УДК 004.77: 372.862

**Кадан М.А.[1], Макарычев М.А.[2]**

[1] Гродненский государственный университет им. Я. Купалы, г. Гродно, Беларусь
[2] Московский Государственный Технический Университет имени Н.Э. Баумана., г. Москва, Россия

# БЕЗОПАСНЫЕ ВЫЧИСЛЕНИЯ С ИСПОЛЬЗОВАНИЕМ ГОМОМОРФНОЙ КРИПТОГРАФИИ ДЛЯ ОБЛАЧНЫХ ХРАНИЛИЩ ДАННЫХ

**АННОТАЦИЯ**

*В работе рассматривается задача определения требований и подходов к проведению безопасных вычислений над данными облачного хранилища с использованием методов гомоморфного шифрования. Обсуждается возможность использования предложенного метода. Исследована его производительность с использованием приложения на языке Python.*

**КЛЮЧЕВЫЕ СЛОВА**

*Безопасные вычисления; гомоморфное шифрование; защита информации; компьютерная безопасность; облачные технологии; облачные хранилища данных.*

**Kadan M.A.[1], Makarychev M.A.[2]**

[1] Yanka Kupala State University of Grodno, Grodno, Belarus
[2] Bauman Moscow State Technical University, Moscow, Russia

# SECURE COMPUTING IN THE CLOUD STORAGE USING HOMOMORPHIC ENCRYPTION

**ABSTRACT**

*The paper considers the problem of determining the requirements and approaches to safe computing on the data cloud storage using homomorphic encryption methods. The possibility of using the proposed method. Studied its performance using a Python application.*

**KEYWORDS**

*Secure computation; homomorphic encryption; data protection; computer security; cloud technologies; cloud storage.*

## Введение

Не смотря на широкое использование облачных вычислений, хранение и обработка конфиденциальных данных в облачной инфраструктуре небезопасны, что обусловлено, согласно [1], достаточно очевидными рисками нарушения конфиденциальности и целостности данных в облаке:

- доступ к данным со стороны провайдера
- публичное разглашение данных (доступ неограниченного круга лиц)
- выемка данных или носителей из датацентра провайдера (органы правопорядка, сотрудники датацентра)
- ошибки изоляции среды (доступ одного клиента облака к данным других клиентов)
- недостаточное уничтожение данных провайдером при уходе клиента или стирании данных.

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

Одной из основных задач криптографии в данном направлении является обеспечение возможности проведения вычислений над зашифрованными данными без их дешифрования. Данным свойством обладает полностью гомоморфное шифрование.

## Гомоморфное шифрование

Гомоморфное шифрование - форма шифрования, позволяющая производить определённые

математические действия с зашифрованным текстом и получать зашифрованный результат, который соответствует результату операций, выполняемых с открытым текстом [2].

Различают частично гомоморфные и полностью гомоморфные криптосистемы. В то время как частично гомоморфная система позволяет производить одновременно только одну из операций — сложение или умножение, полностью гомоморфные криптосистемы поддерживают одновременное выполнение обеих операций, что позволяет гомоморфно вычислять произвольные вычислительные действия.

**Частично гомоморфные криптосистемы**

Понятие схем шифрования, допускающих нетривиальные вычисления над зашифрованными данными, впервые было предложено еще в 1978 году Рональдом Ривестом, Леонардом Адлеманом, авторами криптосистемы RSA, которая является гомоморфной относительно умножения, в содружестве с Майклом Дертузосом [3].

