Санкт-Петербургский государственный университет
Математико - Механический факультет

Направление «Математическое обеспечение и администрирование информационных систем»
Профиль «Информационные системы и базы данных»

Малыш Константин Игоревич

Система скрытных вычислений

Бакалаврская работа

Научный руководитель:
д. т. н., профессор Крук Е. А.

Рецензент:
Тьютор, Университет ИТМО Ханов А.Р.

Санкт-Петербург
2016

SAINT-PETERSBURG STATE UNIVERSITY
Main Field of Study «Software and Administration of Information Systems»

Area of Specialisation «Information Systems and Databases»

Malysh Konstantin

Secure computation system

Bachelor's Thesis

Scientific supervisor:
DrSci, professor Krouk Eugine

Reviewer:
Tutor, ITMO University Khanov Arthur

Saint-Petersburg
2016

Оглавление

Введение

В последнее время все большее распространение получают облачные вычисления, которые позволяют совершать операции над данными на удаленном устройстве. Так как в процессе работы с данными, особенно при передаче данных, всегда встает вопрос безопасности, задача построения и реализации в программном виде криптографической системы, которая могла бы гарантировать сохранность информации при передаче данных на вычисление и приеме результатов и в то же время работать быстро и с малыми затратами памяти, требует всестороннего изучения.

1. Цель и постановка задачи

Целью работы является написание библиотеки для работы с предложенной полностью гомоморфной криптографической системой на одном из широко распространенных языков программирования. Такая библиотека могла бы позволить совершать требуемые удаленные операции над данными, оставляя их защищенными. Для достижения результата были поставлены следующие задачи:

1. Изучение предметной области;
2. Выбор системы и средств реализации;
3. Разработка библиотеки;
4. Тестирование библиотеки;
5. Сравнительный анализ с другими реализациями систем.

2. Введение в предметную область

Задача построения защищенной гомоморфной системы была поставлена Рональдом Линном Ривестом, Леонардом Адлеманом и Михаилом Дертузосом в своей работе[9].
Пусть имеется алгебраическая система, состоящая из множества-носителя алгебры S, набора алгебраических операций $\{f_1, f_2, ...\}$, набора предикатов $\{p_1, p_2, ...\}$ и набора констант $\{s_1, s_2, ...\}$. В будущем будем обозначать такую систему как $A =< S; f_1, f_2, ...; p_1, p_2, ...; s_1, s_2, ... >$.
К примеру, в данной нотации традиционная алгебраическая система для целых чисел будет выглядеть как $< \mathbb{Z}, +, -, *, \div, \leq, 0, 1 >$. Пусть имеются алгебраическая система пользователя U и алгебраическая система удаленной системы C, заданные как
$U =< S; f_1, f_2, ..., f_k; p_1, p_2, ..., p_l; s_1, s_2, ..., s_m >$
$C =< S'; f'_1, f'_2, ..., f'_k; p'_1, p'_2, ..., p'_l; s'_1, s'_2, ..., s'_m >$
В данном случае дешифрованием будет являться отображение
$\phi : S' \to S$; шифрованием же будет отображение $\phi^{-1} : S \to S'$.
Пользователь обладает набором информации $\{d_1, d_2, ...\}$, где $d_i \in S \ \forall i$, над которым хочет произвести операции из системы C. Однако, предварительно требуется защитить передаваемые данные, применив к ним шифрующее образование ϕ^{-1}, получая набор $\{\phi^{-1}(d_1), \phi^{-1}(d_2), ...\}$

На данном этапе возникает необходимость определения гомоморфности шифрования. Гомоморфным шифрованием называют криптографический примитив, представляющий собой функцию шифрования, удовлетворяющую дополнительному требованию гомоморфности относительно каких-либо алгебраических операций над открытыми текстами[13]. Различают два отдельных вида гомоморфного шифрования:

1. Частично гомоморфное - гомоморфное относительно одной из двух операций умножения и сложения;

2. Полностью гомоморфное - гомоморфное относительно обеих операций.

