц за указанное время, и это алгоритм Копперсмита—Винограда, улучшенный Вильямсом. Однако на практике алгоритм Копперсмита—Винограда не может быть использован в настоящее время, так как он имеет очень большую константу пропорциональности и начинает выигрывать в быстродействии у других известных алгоритмов только для матриц, размер которых превышает память современных компьютеров.

С другой стороны, известная гипотеза Штрассена утверждает, что для сколь угодно малого $\varepsilon > 0$ существует алгоритм, при достаточно больших натуральных n гарантирующий перемножение двух матриц размера $n \times n$ за $O(n^{2+\varepsilon})$ операций.

Гипотеза Штрассена до сих пор остаётся одной из нерешённых задач линейной алгебры, и если считать, что она верна, то описанная здесь криптосистема, предложенная Ф. Б. Буртыкой, на сегодняшний день является самой «быстрой» в плане вычислений из известных схем полностью гомоморфного шифрования.

Криптосистема на матричных полиномах уступает по асимптотическим оценкам системе, описанной в [14], но с практической точки зрения в настоящий момент она представляет ценность, поскольку допускает широкое распараллеливание. В частности, был произведён эксперимент с использованием технологии для массовых параллельных вычислений CUDA фирмы Nvidia, который показал превосходство по времени вычислений криптосистемы на матричных полиномах.

1.5. Использование параллельных вычислений при реализации

Наилучшая оценка вычислительных издержек при гомоморфных вычислениях принадлежит криптосистеме, описанной в [14]. В компании IBM реализована так называемая библиотека гомоморфного шифрования, которая получила название HElib (https://github.com/shaih/HElib). Данная библиотека содержит реализацию криптосистемы, считающейся на сегодняшний день асимптотически лучшей в плане вычислительных издержек среди гомоморфных систем шифрования.

Для проведения сравнительного анализа описанной в статье криптосистемы и модифицированной криптосистемы Джантри из работы [14] был реализован макет криптосистемы на матричных полиномах, в котором для умножения матриц и полиномов использовалась технология параллельных вычислений CUDA, а модифицированная система Джантри была взята из библиотеки HElib.

Был проведён ряд экспериментов с различными параметрами криптостойкости. Оценивалось время требуемое для следующих операций:

1) шифрование;
2) дешифрование;
3) умножение.

Результаты приведены в таблицах 1 и 2. По приведённым оценкам производительности видно, что на практике криптосистема, основанная на матричных полиномах, не уступает усовершенствованной модели шифрования Джантри, а даже выигрывает при использовании технологий параллельных вычислений. Мы

Таблица 1. Оценка производительности криптосистемы на матричных полиномах с использованием параллельных вычислений

Параметр λ	Шифрование	Дешифрование	Умножение шифротекстов
16	4 мс	13 мс	8 мс
24	79 мс	13 мс	15 мс
32	1,5 с	14 мс	22 мс
64	2 мин	20 мс	1 с

Таблица 2. Оценка производительности модифицированной криптосистемы Джантри

Параметр λ	Шифрование	Дешифрование	Умножение шифротекстов
16	2 мс	6 мс	5 мс
24	40 мс	11 мс	12 мс
32	1 с	15 мс	50 мс
64	5 мин	200 мс	10 с

провели эксперимент для максимального значения параметра $\lambda = 64$ и уже при значении $\lambda = 32$ получили, что модель Джантри, считающаяся асимптотически самой «быстрой», уступает модели Буртыки.

2. Шифрование с возможностью поиска

Первыми были предложены схемы, которые позволяют производить поиск по ключевым словам. Наиболее оптимальными с точки зрения сложности и безопасности являются схемы Д. Сонг, Д. Вагнера, А. Перрига (2000 г.) и Р. Куртмолы, Х. Гарая, С. Камара, Р. Островского (2006 г.) [13]. Также можно выделить схему с возможностью поиска по булевым выражениям Д. Кэша, С. Ярецкого, Ч. Джатлы (2013) [8]. Одной из последних является схема Д. Кэша, Дж. Джейгера, С. Ярецкого, Х. Кравчика, М.-К. Розу, М. Штейнера (2014 г.) [9], она уже используется в некоторых приложениях на практике.

