УДК 004.056.55

Ковалев Олег Сергеевич
Санкт-Петербургский государственный университет аэрокосмического приборостроения, г. Санкт-Петербург, Россия
Букреев Богдан Александрович
Санкт-Петербургский государственный университет аэрокосмического приборостроения, г. Санкт-Петербург, Россия

ГОМОМОРФНОЕ ШИФРОВАНИЕ

В статье сравнивается производительность двух популярных схем гомоморфного шифрования FHE — BFV и CKKS — с использованием библиотеки Microsoft SEAL. Исследование фокусируется на анализе времени выполнения базовых арифметических операций (сложение, умножение) и динамики бюджета шума. Результаты демонстрируют различия между схемами в скорости операций и в максимальном количестве последовательных вычислений, которые они могут обработать до потери данных.

Ключевые слова: Полностью гомоморфное шифрование, Машинное обучение, BFV, CKKS, Microsoft SEAL

This article compares the performance of two popular Fully Homomorphic Encryption (FHE) schemes, BFV and CKKS, using the Microsoft SEAL library. The research focuses on analyzing the execution time of basic arithmetic operations (addition, multiplication) and the noise budget dynamics. The results demonstrate the differences between the schemes in terms of operation speed and the maximum number of sequential computations they can handle before data loss occurs.

Keywords: Fully Homomorphic Encryption, Machine Learning, BFV, CKKS, Microsoft SEAL

Возможность выполнять вычисления над зашифрованными данными открывает новые перспективы для обеспечения конфиденциальности в

государственном и частном секторах. В отличие от традиционного шифрования, которое подобно помещению документа в сейф (данные необходимо извлечь и расшифровать для работы, что повышает риски), гомоморфное шифрование (ГЭ) позволяет проводить операции непосредственно над зашифрованными данными без их расшифровки. Результаты операций также остаются зашифрованными.

Прорывом стала технология полностью гомоморфного шифрования (FHE), предложенная Крейгом Джентри в 2010 году. FHE позволяет выполнять над зашифрованными данными любые вычисления (сложение, умножение) неограниченное число раз, что является «Святым Граалем» криптографии и крайне перспективно для безопасных облачных вычислений. С тех пор было разработано несколько поколений и схем FHE, включая BFV (2012 г.) и CKKS (2016 г.).

Различные схемы FHE используют уникальные методы управления ростом шума и имеют различные параметры, влияющие на безопасность, производительность и типы поддерживаемых операций. Существует недостаток сравнительных исследований, объясняющих компромиссы между схемами.

В данной статье проводится сравнительный анализ производительности двух популярных схем — BFV и CKKS — с использованием библиотеки Microsoft SEAL. Исследуется время выполнения базовых арифметических операций (сложение, умножение, возведение в квадрат) для определения оптимальной схемы для конкретных случаев использования.

Гомоморфное шифрование классифицируется на три уровня в зависимости от количества и вида математических операций над зашифрованным сообщением.

1. Частично гомоморфное шифрование (PHE): позволяет неограниченное количество операций одного типа (только сложение или умножение).

2. Отчасти гомоморфное шифрование (SHE): поддерживает и сложение, и умножение, но в ограниченном количестве, что сужает круг приложений.

3. Полностью гомоморфное шифрование (FHE): FHE допускает большое разнообразие различных форм аналитических операций над зашифрованным сообщением неограниченное количество раз. К сожалению, вычислительные затраты FHE велики. Выполнение слишком большого количества умножений использует дорогой алгоритм бутстраппинга для обработки зашифрованных данных, позволяя выполнять дополнительные умножения.

Гомоморфное шифрование имеет широкий спектр применений:

Данные в облаке: как обеспечить их безопасность:

Данные могут быть защищены в облаке, сохраняя при этом возможность вычислять и искать зашифрованную информацию, которая впоследствии может быть расшифрована без ущерба для целостности данных в целом, с помощью гомоморфного шифрования.