Пусть $n$ – модуль RSA, и $e$ – открытый ключ шифрования. Функция шифрования имеет вид:

$$E(m) = m^e \bmod n$$

Гомоморфизм по умножению на примере открытых текстов *m1* и *m2* следует из соотношения:

$$E(m1) * E(m2) = m1^e * m2^e \bmod n = (m1m2)^e \bmod n = E(m1m2)$$

Другим известным примером мультипликативно-гомоморфной криптосистемы является криптосистема Эль-Гамаль [4], где

$$E(y, m1) = (y^{r1} m1, g^{r1}),$$
$$E(y, m2) = (y^{r2} m2, g^{r2}).$$

Тогда

$$E(y, m1\ m2) = (y^{r1} y^{r2}\ m1\ m2, g^{r1} g^{r2}).$$

Однако, как говорилось ранее, частично гомоморфные криптосистемы позволяют производить гомоморфные вычисления только для одной операции (или сложения или умножения) открытых текстов.

Гораздо больший интерес в приложениях современной криптографии вызывают полностью гомоморфные криптосистемы.

**Полностью гомоморфные криптосистемы**

Пусть *E(m,k)* - функция шифрования, где *m* - открытый текст, *k* - ключ шифрования.

Функция *E* гомоморфна относительно операции *op* над открытыми текстами, если существует эффективный алгоритм *M*, который, получив на вход любую пару криптограмм вида *E(m1,k), E(m2,k)*, выдает криптограмму, при дешифровании которой будет получен открытый текст *m1 op m2* [5].

Как правило, рассматривается следующий важнейший частный случай гомоморфного шифрования. Для данной функции шифрования *E* и операции *op1* над открытыми текстами существует операция *op2* над криптограммами такая, что из криптограммы *E(m1,k) op2 E(m2,k)* при дешифровании извлекается открытый текст *m1 op1 m2*:

$$E(m1\ op1\ m2, k) = E(m1,k)\ op2\ E(m2,k).$$

Особый интерес представляет возможность построения полностью гомоморфного шифрования, т.е. шифрования, позволяющего проводить над шифротекстами любые необходимые вычисления. К примеру, такую криптосистему можно было бы получить в случае, если бы она была гомоморфна одновременно и по операции сложения, и по операции умножения [6]:

$$D(E(m1,k)\ op1\ E(m2,k),k) = m1 * m2,$$
$$D(E(m1,k)\ op2\ E(m2,k),k) = m1 + m2.$$

Здесь *c* - шифротекст, *op1* и *op2* - операции над шифротекстами, соответствующие операциям умножения (*) и сложения (+) над открытыми текстами.

Таким образом, для создания защищенного облачного сервиса необходимо шифровать поступающие на него данные с помощью полностью гомоморфной схемы шифрования. В этом случае окажется возможным проводить вычисления над данными непосредственно в зашифрованном виде на стороне сервера. При этом шифрование данных будет проводиться на стороне клиента.

**Требования к реализации гомоморфного шифрования**

Авторам не известны реализации гомоморфного шифрования, пригодные для внедрения в реальные программные системы. В то же время, не составляет труда сформулировать, что такая реализация должна удовлетворять, как минимум, следующим требованиям:

- множество поддерживаемых математических функций должен покрывать повседневные нужды программистов.
- диапазоны значений чисел должны покрывать по крайней мере стандартные типы данных, а вычисления, производимые над зашифрованными данными, соответствующие такому размеру чисел, – иметь приемлемую производительность.
- точность и скорость вычислений не должны деградировать в течение вычислений.
- количество доступных ключей должно быть достаточно велико, чтобы исключить атаку полным перебором.

Самым сложным является первое требование, и на данный момент имеется лишь приближенное решение этой проблемы с помощью рядов Фурье, хотя в большинстве приложений, связанных, к примеру, с обработкой экономической информации, круг используемых математических функций весьма сильно ограничен. Более того, вычисления значений многих функций могут быть сведены к работе с заранее подготовленными таблицами.

**Требования к безопасному облачному сервису**

Существующие облачные сервисы не являются полностью защищенными. В лучшем случае есть возможность лишь зашифровать данные на стороне пользователя, но это бывает крайне редко. Обычно данные шифруются ключом, который хранится в том же самом облаке.