Рассмотрим основной принцип таких схем на примере криптосистемы с использованием симметричной системы шифрования. Алгоритм предполагает наличие двух частей: клиентской и серверной. Предположим, что $D = (D_1, \ldots, D_n)$ — это массив данных, например набор текстовых файлов, DB — это база данных, содержащая соответствие между всеми ключевыми словами w из D и соответствующими идентификаторами $\mathrm{DB}[w]$, т. е. всеми файлами, содержащими слово w. Тогда схема шифрования состоит из тройки алгоритмов $(\mathrm{Setup}, \mathrm{Token}, \mathrm{Search})$, где алгоритм $\mathrm{Setup}\colon (1^k, \mathrm{DB}) \to (K, \mathrm{EDB})$ вырабатывает секретный ключ K и шифрует базу данных DB. Алгоритм на клиентской части $\mathrm{Token}\colon (w, K) \to \mathrm{tk}_w$ вырабатывает специальный токен для заданного ключевого слова, и алгоритм $\mathrm{Search}\colon (\mathrm{EDB}, \mathrm{tk}_w) \to \mathrm{DB}[w]$ запускается на серверной части алгоритма и возвращает множество идентификаторов, содержащих ключевое слово w.

Алгоритм Search использует симметрическую систему шифрования $(\mathrm{Gen}, \mathrm{Enc}, \mathrm{Dec})$, псевдослучайную функцию $F\colon \{0,1\}^k \times W \to \{0,1\}^k$ и псевдослучайную перестановку $P\colon \{0,1\}^k \times W \to \{1, |W|\}$. Сначала вырабатываются

случайные ключи K_t и K_f для соответствующих преобразований и выделяется память под массивы T и RAM_1. Для каждого слова $w \in W$ и для всех $1 \leqslant i \leqslant |\mathrm{DB}[w]|$ формируется массив элементов

$$N_{w,i} = \langle \mathrm{id}_{w,i}, \mathrm{ptr}_1(w, i+1) \rangle,$$

где $\mathrm{id}_{w,i}$ — i-й идентификатор в $\mathrm{DB}[w]$ и $\mathrm{ptr}_1(w, i+1)$ — адрес в массиве RAM_1. После этого в массив RAM_2 на случайные позиции помещаются значения из RAM_1. Далее происходит шифрование каждого значения из RAM_2. Для этого создаётся новый массив RAM_3, такой что для всех $w \in W$ и всех $1 \leqslant i \leqslant |\mathrm{DB}[w]|$ выполняется

$$\mathrm{RAM}_3[\mathrm{addr}_2(N_{w,i})] = \mathrm{Enc}_{K_w}(\mathrm{RAM}_2[\mathrm{addr}_2 * (N_{w,i})]),$$

где $K_w = F_{K_f(w)}$ и addr_2 — функция, которая возвращает адрес аргумента в массиве RAM_2. Теперь для всех ключевых слов $w \in W$ создаётся массив

$$T[P_{K_T}(w)] = \mathrm{Enc}_{K_w}\big(\mathrm{addr}_3(N_{w,1})\big),$$

где addr_3 — функция, которая возвращает адрес аргумента в массиве RAM_3. Финальным шагом на клиентской части является создание массива $\mathrm{EDB} = (T, \mathrm{RAM}_3)$. Данный массив хранится на серверной части и используется для поиска ключевых слов по массиву данных.

Теперь для того чтобы найти ключевое слово w в массиве EDB, который хранится на серверной части, необходимо вычислить токен

$$\mathrm{tk} = (\mathrm{tk}_1, \mathrm{tk}_2) = \big(P_{K_T}(w), F_{K_f(w)}\big)$$

на клиентской части. Сервер вычисляет значение $c = T[\mathrm{tk}_1]$, расшифровывает его и получает адрес $a_1 = \mathrm{Dec}_{\mathrm{tk}_2}(c)$. Далее для всех i, пока $a_i = \perp$, сервер расшифровывает значения $(N_{w,1}, \ldots, N_{w,|\mathrm{DB}[w]|})$, вычисляя

$$(\mathrm{id}, a_{i+1}) \leftarrow \mathrm{Dec}_{K_F}(\mathrm{RAM}_3[a_i]).$$

Последним шагом является нахождение документов с идентификаторами $(\mathrm{id}_1, \ldots, \mathrm{id}_{|\mathrm{DB}[w]|})$.

Сложность поиска для данной схемы равна $O(|\mathrm{DB}[w]|)$, что позволяет использовать её для практических приложений. Знание $\mathrm{EDB} = (T, \mathrm{RAM}_3)$ позволяет злоумышленнику получить количество ключевых слов как значение T и $\sum\limits_{w \in W} |\mathrm{DB}[w]|$ как размер массива RAM_3. Однако злоумышленник не может получить никакой нетривиальной информации, например частотности ключевого слова.