Итак, по определению гомоморфной системы, нам необходимо, что-

бы преобразование ϕ являлось гомоморфизмом из C в U.
Пусть пользователь хочет вычислить значение $f_1(d_1, d_2)$. Тогда он просит систему вычислить значение $f_1'(\phi^{-1}(d_1), \phi^{-1}(d_2))$. Так как функция ϕ является гомоморфизмом, то верно, что

$$\phi(f_1'(\phi^{-1}(d_1), \phi^{-1}(d_2))) = f_1(d_1, d_2),$$

поэтому условие гомоморфности системы выполняется.
Также на функцию ϕ и систему C накладываются следующие ограничения:

1. Функции ϕ и ϕ^{-1} должны быть легко вычислимы;
2. f_i' и p_i' из C должны вычисляться рационально;
3. Память, требуемая для размещения зашифрованного значения $\phi^{-1}(d_i)$, не должна превышать память для представления d_i;
4. По большому набору значений $\phi^{-1}(d_i)$ нельзя точно угадать функцию ϕ;
5. По набору значений d_i и некоторым значениям $\phi^{-1}(d_i)$ нельзя точно угадать функцию ϕ.

В конце XX века были опубликованы множество версий частично гоморфных криптографических систем; самой известной из которых является система RSA, гомоморфная только относительно умножения.

В начале XXI века стали появляться первые полностью гомоморфные криптографические системы. В своей работе[5] Крейг Джентри предложил реализацию системы, основанной на теории целочисленных решеток. В основании системы лежит идея добавления числового шума к данным, который делает невозможным их восстановление без знания закрытого ключа. Система Джентри состоит из шифрования, дешифрования и вычисления над данными. При выполнении вычислений числовой шум в данных возрастает, в итоге после выполнения определенного количества операций становится невозможным правильно восстановить

результат. Основным достижением Джентри является то, что в его работе описан механизм уменьшения шума без полного расшифрования данных, называющийся бутстраппингом. Таким образом, с помощью таких перешифрований можно было выполнять бесконечно много операций сложения и умножения.

В 2011 году Кренделев С.Ф. предложил свою полностью гомоморфную криптосистему[12], работа которой основана на гомоморфизме в кольце многочленов от некоторой переменной. Каждому числу $a \in \mathbb{Z}_n$ сопоставляется полином $a(x) = a_0 + a_1x + ... + a_kx^k$, где k и a_i выбирются случайно. По построению, для двух полиномиальных представлений чисел a_0 и b_0 свободный член их суммы $a(x) + b(x)$ и произведения $a(x)b(x)$ равен соответственно $a(0) + b(0)$ и $a(0)b(0)$. Тогда отображение $\phi : \mathbb{Z}_n[x] \to \mathbb{Z}_n[y], x = c_0 + c_1y + ... + c_ty^t = \phi(y)$ - гомоморфизм, сохраняющий операции сложения и умножения. Для дешифрования в работе предлагается использовать схему Горнера для деления на многочлен ϕ(y). Реализация Кренделева является более эффективной, чем у Джентри. В то же время, она имеет ряд недостатков:

1. Неограниченный рост степеней многочленов может привести к неэффективности вычислений;

2. Хотя все действия фактически выполняются над свободными членами, приходится хранить в памяти и делать вычисления над многочленами больших степеней;

3. В системе Кренделева нельзя реализовать деление.

В 2015 году А.Абрамовым была предложена система[11], основанная на гомоморфизме в поле многочленов с рациональными коэффициентами, которая также использует более эффективное представление данных, чем система Кренделева.

3. Описание реализуемой системы

3.1. Задача системы

Пусть имеется целочисленный вектор данных $\{a_1, a_2, ..., a_n\}$. Данная система позволяет вычислить полиномиальную функцию $f(x_1, x_2, ..., x_n)$ на заданном целочисленном векторе, а также выполнить деление с остатком для любой пары элементов.