Из сказанного выше вытекают несколько конкретных требований к безопасному (защищенному) облачному сервису:

- данные клиента должны храниться в зашифрованном виде, что при их чтении невозможно было бы понять, что это за данные. Причем очевидно, что данные должны поступать на сервер уже зашифрованными;
- шифрование данных должно проводиться на стороне клиента. Более того, ключи шифрования также должны храниться на стороне клиента и должны быть не доступны облачному серверу;
- должна быть возможность обрабатывать эти данные не расшифровывая. Иначе облачный сервер становится всего лишь безопасным хранилищем. А для каждой операции над данными потребуется пересылать их на сторону клиента.

**Реализация гомоморфного шифрования**

В качестве практического этапа исследования было разработано экспериментальное приложение на языке Python для оценки производительности и пригодности алгоритмов, реализующее полностью гомоморфную криптосистему, с возможностью сложения и умножения над зашифрованными данными в кольце $Zn$.

Алгоритм, реализованный в работе, предполагает расширение кольца $Z_2$ на кольцо $Zn$, где $n$ – некоторое достаточно большое натуральное число. Возможности языка Python позволяют задавать значение $n$ произвольно большим, практически ограничивая его значение лишь размером доступной оперативной памяти машины.

Данное требование, связанное с вычислениями в кольце $Zn$, означает, что операнды вычислений должны быть также неотрицательными числами, не превосходящими $n$.

В то же время, для корректности (однозначности) вычислений, сумма и произведение двух чисел также не должна превосходить значение $n$. Это требование кажется сложно выполнимым, так как заранее нельзя ограничивать размер чисел, получаемых в результате умножения произвольных операндов. В то же время, у нас есть возможность выбрать модуль $n$ заведомо достаточно большим.

**Алгоритм работы по организации безопасных вычислений**

Алгоритм работы по организации безопасных вычислений с использованием гомоморфного шифрования может быть описан следующим образом:

1. Генерация публичного и приватного ключей
2. Шифрование операндов
3. Применение к операндам одной из допустимых функций (сложение или умножение)
4. Дешифрование полученного на предыдущем шаге результата.

Рассмотрим данный процесс более детально. В качестве открытого ключа будем использовать некоторое число $p$, взаимно простое с числом $n$, порядком группы $Zn$. Таким образом, выбираем $p$ такое, что наибольший общий делитель $p$ и порядка группы равнялся единице.

Процесс шифрования будет иметь следующий вид:

$$c = z + pq = m + n*r + pq.$$

Процесс дешифрования:

$$m = (c \bmod p) \bmod n.$$

Таким образом, процесс шифрования требует от пользователя еще двух дополнительных значений: $r$ и $q$. Отталкиваясь от этой потребности в качестве публичного ключа будем использовать массив *PublicKey* из 100 элементов, где $i$-тый элемент задан следующим способом:

$$PublicKey\ [i] = p * randomValue + n * r.$$

Здесь randomValue – случайное значение, n– порядок группы, $r$ – случайное число в диапазоне (10, 100).

Видоизменим формулу шифрования следующим образом:

$$c = m + \sum PublicKey^*.$$

Здесь $\sum PublicKey^*$ - сумма некоторого случайного набора приватных ключей, разный для каждого из операндов. Случайность данных наборов обуславливает различие констант шифрования.

Нетрудно проверить, что данный выбор удовлетворяет приведенным выше формулам для шифрования и дешифрования.

Ниже, на рисунке 1, для визуализации результатов работы алгоритма работы по организации безопасных вычислений представлен пример шифрования двух целых чисел и выполнения операций сложения над ними. Данные отображаются в hex-формате и в виде символов таблицы ASCII.