Включение анализа данных в регулируемых отраслях:

Гомоморфное шифрование защищает конфиденциальность пользователей и пациентов путем шифрования данных и их отправки в коммерческие облачные среды для исследований и обмена данными.

Повышение безопасности выборов:

Гомоморфное шифрование использует операции сложения, что идеально подходит для приложений, связанных с голосованием, поскольку оно позволяет пользователям беспристрастно суммировать различные числа, сохраняя при этом их конфиденциальность

Приватное машинное обучение:

FHE позволяет обучать модели и делать прогнозы на зашифрованных данных, что критически важно для защиты приватности в здравоохранении и машинном нейронном переводе, когда необходимо скрыть смысл текста.

Три основных алгоритма, используемых в схеме шифрования, перечислены ниже:

– KeyGen (параметры безопасности): Параметры безопасности подаются на процесс генерации ключей. Затем выводятся ключи схемы.

– Encrypt (открытый текст, ключ, случайность): Открытый текст, ключ и некоторая случайность являются входными данными для алгоритма шифрования. Затем он выводит соответствующий зашифрованный текст.

– Decrypt (зашифрованный текст, ключ): Алгоритм дешифрования имеет два входных параметра: зашифрованный текст и ключ. Он возвращает эквивалентный открытый текст.

FHE-схемы используют асимметричное шифрование (с разными ключами для шифрования и дешифрования) и классифицируются по вычислительной модели: модульная арифметика, булевы операции или арифметика с плавающей запятой. Существует множество библиотек с открытым исходным кодом (в Таблице 2 приведены несколько библиотек с открытым исходным кодом, например, Microsoft SEAL), однако ощущается недостаток информации о компромиссах между схемами с одинаковой моделью. Выбор оптимальной схемы зависит от конкретных критериев и операций, требуемых приложением.

Таблица 1 – Библиотеки гомоморфного шифрования и поддерживаемые ими схемы

Библиотеки	Поддерживаемые схемы
Microsoft SEAL	Широко используемая библиотека Microsoft с открытым исходным кодом поддерживает схемы BFV и CKKS.
HElib	Первая и наиболее часто используемая библиотека IBM, которая поддерживает схемы CKKS и BGV, а также бутстраппинг.
Lattigo	Это криптографическая библиотека на решетках на языке Go-Lang.
PALISADE	Широко используемая библиотека с открытым исходным кодом, которая поддерживает многочисленные алгоритмы гомоморфного

	шифрования с поддержкой multiparty, включая BGV, BFV, CKKS, TFHE и FHEW.

Шум в зашифрованном тексте — критический параметр. Каждая операция потребляет бюджет шума, и его исчерпание делает расшифровку невозможной. При сложении шум аддитивно накапливается (Уравнение 1), а при умножении и возведении в квадрат — мультипликативно (Уравнение 2), что может привести к взрывному росту и потере данных после нескольких операций.

Эксперимент проводился на векторах размером 2^n. Для каждой схемы (BFV и CKKS) были подобраны соответствующие параметры безопасности.

Для BFV: степень полиномиального модуля, модуль зашифрованного текста, модуль открытого текста.

Для CKKS: степень полиномиального модуля, модуль зашифрованного текста (схема не использует модуль открытого текста).

Результаты сравнения операций между схемами представлены в таблицах и на рисунках ниже.

Noise (x' + y') = Noise(x') + Noise(y') (1)

Noise (x' * y') = Noise(x') * Noise(y') (2)

Таблица 2 – Умножение в BFV

Размер векторов Степень полиномиального модуля(2^n)	Кодирование (мкс)	Шифрование (мкс)	Умножение (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	135	2834	4720	423	62	8174
8192	259	7968	16339	1493	130	25989
16384	524	26797	61986	6377	248	95932

Умножение двух векторов в BFV выполняется, как показано в Таблице 2, путем прохождения 5 операций: кодирование, шифрование, умножение, дешифрование и декодирование. Общее время умножения

увеличивается с увеличением размера вектора, и две наиболее затратные по времени операции — это шифрование и умножение.