3. Функциональное шифрование

Подобно гомоморфному, функциональное шифрование позволяет вычислять различные функции над зашифрованной информацией. Однако такой вид шифрования ограничивает список возможных функций и информацию, которая ста-

нет открыта при вычислении. Это становится возможным благодаря специальному секретному ключу, соответствующему каждой функции. Первой работой в этой области была статья [6]. Дальнейшее развитие связано с поиском практической реализации функционального шифрования для любой случайной функции. Один из подходов основан на использовании многосторонних вычислений [12].

Формальное определение такого рода схем было дано в [6]. Алгоритмы $(\mathrm{Setup}, \mathrm{Keygen}, \mathrm{Enc}, \mathrm{Dec})$ называются функциональной схемой шифрования, если они удовлетворяют следующим условиям:

- $(\mathrm{pk}, \mathrm{mk}) \leftarrow \mathrm{Setup}(1^k)$ — генерация пары открытого и секретного ключей;
- $\mathrm{sk} \leftarrow \mathrm{Keygen}(\mathrm{mk}, x)$ — генерация секретного ключа для k;
- $c \leftarrow \mathrm{Enc}(\mathrm{pk}, x)$ — шифрование сообщения x;
- $y = F(k, x) \leftarrow \mathrm{Dec}(\mathrm{sk}, c)$ — вычисление $F(k, x)$ из c при помощи ключа sk.

Примерами функциональных схем шифрования являются схемы, позволяющие осуществлять

- предикатное шифрование (для многих приложений открытым текстом является пара $(\mathrm{ind}, m) \in I \times M$, где ind — индекс сообщения m; например, индексом может быть имя получателя, m — сообщение в системе электронной почты);
- шифрование, основанное на идентификаторах (оно позволяет зашифровать сообщение без открытого ключа; примером использования такой схемы может служить шифрование электронного письма к получателю, который не имеет сформированного сертификата открытого ключа);
- атрибутное шифрование (шифротекст выполняет роль ключа для расшифровки).

Рассмотрим схему шифрования, основанного на идентификаторах, предложенную А. Сахаи и Б. Уотерсом. Пусть G_1 — билинейная группа порядка p с порождающим элементом g и $e\colon G_1 \times G_1 \to G_2$ — билинейное отображение. Идентификаторы принадлежат множеству U элементов из $\mathbb{Z}_p^*$.

Алгоритм Setup. Открытым параметром является набор элементов $T_1 = g^{t_1}, \ldots, T_{|U|}^{t_{|U|}}$, $Y = e(g, g)^y$, главным секретным ключом является набор $t_1, \ldots, l_{|U|}, y$.

Алгоритм Keygen. Для формирования секретного ключа для идентификатора $w \subset U$ генерируется многочлен q порядка $d - 1$, где d — параметр схемы, такой что $q(0) = y$. Тогда секретный ключ состоит из компонент $(D_i) = g^{q(i)/t_i}$ для всех $i \in w$.

Алгоритм Enc. Шифротекстом для сообщения $M \in G_2$ для значения w' является набор $E = (w', E' = MY^s, \{E_i = T_i^s\}_{i \in w'})$, где s — случайный элемент из $\mathbb{Z}_p$.

Алгоритм Dec. Для расшифрования используется коэффициент Лагранжа

$$\Delta_{i,S}(x) = \prod_{j \in S,\ j \neq i} \frac{x - j}{i - j},$$

выполняются следующие действия:

$$E'/\prod_{i\in S}(e(D_i,E_i)^{\Delta_{i,S}(0)} = Me(g,g)^{sy}/\prod_{i\in S}\left(e(g^{q(i)/t_i},g^{st_i})\right)^{\Delta_{i,S}(0)} =$$
$$= Me(g,g)^{sy}/\prod_{i\in S}\left(e(g,g^{\mathrm{sq}(i)})\right)^{\Delta_{i,S}(0)} = M.$$

Последнее равенство верно, так как многочлен $\mathrm{sq}(x)$ степени $d-1$ может быть найден по d значениям.

