*А. О. Жиров, О. В. Жирова, С. Ф. Кренделев*

# БЕЗОПАСНЫЕ ОБЛАЧНЫЕ ВЫЧИСЛЕНИЯ С ПОМОЩЬЮ ГОМОМОРФНОЙ КРИПТОГРАФИИ[1]

## Введение

В настоящее время одно из наиболее активных направлений развития информационных технологий — облачные вычисления. Основной причиной такого развития является возможность для компаний и частных лиц снижения расходов на поддержание собственной ИТ-инфраструктуры за счет передачи этой работы провайдеру облачного сервиса. Однако в такой ситуации становятся небезопасными хранение и обработка конфиденциальных данных в облачной инфраструктуре, так как у ее провайдера появляется возможность неконтролируемого доступа к обрабатываемым данным.

Единственным решением этой проблемы может служить шифрование всех приватных данных перед передачей в облако. К сожалению, все распространенные в настоящее время криптографические алгоритмы не позволяют производить произвольные вычисления над зашифрованными данными, существенно ограничивая возможности использования облачных ресурсов.

## 1. Полностью гомоморфное шифрование: PoC

В связи с необходимостью повышения безопасности облачных вычислений становится важным создание эффективных алгоритмов полностью гомоморфного шифрования, позволяющего производить произвольные вычисления над данными без предварительной расшифровки. Принципиальную возможность такого шифрования доказал исследователь из IBM Крейг Гентри в своей диссертации в 2009 г. [1]. Рассмотрим предложенную им схему на примере вычислений в $\mathbb{Z}_2$, что можно понимать как работу с битами. Выберем какое-нибудь нечетное число $\rho$, раз оно нечетное, то $\rho = 2k + 1$. Это число $\rho$ является секретным параметром. Пусть $m$ принимает значения $\{0,1\}$, тогда построим число $z \in \mathbb{Z}$ по правилу $z = 2r + m$, где $r$ произвольно. Этот выбор означает, что $z = m \bmod 2$. Шифрование заключается в том, что всякому $m$ сопоставляется число $c = z + \rho q$, где $q$ произвольно. Следовательно:

$$c = 2r + m + (2k + 1) \cdot q.$$

Это число $c$ отправляется на вычисления. Заметим, что $(c \mod 2) = (m + q) \mod 2$, а значит, злоумышленник может узнать только четность выхода из шифрования, но не более того.

Рассмотрим процесс расшифрования. Пусть нам известны числа $c, \rho$, где $c$ — зашифрованное число, а $\rho$ — известный нам секрет. Тогда предлагается выполнить следующие действия:

1. Произведем непосредственное расшифрование при помощи нашего секретного ключа $\rho$:

$$r = c \bmod p = (z + pq) \bmod p = z \bmod p + (pq) \bmod p .$$

Число $r = c \mod p$ называется шумом, его возможные значения лежат в интервале $(-\frac{p}{2}, \frac{p}{2})$.

2. Затем получим исходный зашифрованный бит:

$$m = r \bmod 2.$$

Убедимся, что данная операция является гомоморфным шифрованием. Пусть есть два числа (бита) $m_1, m_2 \in \mathbb{Z}_2$. Сопоставим им пару чисел:

[1] Работа выполнена при финансовой поддержке Минобрнауки РФ (договор № 02.G25.31.0054).

$z_1 = 2r_1 + m_1$
$z_2 = 2r_2 + m_2$

Выберем секретное число $p = 2k + 1$ и вычислим «зашифрованные» значения:

$c_1 = z_1 + p \cdot q_1$
$c_2 = z_2 + p \cdot q_2$

Тогда их сумма и произведение будут равны соответственно:

$$\begin{aligned} c_1 + c_2 = z_1 + z_2 + p(q_1 + q_2) = 2r_1 + m_1 + 2r_2 + m_2 + p(q_1 + q_2) = \\ = 2(r_1 + r_2) + m_1 + m_2 + (2k+1)\cdot(q_1 + q_2) \end{aligned} \quad (1)$$

$$\begin{aligned} c_1 c_2 = z_1 z_2 + p(z_1 p_2 + z_2 q_1) + p^2 q_1 q_2 = \\ = (2r_1 + m_1)(2r_2 + m_2) + 2k(z_1 q_2 + z_2 q_1) + z_1 q_2 + z_2 q_1 = \\ = 4r_1 r_2 + 2(r_1 m_2 + r_2 m_1) + m_1 m_2 + 2k(z_1 q_2 + z_2 q_1) + \\ + 2r_1 q_2 + 2r_2 q_1 + m_1 q_2 + m_2 q_1 \end{aligned} \quad (2)$$