Таблица 3 – Сложение в BFV

Размер векторов Степень полиномиального модуля(2^n)	Кодирование (мкс)	Шифрование (мкс)	Сложение (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	135	2834	62	309	85	3425
8192	259	7968	239	1104	103	9673
16384	524	26797	1081	4294	225	32921

В Таблице 3 показаны операции, необходимые для сложения в BFV. Операция сложения по сравнению с умножением в Таблице 2 показывает, что умножение занимает больше времени, чем сложение. Например, при размере вектора 4096 операция умножения заняла 4720 мкс, а сложение — 62 мкс. Более того, общее время для гомоморфного умножения двух векторов составило 8714 мкс, тогда как гомоморфное сложение двух векторов заняло 3425 мкс. Это показывает, что среднее время, затрачиваемое на умножение, на 42% больше, чем на сложение в схеме BFV.

Таблица 4 – Возведение в квадрат в BFV

Размер векторов Степень полиномиального модуля(2^n)	Кодирование (мкс)	Шифрование (мкс)	Возведение в квадрат (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	65	1450	3210	716	55	5436
8192	128	3975	11052	2234	98	17487
16384	264	13412	47461	8883	196	70216

В Таблице 4 показано возведение в квадрат вектора в BFV, которое практически аналогично умножению двух векторов, но с кодированием и

шифрованием только одного вектора, что уменьшает общее время по сравнению с умножением. Для схемы BFV она позволяет пользователю упаковывать целые числа в один полином открытого текста и оперировать этими целыми числами в стиле SIMD (Одна инструкция — множество данных), называемом batch encoding (пакетное кодирование). Однако в схеме CKKS кодирование также осуществляется в полиномы, но по другой технике, чем в BFV, и с использованием других параметров. Теория, стоящая позади CKKS, полностью отличается от BFV, хотя обе функции кодирования выглядят похожими.

Таблица 5 – Умножение в CKKS

Размер векторов	Кодирование (мкс)	Шифрование (мкс)	Умножение (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	8907	9991	1451	298	3989	24636
8192	18094	19589	2924	556	8470	49633
16384	37806	41156	5973	1253	18277	104465

Как показано в Таблице 5, умножение в CKKS выполняется между двумя векторами, и указано время, затраченное на каждую операцию, выполненную над векторами. Две наиболее затратные по времени операции — это кодирование и шифрование, как показано в таблице.

Таблица 6 – Сложение в CKKS

Размер векторов	Кодирование (мкс)	Шифрование (мкс)	Сложение (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	8907	9991	56	330	3767	23051
8192	18094	19589	120	552	8063	46418
16384	37806	41156	315	1133	16826	97236

В Таблице 6 выполняется сложение в CKKS между двумя векторами, и показано, что время, затрачиваемое на сложение, по сравнению с умножением, невелико. Общая разница во времени между умножением и

сложением между двумя векторами невелика и в среднем на 7% больше, чем сложение в схеме CKKS.

Таблица 7 – Возведение в квадрат в CKKS

Размер векторов	Кодирование (мкс)	Шифрование (мкс)	Возведение в квадрат (мкс)	Дешифрование (мкс)	Декодирование (мкс)	Общее время (мкс)
4096	4287	5078	680	904	3964	14913
8192	9079	9782	1378	1725	8125	30107
16384	18840	20446	2884	3844	16913	62927

Наконец, в Таблице 7 показано время, затраченное на операцию возведения в квадрат в CKKS. Две наиболее затратные по времени операции — это кодирование и шифрование Время выполнения операций (кодирование, шифрование, операция, дешифрование, декодирование) измерялось усреднением 50 запусков для каждой схемы (BFV и CKKS) и представлено в Таблицах 3-8.

На Рисунке 1 BFV показала меньшее время для всех размеров векторов. На Рисунке 2 BFV была быстрее на векторах 4096 и 8192, тогда как CKKS показала небольшое преимущество на размере 16384.