3.2. Выбор закрытого ключа системы

На первом шаге выбирается целое число
$r < \max\{a_1, a_2, ..., a_n\}$. Каждое из a_i переводится в систему счисления с основанием r и получается представление в виде полинома:
$a_i(x) = a_{i0} + a_{i1}x + ... + a_{im}x^m$, $a_i(r) = a_i$. Полученные многочлены являются элементами поля многочленов с рациональными коэффициентами, над которыми можно выполнять все арифметические операции. Закрытым ключом выступают:

1. Полином с целочисленными коэффициентами $s(x)$, такой, что уравнение $s(x) = r$ имеет ровно одно целочисленное решение. Количество комплексных решений выбирается произвольным и служит степенью безопасности системы;

2. Целое x_r, такое, что $s(x_r) = r$;

3. Целое r, описанное ранее.

3.3. Шифрование

Полиномиальному представлению $a_i(x)$ целого a_i ставится в соответствие многочлен-шифротекст $c_i(x) = a_i(x) \circ s(x)$ Теперь вычисление исходной функции сводится к вычислению $f((c_1(x), c_2(x), ..., c_n(x))$.

3.4. Расшифрование

Пусть требовалось вычислить значение полинома
$R(x) = f(a_1(x), ..., a_k(x))$. Так как использовалось шифрующее преобразование $s(x)$, то полученный с удаленной машины результат можно представить как $R(s(x))$. Следовательно, для вычисления требуемого значения необходимо подставить x_r, так как $s(x_r) = r$ по построению, а $R(r)$ - требуемое значение, так как шифрование было гомоморфным и по свойствам первого шага алгоритма (представления числа в виде многочлена).

3.5. Пример работы системы

Пусть требуется вычислить сумму, произведение и частное от деления чисел a и b, где $a = 257$, $b = 18$. Выберем число $r < \max\{257, 18\} = 257$.
Пусть $r = 7$. Тогда:

$$a_r(x) = 5x^2 + x + 5$$

$$b_r(x) = 2x + 4$$

По построению, $a_r(r) = a, b_r(r) = b$
Введем шифрующее преобразование-полином $s(x)$. Пусть он будет равен
$s(x) = x - 2$, тогда
$x_r = 9$, так как x_r - целый корень уравнения $s(x_r) = r$.
Итого, закрытый ключ системы имеет вид $(s(x), x_r, r) = (x - 2, 9, 7)$ Таким образом, получены зашифрованные представления для a и b:
$c_a(x) = a(x) \circ s(x) = 5(x-2)^2 + (x-2) + 5 = 5x^2 - 19x + 23$
$c_b(x) = b(x) \circ s(x) = 2(x-2) + 4 = 2x$
Теперь необходимо вычислить следующие значения:

1. $S(x) = c_a(x) + c_b(x) = 5x^2 - 17x + 23$
   $S(x_r) = S(9) = 275 = a + b$;

2. $M(X) = c_a(x)c_b(x) = 10x^3 - 38x^2 + 46x$
   $M(x_r) = M(9) = 4626 = ab$

3. $D(x) = (t(x), r(x)) : c_a(x) = t(x)c_b(x) + r(x)$
   $D(x) = (\frac{5}{2}x - \frac{19}{2}, \frac{23}{2x})$
   $D(x_r) = D(9) = (13, \frac{23}{18}) = (14, \frac{5}{18}) = \frac{a}{b}$

3.6. Уязвимости системы

1. Недостаточность шифрования
   Вернемся к представлению изначального значения a_i как полинома. Если выбранное основание системы r оказывается большим, чем a_i, то шифрование становится бессмысленным. Данный факт является одним из основных недостатков системы.

2. Подверженность атаке с подобранным текстом
   Пусть у злоумышленника есть возможность пользоваться системой: передавать в нее функцию для вычисления, набор исходных данных $\{a_i\}$ и получать шифротексты $\{c_i\}$. Тогда атака состоит в следующем:

   (a) Из построения закрытого ключа видно, что секретное значение основания системы r ограничено сверху известным значением. Тогда перебором по всем возможным значениям r можно строить набор многочленов $\{a_i(x)\}$, а затем решать задачу декомпозиции(разложения многочлена в произведение набора многочленов фиксированной длины и фиксированных степеней) для полученного набора шифротекстов.