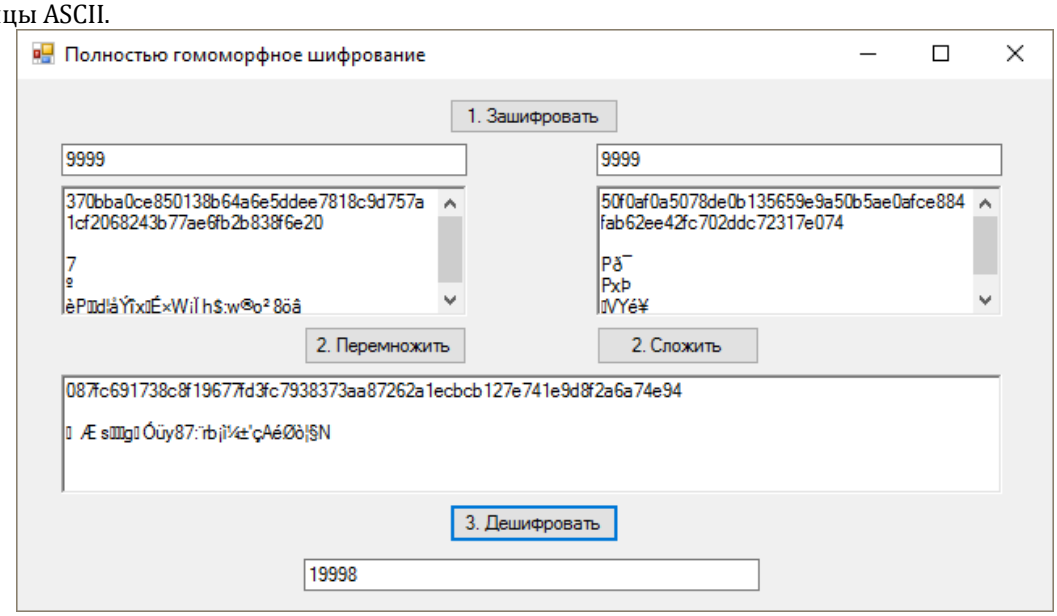


*Рис.1. Пример визуализации работы алгоритма работы по организации безопасных вычислений*

Программный код на язык Python, реализующей функцию генерацию ключей, представлен в листинге 1.

```
from random import randint
nkeys = 100     # размер массива публичных ключей
PublicKey = [0]*nkeys # массив публичных ключей
PrivateKey = 0     # секретный ключ
b1 = 10**10     # нижняя граница диапазона значений ключей
b2 = b1**2      # верхняя граница диапазона значений ключей
modulo = b1     # порядок группы
def GreatestCommonDivisor(a,b):  # вычисление НОД(a,b), итеративный вариант
  while (b != 0):
    a = a % b
    a, b = b, a
  return a
def KeyGen():         # функция генерации ключей
  global nkeys, PublicKey, PrivateKey, modulo, b1, b2
  while True:
    PrivateKey = randint(b1, b2)
    if GreatestCommonDivisor(PrivateKey,modulo) == 1: break
  PublicKey                    =                    [(randint(b1,b2)*PrivateKey)+(modulo*randint(10,100))
      for i in range(0,nkeys)]
  return
```

*Листинг 1. Константы и функция генерации ключей*

Программный код на языке Python, реализующей функции шифрования и дешифрования, приведен в листинге 2.

```
def Encrypt(m):
  global nkeys, PublicKey
  return m+sum([PublicKey[i] for i in range(0,nkeys) if randint(0,1) == 1])

def Decrypt(c):
  global PrivateKey, modulo
  return (c % PrivateKey) % modulo
```

*Листинг 2. Функции шифрования и дешифрования*

Для исследования эффективности реализации системы гомоморфного шифрования был проведен эксперимент, параметрами которого являлись уровень защищенности, типы выполняемых операций, максимальная длина операндов и время работы. Уровень защищенности – характеристика, определяющая значение модуля группы и длину используемых ключей шифрования. В эксперименте использовались модуль и ключи, длина которых составляла от 5 до 8000 десятичных цифр. Используемые операции – шифрование и дешифрование операндов, умножение и сложение операндов. Данные получены для двух значений максимальной длины операндов – 1e10 и 1e20, т.е. для 10-значных и 20-значных десятичных чисел.