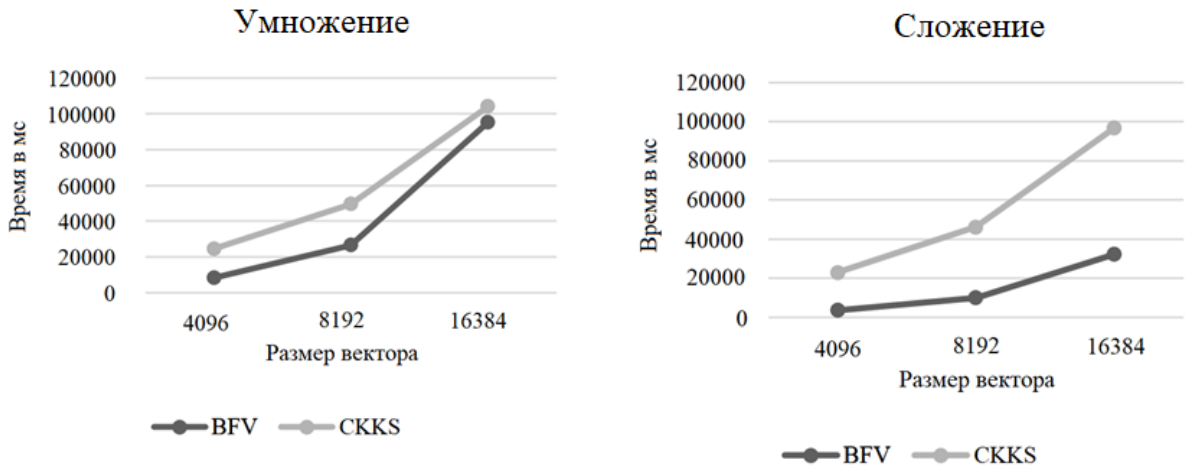

Рисунок 1 – Время сложения и умножения в BFV и CKKS

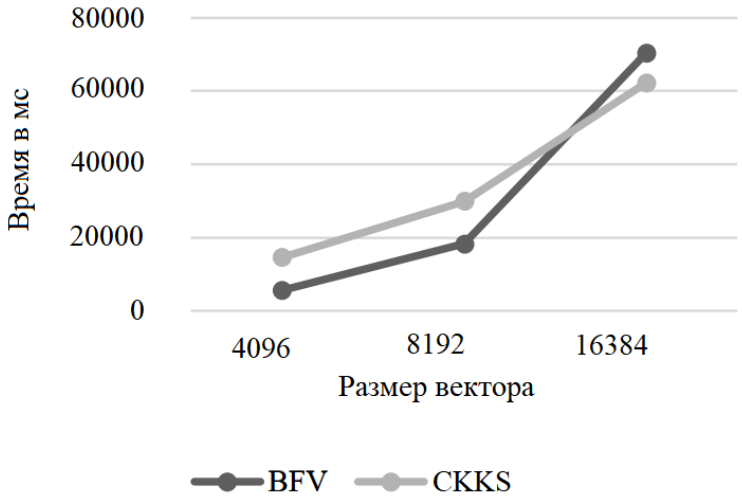

Рисунок 2 – Время возведения в квадрат в BFV и CKKS

Рост шума, особенно при умножении, критически влияет на количество возможных последовательных операций. Для управления шумом и размером зашифрованного текста применялась релинеаризация. Релинеаризация — это процедура, которая уменьшает размер зашифрованного текста обратно до его исходного размера. Хотя релинеаризация имеет высокую вычислительную стоимость, она может оказать большое положительное влияние как на рост шума, так и на производительность. Оба метода, BFV и CKKS, используют релинеаризацию одинаковым образом. Однако в схеме CKKS с тем же количеством слотов вектора расчетное количество последовательных умножений составило 2 раза. Остальные размеры векторов показаны на рисунке 4. Когда бюджет шума достигает нуля, нельзя гарантировать, что дешифрование произведет желаемый результат. Когда бюджет шума достигает нуля, корректная расшифровка невозможна. Вычисления над меньшими зашифрованными текстами менее затратны и имеют больший запас по шуму.