Заметим, что применение процедуры расшифрования к результату уравнения (1) дает сумму исходных бит $m_1$ и $m_2$:

$$\begin{aligned} [(c_1 + c_2) \bmod p] \bmod 2 = \\ = [2(r_1 + r_2) + m_1 + m_2] \bmod 2 = m_2 + m_2 \end{aligned} . \quad (3)$$

В то же время, не зная числа $p$, расшифровать результат невозможно:

$$(c_1 + c_2) \bmod 2 = m_1 + m_2 + q_1 + q_2 . \quad (4)$$

Аналогичные выводы можно получить и для операции умножения, пользуясь уравнением (2).

Следовательно, результат расшифрования совпадает с ожидаемым, а также «взлом» шифра без знания числа $p$ по-прежнему невозможен. Таким образом, мы убедились, что схема Крейга Гентри действительно представляет собой полностью гомоморфное шифрование.

Тем не менее следует заметить, что настоящего гомоморфизма в математическом смысле здесь нет. Выполнение вычислений приводит к накоплению ошибки $r$, и после того, как она превышает $p$, правильно расшифровать сообщение становится невозможным. В то же время если бы существовал настоящий гомоморфизм, то данной проблемы не возникло бы по определению.

Подводя итог, следует заметить, что рассмотренная схема не реализуема на практике, поскольку в реальных вычислениях ошибка $r$ быстро достигает критического предела, а использование техники bootstrapping, предложенной Гентри для преодоления этой проблемы, приводит к очень быстрому росту объема шифротекста. Для практического использования такого шифрования придется применять для вычислений чрезвычайно сложные алгоритмы либо ограничивать количество операций, которые можно производить над данными без риска выйти за границу диапазона значений.

## 2. Обобщенная схема гомоморфного шифрования

Схему Гентри можно интерпретировать и иначе, опираясь на формулу:

$c \mod 2 = (m + q) \mod 2$.

Фактически секретное число $q$ задает, во что перейдет бит $m$ в результате шифрования. Поскольку мы рассматриваем числа из $\mathbb{Z}_2$, для произвольного $q$ возможно всего два варианта:

$$\begin{cases} 1 \to 0 \\ 0 \to 1 \end{cases} \qquad \begin{cases} 1 \to 1 \\ 0 \to 0 \end{cases} \quad (5)$$

Таким образом, выбирая $q$, мы задаем перестановку чисел, которая будет определять шифрование исходного бита $m$. Так как при шифровании мы имеем право выбирать разные $q$ для разных чисел, то для функций от двух битов вариантов перестановок может быть 4, для $n$ бит — $2^n$.

Следовательно, с точки зрения безопасности шифрования данная схема эквивалентна подходу, при котором на вычисления отправляются просто все возможные входные комбинации входных значений (получаемые всеми перестановками) и после получения всех возможных ответов выбирается нужный по номеру изначально выбранной перестановки. Рассмотрим данный подход на примере вычисления произвольной функции $f(m_1, m_2)$:

| № перестановки | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $m_1$ | 0 | 0 | 1 | 1 |
| $m_2$ | 0 | 1 | 0 | 1 |

На выходе мы получаем таблицу:

| № перестановки | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| $f(m_1, m_2)$ | f(0, 0) | f(0, 1) | f(1, 0) | f(1, 1) |

Однако с точки зрения реального использования данная идея имеет два существенных недостатка:

1. Необходимо передавать огромное количество данных;

2. Вычисление результата даже простых функций потребует огромного вычислительного ресурса.

Для решения этих проблем можно кодировать таблицу аргументов с помощью функций $m_1(k)$ и $m_2(k)$, где $k$ — номер перестановки, передавать их как функции и в качестве результата получать функцию $f(k)$. Номер перестановки по-прежнему считается секретным, и, имея результат в виде $f(k)$, мы можем вычислить её значение в точке $k$, получая конечный результат.