   (b) Для каждого возможного значения r_j вычисляется соответствующее представление a_i в полиномиальном виде $a_i^{r_j}(x)$; затем решается задача декомпозиции $c_i(x) = a_i^{r_j}(x) \circ s'(x)$, где $s'(x)$ - необходимый злоумышленнику шифрующий полином. При проверке истинного значения r многочлен $a_i^r(x)$ будет являться композицией $a_i'(x)$ и некого одночлена $(x - m_i)$. Отсюда

можно найти m_i, а затем и исходное значение $s(x)$. Следовательно, описанная система подвержена атаке с подобранным текстом, но к ней уязвимы все существующие системы гомоморфного шифрования.

4. Обзор имеющихся решений

На данный момент имеются три реализации гомоморфных криптосистем в открытом доступе:

1. Библиотека на языке C++, автор Shai Halevi.[7]
   Библиотека реализует идею работы Бракерски, Джентри и Вайкунтанатана[1], основанной на теории решеток с оптимизациями
   Джентри-Халеви-Смарта[6]

2. Библиотека на языке C++, авторы Leo Ducas и Daniele Micciancio.[4]
   Эта библиотека представляет собой реализацию улучшения криптосистемы Джентри: обновление результатов(бутстраппинг) происходит быстрее благодаря произведению побитовой операции NAND (не-и). Полностью работа библиотеки описана авторами в их работе[3].

3. Библиотеки на языках C++[2] и Java[8], авторы Kryptnostic.
   Принцип работы решений основан на шифровании с помощью многочленов от нескольких переменных над неким конечным полем. Подробнее система описана в работе[10]

5. Реализация системы

В качестве языка реализации был выбран язык Python. Выбор был сделан по нескольким причинам:

1. На данный момент для языка Python отсутствуют библиотеки полностью гомоморфного шифрования;

2. Python удобен в работе с крупными целыми числами, что является преимуществом, так как при генерации шифрующего многочлена и последующего его применения возможен неограниченный рост коэффициентов конечного полинома.

5.1. Модуль key

В модуле *key* реализованы функции для генерации закрытого ключа:

1. Функция $complexroot(n)$
   Функция *complexroot* по переданному ей целому числу n возвращает пару комплексных чисел $x + i * y, x - i * y$, где $|x| < n$ и $0 < y < n$, причем x и y - целые. Сгенерированные подобным образом пары в дальнейшем становятся корнями многочлена-ключа;

2. Функция $newkey(sec, data)$
   Функция *newkey* принимает на вход целое значение *sec*, являющееся параметром безопасности системы и данные для шифрования *data*. Сначала функция последовательно генерирует с помощью функции *complexroot*
   $$\lfloor \frac{sec - 1}{2} \rfloor$$
   пар комплексных корней, затем получает единственный целый корень. На выходе функция выдает закрытый ключ системы: целочисленный массив коэффициентов шифрующего многочлена, целый корень уравнения и основание системы счисления, в которую мы перевели шифруемые числа.

3. Дополнительные функции для тестирования:

   (a) Для проверки целочисленности коэффициентов полученного массива;

   (b) Для нахождения верности значения вычисленных корней.

5.2. Модуль encryption

В модуле *encryption* реализованы функции для шифрования исходных данных:

1. Функция $inttopoly(num, base)$
   Функция принимает на вход целое число num для шифрования и основание системы $base$. В ходе работы переводит num в многочлен относительно системы счисления с основанием $base$, как описано ранее; в итоге возвращается целочисленный массив коэффициентов.

2. Функция $encrypt(num, key)$
   Функция принимает на вход целое число num для шифрования и закрытый ключ системы key, сгенерированный заранее. В ходе работы функции число сначала представляется в виде многочлена с основанием из закрытого ключа, как описано ранее; затем выполняется функция композиции для полинома закрытого ключа и полученного многочлена. На выходе получается зашифрованное представление числа в виде многочлена с целочисленными коэффициентами.

3. Дополнительные функции для тестирования:

   (a) Для проверки верности представления числа в виде многочлена;

   (b) Для проверки целочисленности коэффициентов сгенерированного многочлена;

(c) Для перевода всех элементов итерируемого объекта в многочлены.