Все наши эксперименты проводились на AMD Ryzen 7 5800X 8-Core Processor 3.80 GHz, NVIDIA GeForce RTX 4070.

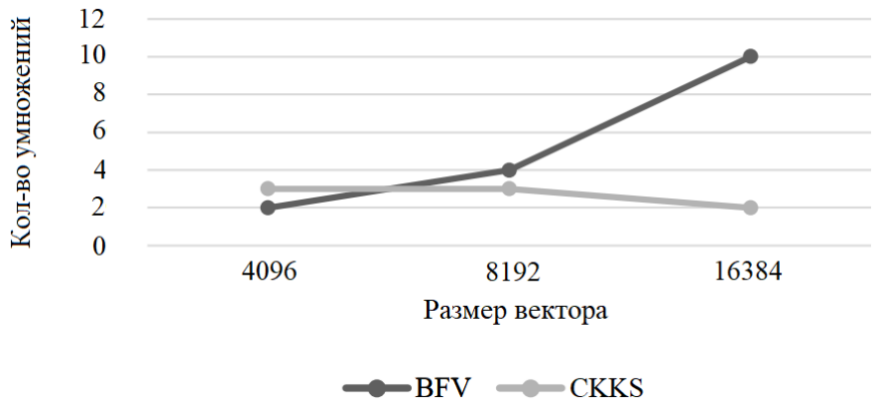

Рисунок 3 - Максимальное количество последовательных умножений в схемах BFV и CKKS

На Рисунке 4 показано количество последовательных умножений, которое каждая схема может обработать в соответствии с размером вектора, по сравнению друг с другом. Для размера вектора 4096 схема BFV обработала 2 последовательных умножения. В то же время CKKS обработал 3 раза, что лучше, чем BFV. При больших размерах векторов, таких как 8192 и 16384, CKKS обработал меньшее количество последовательных умножений по сравнению со схемой BFV.

В этой статье был проведен анализ производительность двух схем полностью гомоморфного шифрования, BFV и CKKS, с использованием библиотеки FHE «Microsoft SEAL» путем применения определенных арифметических операций и анализа ее производительности путем вычисления затраченного времени и наблюдения за загруженным шумом, особенно в последовательных операциях. Будущая работа будет заключаться в сравнении большего количества библиотек, таких как Lattigo и HElib, чтобы их можно было использовать с подходящей схемой в приложениях машинного обучения. Кроме того, в области вычислительной лингвистики выполнение безопасного нейронного машинного перевода без раскрытия любого текста будет переводиться с использованием схем гомоморфного кодирования, на которых будет сосредоточена будущая работа.

Список использованных источников

1. В. Ф. Роша, Дж. Лопес и В. Фалькан Да Роша, «Обзор алгоритмов гомоморфного шифрования», 2019.

2. Кустов И.А., Садов В.С. Применимость гомоморфных криптографических систем. В сборнике: Компьютерные технологии и анализ данных (CTDA'2020). материалы II Международной научно-практической конференции. Минск, 2020. С. 77-81.

3. Стародубцев В.С. Реализация алгоритмов гомоморфного шифрования с использованием различных библиотек. Неделя науки 2022. Сборник тезисов : в двух частях. Редакционная коллегия: Я. А. Асланов, О.В. Батычко, М. А. Лачугина, Н. П. Сохиева ; МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ, Федеральное государственное автономное образовательное учреждение высшего образования «ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ». 2022. С. 678-681.

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

5. Чхайло И.Д., Трацевский И.Д. Гомоморфное шифрование и теория категорий. Перспектива-2021. Материалы IX Всероссийской молодежной школы-семинара по проблемам информационной безопасности. Москва, 2022. С. 47-50.

6. Донченко М.А., Синицын А.А. Сравнительный анализ библиотек гомоморфного шифрования. Современное программное обеспечение систем информационной безопасности и интеллектуальной поддержки управленческих решений. Сборник научных статей аспирантов. Москва, 2024. С. 21-25.

7. HElib, официальный репозиторий библиотеки. URL: https://github.com/homenc/HElib (дата обращения 11.08.2025).