4. Гомоморфное шифрование облачных баз данных

Облачные технологии — современная и многогранная область науки и техники, бурно развивающаяся в последнее время. Одним из примеров использования облачных технологий является облачная база данных. Согласно законодательству многих стран, в том числе и законодательству Российской Федерации, не всегда можно использовать облачную базу данных для хранения информации в открытом виде. Полностью гомоморфное шифрование способно решить данную проблему за счёт того, что все данные, которые должны будут храниться в облачном хранилище, будут зашифрованы, более того, доступ к расшифрованным данным будет только у владельца хранилища. Также посторонние лица, имеющие доступ к данному облачному хранилищу, не смогут получить информацию о запросах, которые посылает владелец на облако, а также о результатах соответствующих запросов. Модель использования полностью гомоморфного шифрования для реализации надёжного облачного хранилища описывается далее.

Модель базы данных на основе гомоморфного шифрования

Рассмотрим случай реляционной базы данных, описанный в [1]. Реляционная база данных представляется набором прямоугольных страниц. Для простоты, без потери общности, можно считать что база данных состоит из одной таблицы. Таблица имеет атрибуты $a_1, a_2, \ldots, a_m$ и состоит из множества записей $\{R_i\}_{i=1}^n$, где $R_i = \{w_{i,j}\}_{j=1}^m$ — значение записи R_i в атрибуте a_j.

Рассмотрим случай, когда клиенту необходимо делать к базе запросы двух видов:

$$\texttt{SELECT * FROM db WHERE } (a_{t_1} = v_1) \texttt{ OR } (a_{t_2} = v_2) \texttt{ OR } \ldots \texttt{ OR } (a_{t_k} = v_k), \quad (13)$$

$$\texttt{SELECT * FROM db WHERE } (a_{t_1} = v_1) \texttt{ AND } (a_{t_2} = v_2) \texttt{ AND } \ldots \texttt{ AND } (a_{t_k} = v_k). \quad (14)$$

Запросы типа (13) будем называть дизъюнктивными, а запросы типа (14) — конъюнктивными. Теперь рассмотрим, как можно построить надёжное облачное хранилище данных, имея полностью гомоморфную схему шифрования.

Пусть S — некоторый облачный сервер, хранящий базу $\mathrm{db} = \{R_i\}_{i=1}^n$, принадлежащую клиенту K. Периодически клиент K обращается на сервер S с запросами. В результате он должен получить список записей, удовлетворяющих условию `WHERE` из запроса клиента, при этом необходимо, чтобы сервер S ничего не узнал о значениях v_i из условия `WHERE` запроса клиента, а также о записях в базе данных, которые удовлетворяют условию запроса.

Один из подходов к решению данной задачи заключается в следующем:

— на первом шаге клиент K получает номера записей $i_1, i_2, \ldots, i_t$, удовлетворяющих запросу. Этот шаг должен быть реализован так, чтобы S не узнал $i_1, i_2, \ldots, i_t$;

— клиент K по очереди извлекает из базы записи с номерами $i_1, i_2, \ldots, i_t$ так, чтобы S не узнал значения индексов.

Для того чтобы сервер S, а также сторонние пользователи не могли получить доступ к базе клиента K, она хранится в зашифрованном виде, т. е. на самом деле S хранит значение каждого атрибута для каждой записи в зашифрованном виде. Пусть для шифрования используется полностью гомоморфная схема шифрования E, а sk — секретный ключ.

Защищённое получение индексов

Клиент K хочет скрыть значения v_i, $1 \leqslant i \leqslant k$, поэтому ему необходимо их зашифровать. На сервер S клиент K передаёт пары $\big(a_{z_i}, E(v_i)\big)$, $1 \leqslant i \leqslant k$.

В свою очередь, S должен для каждой записи $R_i = \{E(w_{i,j})\}_{j=1}^m$, $i = 1, \ldots, n$, провести следующие вычисления:

1) для всех $z_j = 1, \ldots, k$ вычислить $e_k = f\big(E(w_{i,z_j}), E(v_k)\big)$, где f — функция, осуществляющая проверку на равенство w_{i,z_j} и v_j гомоморфно, т. е. $e_j = E(1)$ в случае равенства w_{i,z_j} и w_j и $e_k = E(0)$ в противном случае;
2) в зависимости от типа запроса выполнить следующее:
   — конъюнктивный запрос:
   $$e'_i = H_{\mathrm{AND}}(e_1, e_2, \ldots, e_t),$$
   где $H_{\mathrm{AND}}(e_1, e_2, \ldots, e_t) = e_1 \cdot e_2 \cdot e_3 \cdot \ldots \cdot e_t$;
   — дизъюнктивный запрос:
   $$e'_i = H_{\mathrm{XOR}}(e_1, e_2, \ldots, e_t),$$
   где $H_{\mathrm{XOR}}()$ — функция, вычисляющая XOR от аргументов;
3) сервер S отправляет клиенту K вектор $\mathrm{Res} = (e'_1, e'_2, \ldots, e'_n)$.