Обобщение этой методики формирует основу для алгоритмов, над которыми работает наша команда. Пусть нам требуется вычислить функцию $y_0 = f\ (x_1,\ ...,\ x_n)$. Для простоты будем считать, что в $f$ используются только операции $+$ и $\times$. Тогда схема алгоритма будет такова:

1. Сопоставляем набору чисел $x_1,\ ...,\ x_n$ функции $g_1(k),\ ...,\ g_n(k)$, такие, что $\forall i = 1 \ldots n$, $g_i(k_0) = x_i$, где $k_0$ будет нашим секретным ключом.

2. Отправляем на вычислитель (в облако) полученные на предыдущем шаге функции $g_1(k),\ ...,\ g_n(k)$.

3. В облаке вычисляется функция $y(k) = f(g_1(k),\ ...,\ g_n(k))$. Важным моментом является то, что в качестве аргументов используются не конкретные значения, а функции как математические объекты.

4. Результат вычислений отправляется нам.

5. Мы вычисляем функцию $y_0 = y(k_0)$, где $y_0$ является нашим желаемым результатом.

Чтобы данный подход работал, на класс функций, к которым принадлежат $g_i$, налагается ряд ограничений:

1. Эти функции должны кратко записываться, т. е. объем информации, передаваемой на сервер, должен быть существенно меньше списка пар «аргумент, значение».

2. Этот класс должен быть замкнут относительно операций сложения и умножения, а также всех дополнительных операций, которые мы хотим производить над зашифрованными данными.

3. Класс должен быть достаточно велик, чтобы обеспечить устойчивость к атакам грубой силой.

4. Должна быть возможность легкого конструирования функций из данного класса по уравнению $x_i = g_i(k_0)$.

5. Наконец, функции из этого класса должны быть легко вычислимы, по крайней мере в счетном количестве точек.

В наших алгоритмах в качестве такого класса функций выбраны полиномы от одной или нескольких переменных. В результате появились три схемы, являющиеся полностью гомоморфным шифрованием, применимые для шифрования целых чисел из кольца $\mathbb{Z}_n$, что фактически является обобщением компьютерной арифметики целых чисел, отдельных бит и чисел с плавающей точкой.

## 3. Три новые схемы гомоморфного шифрования

### *3. 1. Шифрование для чисел из R*

Первый алгоритм является прямой реализацией для приведенной выше общей схемы с использованием полиномов от одной переменной и коэффициентами из R. Использование действительных чисел в качестве коэффициентов позволяет вычислять практически произвольные функции от действительных аргументов. В частности, при помощи рядов Фурье было реализовано вычисление частного, квадратного корня, степени и т. д. Другим существенным достоинством данного подхода является возможность сравнения чисел без раскрытия их точного значения, чего нет ни в одной другой схеме.

Однако, данной схеме свойственны и недостатки, схожие с недостатками схемы Гентри: быстрый рост степени многочленов при умножении и рост значений коэффициентов при сложении. Вкупе с ограниченной машинной точностью точность вычислений получается не очень высокой и сами вычисления — довольно медленными.

### *3. 2. Шифрование для чисел из $Z_n$*

Второй алгоритм основан на гомоморфизмах колец полиномов от одной переменной над $\mathbb{Z}_n$, шифрование применяется к целым числам.

Первый этап шифрования — сопоставление полинома исходному числу. Пусть у нас есть число $a_0 \in \mathbb{Z}_n$, сопоставим ему полином $a(x) = a_0 + a_1x^1 + \ldots + a_kx^k$, где коэффициенты $a_1, \ldots, a_k$ выбираются случайно. Аналогично сопоставим числу $b_0 \in \mathbb{Z}_n$ полином $b(x)$. Заметим, что свободные члены полиномов $a(x) \cdot b(x)$ и $a(x) + b(x)$ равны $a_0b_0$ и $a_0 + b_0$ соответственно.

Второй этап — непосредственно шифрование. Рассмотрим отображение $\mathbb{Z}_n[x] \to \mathbb{Z}_n[y]$ при помощи замены переменных $x = c_0 + c_1y^1 + \ldots + c_my^m = c(y)$. Данное отображение используется в качестве шифрующего, где $c(y)$ — секретный ключ. Оно задает гомоморфизм, причем сохраняются операции $+$ и $\times$, и теоретически с их помощью можно осуществить практически любые вычисления.