5.3. Модуль decryption

В модуле *decryption* реализована функция для дешифрования полученных данных:

Функция $decrypt(res, key)$
Функция получает на вход многочлен с целыми коэффициентами res и сгенерированный ранее закрытый ключ key. После подстановки корня из ключа, описанной ранее, возвращается дешифрованное значение результирующего многочлена.

5.4. Вспомогательные модули

Также в библиотеке присутствуют:

1. Модуль *sending*
   В модуле *sending* реализованы простейшие функции для сериализации и десериализации полученных полиномов и пересылки/приема данных;

2. Модуль *computation*
   В модуле *computation* реализованы функции для выполнения простейших операций над полиномами. В основном используются в тестах для проверки результатов вычислений на машине и на удаленном сервере;

3. Директория с тестами, проверяющими работоспособность и согласованность работы системы;

4. Документация библиотеки;

5. Установочный файл *setup.py*.

6. Характеристики работы библиотеки

6.1. Генерация ключа

Были произведены замеры времени генерации закрытого ключа с тремя различными параметрами безопасности: $low = 6$, $medium = 12$ и $high = 18$.

Параметр безопасности	Время(с)
low	0.0009
medium	0.0017
high	0.0024

Таблица 1. Время генерации закрытого ключа в зависимости от параметра безопасности системы

6.2. Шифрование

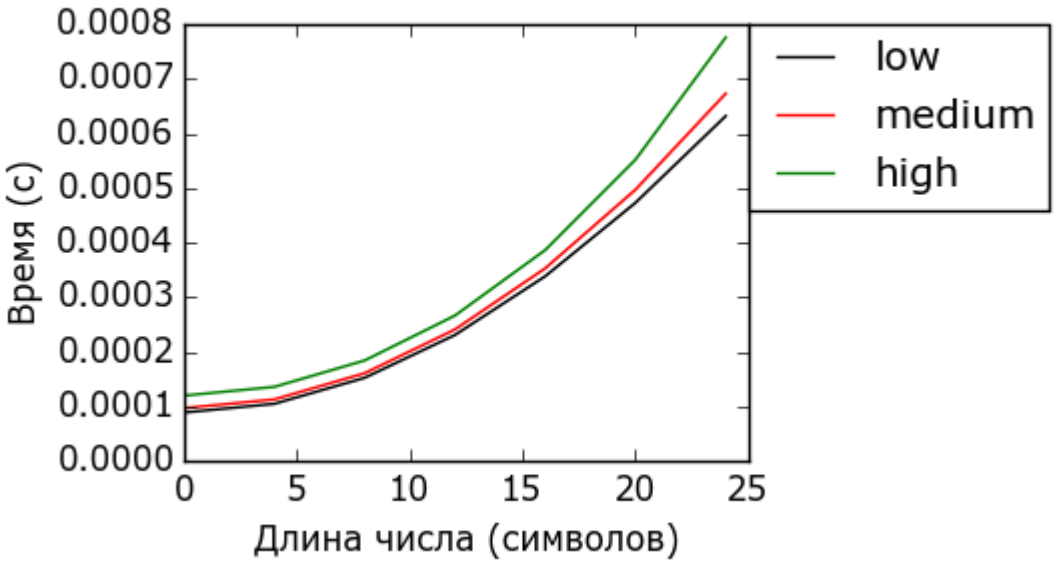

Рис. 1. Время шифрования числа в зависимости от длины числа и параметра безопасности системы