Оценки получены при 1000-кратном повторении вычислений с заданными параметрами на компьютере класса Intel(R) CORE (TM) i3-2728 CPU @ 2.20 GHz RAM 4 GB

Результаты приведены в таблице 1 и на графике на рисунке 2.

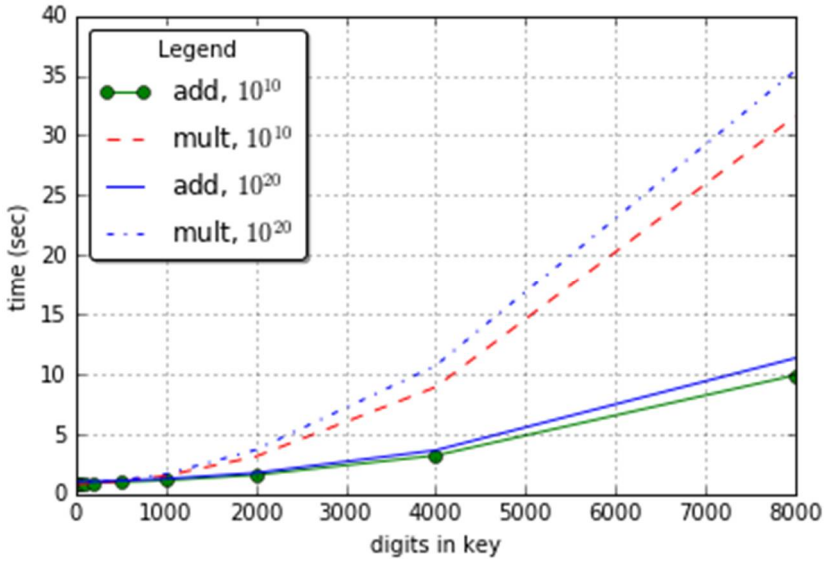


*Рис.2. Графические зависимости реализации характеристик гомоморфного шифрования*

*Таблица 1. Характеристики реализации гомоморфного шифрования*

| Уровень защищенности (макс. кол-во цифр ключа) | Макс. длина операндов = $10^{10}$ 1000 операций | | Макс. длина операндов = $10^{20}$ 1000 операций | |
|---|---|---|---|---|
| | Шифрование, дешифрование, сложение (сек) | Шифрование, дешифрование, умножение (сек) | Шифрование, дешифрование, сложение (сек) | Шифрование, дешифрование, умножение (сек) |
| 5 | 0.92 | 0.85 | 1.07 | 0.86 |
| 10 | 0.93 | 0.84 | 1.04 | 1.09 |
| 20 | 0.92 | 0.84 | 1.03 | 0.93 |
| 50 | 0.94 | 0.85 | 1.04 | 0.94 |
| 100 | 0.94 | 0.86 | 1.04 | 0.95 |
| 200 | 0.94 | 0.90 | 1.04 | 0.99 |

| | | | | |
|---|---|---|---|---|
| 500 | 1.00 | 1.06 | 1.11 | 1.18 |
| 1000 | 1.14 | 1.52 | 1.27 | 1.66 |
| 2000 | 1.59 | 3.12 | 1.76 | 3.68 |
| 4000 | 3.21 | 8.90 | 3.64 | 10.68 |
| 8000 | 9.89 | 31.57 | 11.31 | 35.38 |

Проведенный эксперимент подтвердил эффективность использования рассматриваемого подхода к организации безопасных вычислений. Выбор достаточно большого, в сравнении со значениями используемых операндов, модуля позволяет корректно решать проблему однозначности вычислений. Операции вычитания и деления на целое число, наличие которых необходимо для организации полноценных вычислений, могут быть реализованы через операции сложения и умножения на обратное число (в этом случае модуль группы должен быть простым числом).