Для расшифрования предлагается использовать схему Горнера для деления полученного в результате вычислений многочлена на $c(y)$. Результатом деления будет являться многочлен, младший член которого будет числом — результатом вычислений, которые мы хотели произвести.

Таким образом, данная схема представляет собой полностью гомоморфное шифрование. Ее недостатком является рост степени многочленов при их умножении, и решением данной проблемы может стать использование фактор-колец многочленов.

### *3.3. Шифрование для чисел из $\mathbb{Z}_2$*

В основе третьей схемы лежат гомоморфизмы колец полиномов от многих переменных над $\mathbb{Z}_2$ она применяется для шифрования на уровне отдельных битов. Для шифрования чисел $a_0, b_0 \in \mathbb{Z}_2$ построим полиномы $a(x_1, \ldots, x_n)$ и $b(x_1, \ldots, x_n)$, такие, что $a_0$ и $b_0$ соответственно

являются их свободными членами. Как и в предыдущей схеме, свободные члены многочленов $a(x_1, \ldots, x_n) \cdot b(x_1, \ldots, x_n)$ и $a(x_1, \ldots, x_n) + b(x_1, \ldots, x_n)$ будут равны $a_0 b_0$ и $a_0 + b_0$ соответственно.

Для построения «шифрующего» гомоморфизма используется взаимно однозначная замена переменных:

$$\begin{cases} y_1 = f_1(x_1, \ldots, x_n) \\ y_n = f_n(x_1, \ldots, x_n) \end{cases} \quad (6)$$

Для построения таких замен переменных можно использовать интерполяционный многочлен Лагранжа или преобразование Кремоны. Взаимная однозначность замены переменных обеспечивает возможность построения обратной замены и расшифровки данных.

Таким образом, рассмотренная схема также является полностью гомоморфным шифрованием, поскольку позволяет выполнять операции $+$ и $\times$ над зашифрованными данными. Ее важное свойство — отсутствие роста степени полиномов в силу малой теоремы Ферма.

### *3. 4. Дальнейшее усовершенствование алгоритмов шифрования*

Основным направлением текущих исследований являются преодоление роста степени полиномов и улучшение производительности в целом. Так, в данный момент исследуется возможность устранения основного недостатка второй вышеизложенной схемы (рост степени полиномов) с помощью использования фактор-колец полиномов, а также более общая схема, основанная на этой идее.

## 4. Сравнение с существующими схемами полностью гомоморфного шифрования

В целом, предложенные выше схемы обладают следующими свойствами, имеющими важное значение для применения на практике:

1. легкость построения криптосистемы — наиболее вычислительно трудоемкая генерация ключа используется в третьей схеме, где применяется преобразование Кремоны, но все же используемый алгоритм достаточно прост;

2. не ограниченное заранее количество операций — единственным принципиальным ограничением является используемая для хранения шифротекста память, которая в третьей схеме фиксированная, и для первой схемы ограниченная точность чисел с плавающей точкой;

3. легкость распараллеливания вычислений над полиномами.

По сравнению со схемой Гентри наибольшим преимуществом наших схем является второй пункт: в базовой схеме гомоморфного шифрования по Гентри (somewhat homomorphic encryption) количество операций ограничено из-за роста «шума» шифрования. Для подавления этого шума Гентри предложил технику bootstrapping, которая, однако, приводит к значительному росту размеров шифротекста.

Работа Гентри [1] была принята научным сообществом с большим интересом, вследствие чего появилось немало работ, направленных на развитие предложенных в ней идей и устранение ее недостатков. Одна из таких работ «Fully homomorphic encryption from ring-LWE and security for key dependent messages» была представлена З. Бракерски и В. Вайкунтанатаном [3] в 2011 г. В данной работе они предложили альтернативный вариант полностью гомоморфного шифрования, пользуясь идеей LWE (learning with errors) и отказавшись от использования идеальных решеток. Это позволило уменьшить сложность построения криптосистемы, но при этом были унаследованы основные недостатки схемы Гентри:

1. наличие возрастающего шума в шифротексте;

2. рост размера шифротекста при умножении в базовой (somewhat homomorphic) схеме;

3. увеличение объема необходимых данных при использовании механизма bootstrapping.

Фактически предложенная ими схема являлась переложением схемы Гентри на шифрование векторов чисел, для которых было показано, что они образуют кольцо путем задания соответствия с кольцом многочленов по правилу $< a_0, \ldots, a_n > \rightarrow a_0 + a_1 x + \ldots + a_n x^n$.