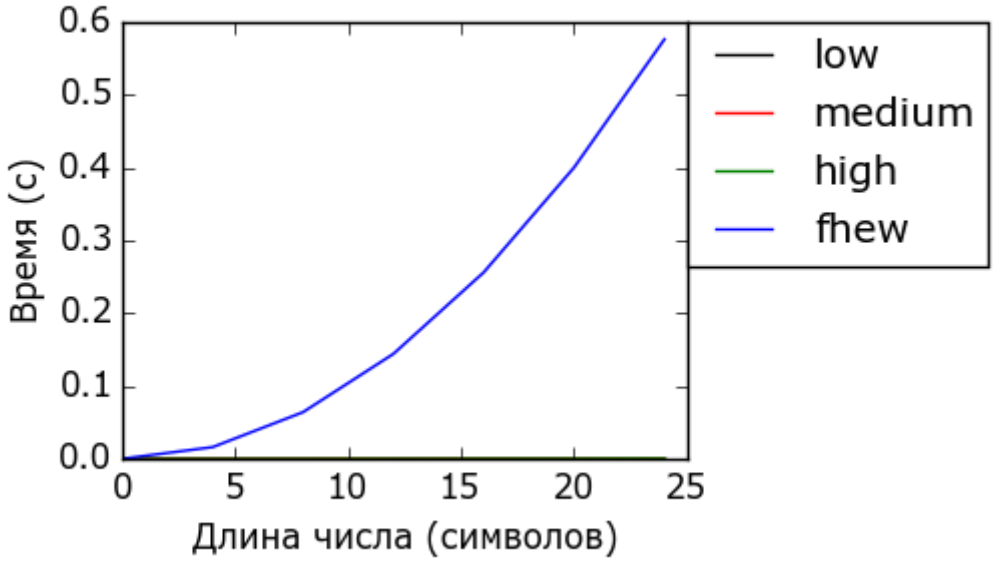

Рис. 2. Время шифрования числа библиотекой FHEW в зависимости от длины числа

На рисунке 2 видно, что кривая для библиотеки FHEW лежит сильно выше, чем три накладывающиеся друг на друга при данном масштабе кривые для библиотеки Абрамова. Оставшиеся две библиотеки показывают результаты большие, чем секунду, уже для чисел, больших 10^9. Результаты работы библиотеки, описанной в работе, превосходят по времени работы результаты других библиотек благодаря несложному построению самой системы и ее незатратности по памяти.

7. Заключение

Результатом данной работы являются:

1. Реализация библиотеки для работы с выбранной полностью гомоморфной криптографической системой на языке Python;
2. Тестирование библиотеки.

Реализация располагается в открытом достуе и может быть найдена по адресу `https://github.com/konstantinmalysh/pyfhe`

Список литературы

[1] Brakerski Zvika, Gentry Craig, Vaikuntanathan Vinod. Fully Homomorphic Encryption without Bootstrapping. — Cryptology ePrint Archive, Report 2011/277. — 2011. — http://eprint.iacr.org/2011/277.pdf.

[2] C++ Implementation of Multivariate Quadratic FHE. — https://github.com/kryptnostic/krypto. — Accessed: 2016-01-16.

[3] Ducas Léo, Micciancio Daniele. FHEW: Bootstrapping Homomorphic Encryption in less than a second. — Cryptology ePrint Archive, Report 2014/816. — 2014. — http://eprint.iacr.org/2014/816.pdf.

[4] FHEW: A Fully Homomorphic Encryption library. — https://github.com/lducas/FHEW. — Accessed: 2016-01-08.

[5] Gentry Craig. A fully homomorphic encryption scheme. — 2009. — URL: https://crypto.stanford.edu/craig/craig-thesis.pdf.

[6] Gentry Craig, Halevi Shai, Smart Nigel P. Homomorphic Evaluation of the AES Circuit. — Cryptology ePrint Archive, Report 2012/099. — 2012. — http://eprint.iacr.org/2012/099.pdf.

[7] HElib: an Implementation of homomorphic encryption. — https://github.com/shaih/HElib. — Accessed: 2016-01-11.

[8] Libraries for fully homomorphic encryption. — https://github.com/kryptnostic/fhe-core. — Accessed: 2016-01-16.

[9] Ronald L. Rivest Len Adleman Michael L. Dertouzos. On data banks and privacy homomorphisms. — 1978.

[10] TAMAYO-RIOS M. Method for fully homomorphic encryption using multivariate cryptography. — 2013. — 12. — US Patent App. 13/915,500. URL: http://www.google.com/patents/US20130329883.

[11] Абрамов Андрей. Гомоморфное шифрование. — 2015.

[12] Кренделев С.Ф. Гомоморфное шифрование (защищенные облачные вычисления) // РусКрипто'2011. — 2011.

[13] Н.П. Варновский А.В. Шокуров. Гомоморфное шифрование. — 2007. — URL: http://www.ispras.ru/proceedings/docs/2007/12/isp_12_2007_27.pdf.