**Заключение**

К сожалению, в настоящее время авторам не известна ни одна реализация гомоморфного шифрования, полностью готовая к внедрению в реальные системы.

Для того, чтобы гомоморфное шифрование было эффективно 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. Количество разнообразных ключей должно быть достаточно велико, чтобы исключить атаку полным перебором.

Тем не менее гомоморфное шифрование является мощным аппаратом для сохранности данных в различных прикладных средах. И лишь полностью гомоморфное шифрование способно исключить необходимость хотя бы частичной расшифровки данных для произведения вычислений над ними. Впрочем, и оно не будет способно вытеснить любые другие виды криптографической защиты, поскольку любое подобное шифрование принципиально уязвимо к атаке с подобранным текстом.

*Авторы работы выражают признательность заведующему кафедрой системного программирования и компьютерной безопасности Гродненского государственного университета Кадану Александру Михайловичу за консультации и поддержку в работе.*

## Литература

1. Cloud security alliance [Электронный ресурс] / CSA. - Режим доступа: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf. - Дата доступа:20.04.2016.
2. Homomorphic encryption [Электронный ресурс] / Википедия. - Режим доступа: https://en.wikipedia.org/wiki/Homomorphic_encryption. - Дата доступа: 01.04.2016.
3. R. Rivest, L. Adleman, M. Dertouzos, “On data banks and privacy homomorphisms” [Электронный ресурс]. - Режим доступа: http://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf. – Дата доступа: 01.10.2016.
4. ElGamal encryption [Электронный ресурс] / Википедия. - Режим доступа: https://en.wikipedia.org/wiki/ElGamal_encryption. - Дата доступа: 01.04.2016.
5. Варновский, Н.~П. Гомоморфное шифрование / Н.П. Варновский, А. В. Шокуров // Труды Института Системного программирования: Том~12. (под Ред. В. П. Иванникова). - М.:ИСП РАН, 2006, с. 27-36.
6. Craig Gentry, Fully homomorphic encryption using ideal lattices, Symposium on the Theory of Computing (STOC), 2009, pp. 169-178.

## References

1. Cloud security alliance [Elektronnyy resurs] / CSA. Rezhim dostupa: https://cloudsecurityalliance.org/guidance/csaguide.v3.0.pdf. Data dostupa:20.04.2016.
2. Homomorphic encryption [Elektronnyy resurs] / Vikipediya. Rezhim dostupa: https://en.wikipedia.org/wiki/Homomorphic_encryption. Data dostupa: 01.04.2016.
3. R. Rivest, L. Adleman, M. Dertouzos, “On data banks and privacy homomorphisms” [Elektronnyy resurs]. Rezhim dostupa: http://luca-giuzzi.unibs.it/corsi/Support/papers-cryptography/RAD78.pdf. – Data dostupa: 01.10.2016.
4. ElGamal encryption [Elektronnyy resurs] / Vikipediya. Rezhim dostupa: https://en.wikipedia.org/wiki/ElGamal_encryption. Data dostupa: 01.04.2016.

5. Varnovskiy, N.P. Gomomorfnoe shifrovanie / N.P. Varnovskiy, A. V. Shokurov // Trudy Instituta Sistemnogo programmirovaniya: Tom 12. (pod Red. V. P. Ivannikova). M.:ISP RAN, 2006, c. 27-36.
6. Craig Gentry, Fully homomorphic encryption using ideal lattices, Symposium on the Theory of Computing (STOC), 2009, pp. 169-178.

Поступила: 15.09.2016

**Об авторах:**

**Кадан Мария Александровна**, студентка факультета математики и информатики Гродненского государственного университета им. Я. Купалы, г. Гродно, Беларусь, kadan.maria@gmail.com;

**Макарычев Михаил Алексеевич**, студент факультета «Специальное машиностроение» Московского государственного технического университета имени Н.Э. Баумана., г. Москва, Россия, makar.tula@gmail.com.