Несколькими месяцами позже З. Бракерски, К. Гентри и В. Вайкунтанатан в работе «Fully homomorphic encryption without bootstrapping» [4] предложили усовершенствование предыдущей схемы с LWE, основным новшеством которого стала возможность построения схемы шифрования таким образом, чтобы иметь можно было выполнить наперед заданное количество операций без необходимости применения bootstrapping. Это стало возможным благодаря технике modulus switching, приводящей к частичному подавлению шума и значительному снижению скорости достижения им критического уровня. Тем не менее по достижении данной границы необходимо все же либо воспользоваться bootstrapping'ом, либо прекратить вычисления.

Таким образом, фундаментальное различие схем, разрабатываемых в лаборатории НГУ-Parallels и основанных на идеях Гентри, состоит в том, что в предлагаемых нами схемах отсутствует шум как таковой и, следовательно, необходимость предпринятия действий по его подавлению. Это позволяет использовать гораздо более простую математическую модель и существенно повышает эффективность вычислений над зашифрованными данными. Кроме того, одна из предложенных схем обладает свойством отсутствия деградации производительности, чего не может предложить никакая другая известная схема полностью гомоморфного шифрования.

## 5. Перспективы гомоморфного шифрования

В настоящее время множество исследователей работает над созданием законченного решения, позволяющего безопасно обрабатывать секретные данные в облаках. Помимо исследований в направлении гомоморфного шифрования, ведутся поиски и в других направлениях. Например, в MIT была разработана база данных CryptDB [2], использующая методику луковичного шифрования, которая обладает рядом недостатков, но предоставляет достаточно высокий уровень безопасности и возможность обработки и поиска по зашифрованным данным.

Тем не менее лишь полностью гомоморфное шифрование способно исключить необходимость хотя бы частичной расшифровки данных для произведения вычислений над ними. Впрочем, и оно не будет серебряной пулей, способной вытеснить любые другие виды криптографии, поскольку любое подобное шифрование принципиально уязвимо к атаке с подобранным текстом.

К сожалению, в настоящее время нет ни одной реализации, готовой к внедрению в реальные системы. Здравая логика подсказывает, что для того, чтобы гомоморфное шифрование было применимо, должна быть реализация, удовлетворяющая, как минимум, следующим требованиям:

1. Спектр поддерживаемых математических функций должен покрывать повседневные нужды программистов.

2. Диапазоны значений чисел должны покрывать по крайней мере стандартные типы данных, а вычисления, производимые над зашифрованными данными, соответствующие такому размеру чисел, — иметь приемлемую производительность.

3. Точность и скорость вычислений не должны деградировать в течение вычислений.

4. Само по себе вычисление примитивных операций над зашифрованными данными должно иметь сложность $O(n \log(n))$ или даже $O(n)$ от мощности допустимого диапазона значений чисел.

5. Количество разнообразных ключей должно быть достаточно велико, чтобы исключить атаку полным перебором.

Предлагаемые нами алгоритмы уже достаточно близки к тому, чтобы удовлетворить третье, четвертое и пятое требования. Проблему производительности, судя по всему, достаточно легко нивелировать применением графических ускорителей, поскольку операции над полиномами легко

распараллеливаются. Самым сложным является первое требование, и на данный момент имеется лишь приближенное решение этой проблемы с помощью рядов Фурье, о чем упоминалось в описании первого нашего алгоритма.

## СПИСОК ЛИТЕРАТУРЫ:

1. *Gentry C.* A fully homomorphic encryption scheme. Standford, 2009. URL: http://crypto.stanford.edu/craig/ (дата обращения: 05.04.2013).
2. *Popa R. A., Redfield C. M. S., Zeldovich N., Balakrishnan H.* CryptDB: Protecting Confidentiality with Encrypted Query Processing,. MIT CSAIL, 2011.
3. *Brakerski Z., Vaikuntanathan V.* Fully homomorphic encryption from ring-LWE and security for key dependent messages // CRYPTO. 2011. Vol. 6841. P. 505–524.
4. *Brakerski Z., Gentry C., Vaikuntanathan V.* Fully homomorphic encryption without bootstrapping. Cryptology ePrint Archive. Report 2011/277. 2011.