Для извлечения записей из базы данных по их индексам клиент K может воспользоваться стандартным протоколом секретного получения информации. На данный момент разработано большое количество различных протоколов секретного получения информации [7].

Литература

[1] Буртыка Ф. Б. Симметричное полностью гомоморфное шифрование с использованием неприводимых матричных полиномов // Изв. ЮФУ. Техн. науки. — 2014. — С. 107—122.

[2] Глухов М. М., Елизаров В. П., Нечаев А. А. Алгебра. — СПб.: Лань, 2015.

[3] Грибов А. В., Золотых П. А., Михалёв А. В. Построение алгебраической криптосистемы над квазигрупповым кольцом // Математические вопросы криптографии. — 2010. — Т. 1, № 4. — С. 23—32.

[4] Катышев С. Ю., Марков В. Т., Нечаев А. А. Использование неассоциативных группоидов для реализации процедуры открытого распределения ключей // Дискрет. матем. — 2014. — Т. 26, № 3. — С. 45—64.

[5] Кузьмин А. С., Марков В. Т., Михалёв А. А., Михалёв А. В., Нечаев А. А. Криптографические алгоритмы на группах и алгебрах // Фундамент. и прикл. матем. — 2015. — Т. 20, вып. 1. — С. 205—222.

[6] Boneh D., Sahai A., Waters B. Functional encryption: Definitions and challenges // Theory Cryptography. — 2011. — P. 253—273.

[7] Boneh D., Gentry C., Halevi S., Wang F., Wu D. J. Private database queries using somewhat homomorphic encryption // Applied Cryptography and Network Security: 11th Int. Conf., ACNS 2013, Banff, AB, Canada, June 25—28, 2013. Proc. / M. Jacobson, M. Locasto, P. Mohassel, and R. Safavi-Naini, eds. — Berlin: Springer, 2013. — (Lect. Notes Comp. Sci. Security Cryptology; Vol. 7954). — P. 102—118.

[8] Cash D., Jaeger J., Jarecki St., Jutla Ch., Krawczyk H., Rosu M.-C., Steiner M. Highly-scalable searchable symmetric encryption with support for Boolean queries // Advances in Cryptology — CRYPTO 2013 / R. Canetti, J. A. Garay, eds. — Berlin: Springer, 2013. — (Lect. Notes Comp. Sci. Security Cryptology; Vol. 8042). — P. 353—373.

[9] Cash D., Jaeger J., Jarecki St., Jutla Ch., Krawczyk H., Rosu M.-Cat., Steiner M. Dynamic Searchable Encryption in Very-Large Databases: Data Structures and Implementation: Cryptology ePrint Archive, Report 2014/853.

[10] Curtmola R., Garay J., Kamara S., Ostrovsky R. Searchable symmetric encryption: improved definitions and efficient constructions // Proc. 13th ACM Conf. Computer Communication Security. — New York: ACM, 2006.

[11] Gentry C. A Fully Homomorphic Encryption Scheme: Ph. D. thesis. — Stanford Univ., 2009.

[12] Gorbunov S., Vaikuntanathan V., Wee H. Functional encryption with bounded collusions via multi-party computation // Advances in Cryptology — CRYPTO 2012 / Safavi-Naini R., Canetti R., eds. — Berlin: Springer, 2012. — (Lect. Notes Comp. Sci.; Vol. 7417). — P. 162—179.

[13] Song D., Wagner D., Perrig A. Practical techniques for searches on encrypted data // SP '00 Proc. 2000 IEEE Symp. Security and Privacy. — Berkeley: Univ. California, 2000.

[14] Stehle D., Steinfeld R. Faster fully homomorphic encryption // Advances in Cryptology — ASIACRYPT 2010: 16th Int. Conf. on the Theory and Application of Cryptology and Information Security, Singapore, December 5—9, 2010. Proc. — Berlin: Springer, 2010. — (Lect. Notes Comp. Sci.; Vol. 6477). — P. 377—394.