ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

На правах рукописи

Луценко Владислав Вячеславович

Разработка математической модели, методов и алгоритмов для повышения скорости обработки данных в туманных вычислениях с использованием модулярной арифметики

Специальность 2.3.5 —
«Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
Доктор физико-математических наук, доцент
Бабенко Михаил Григорьевич

Ставрополь — 2025

Оглавление

Стр.

Стр.

Стр.

Введение

Актуальность темы исследования. Развитие туманных вычислений (Fog Computing, FC) как перспективной технологии обработки данных в условиях роста количества устройств Интернета вещей (Internet of Things, IoT) и необходимости обработки информации в реальном времени привело к значительному увеличению требований к скорости и безопасности обработки и хранения информации. Туманные вычисления, обеспечивая обработку данных ближе к источнику их генерации, позволяют снизить задержки и уменьшить нагрузку на облачные серверы. Однако такие системы сталкиваются с рядом проблем, связанных с ограниченными вычислительными ресурсами граничных устройств и необходимостью обеспечения высокой скорости обработки данных при сохранении их конфиденциальности. Традиционные алгоритмы шифрования, зачастую оказываются слишком ресурсоемкими для устройств с ограниченной производительностью, что делает их применение в туманных вычислениях недостаточно эффективным.

Одним из направлений для решения этой проблемы является применение системы остаточных классов (СОК). СОК позволяет выполнять параллельные вычисления над остатками числа, что значительно ускоряет выполнение арифметических операций в алгоритмах шифрования [100]. Это особенно важно для туманных вычислений, где критически важны низкие задержки и высокая производительность. Кроме того, использование СОК может снизить энергопотребление, что актуально для IoT-устройств, работающих от батарей. Однако, несмотря на потенциальные преимущества, применение СОК в криптографических алгоритмах для туманных вычислений остается недостаточно изученным, что делает данное направление актуальным.

Для успешного внедрения СОК в туманные вычисления необходимо решить ряд задач, таких как оптимизация модульных операций СОК, а также разработка эффективных методов выполнения немодульных операций СОК, таких как обратное преобразование, определение знака числа и сравнение чисел. Решение этих задач открывает новые возможности для создания высокопроизводительных и безопасных систем обработки данных в туманных вычислениях.

Значительный вклад в развитие методов вычислений в СОК оказали российские ученые И.Я. Акушский, Д.И. Юдицкий, В.М. Амербаев, Н.И. Чер-

вяков, В.С. Князьков, А.А. Коляда, Ш.А. Оцоков, за рубежом — H.L. Garner, A. Omondi, N. S. Szabo D. Schoinianakis, G.C. Cardarilli, J.C. Bajard, G. Pirlo и другие.

Целью диссертационного исследования является повышение скорости работы узлов туманных вычислений в алгоритмах шифрования за счет оптимизации вычислительно сложных процедур модулярного кода и разработки программного обеспечения для их проектирования.

Объектом диссертационного исследования является теория обработки данных в распределенных вычислительных системах.

Предмет исследования — методы и алгоритмы модулярной арифметики, направленные на ускорение обработки данных в туманных вычислениях.

Научная задача диссертационной работы состоит в исследовании и разработке математических методов и алгоритмов выполнения вычислительно сложных операций СОК на основе функции ядра Акушского, способных повысить скорость при выполнении алгоритмов шифрования вычислительными узлами туманной среды, работающими в системе остаточных классов.

Для решения поставленной научной задачи была произведена ее декомпозиция на следующие частные **задачи**:

1. Разработка алгоритмов для выбора оптимальных параметров СОК, таких как построение компактных базисов специального вида и поиск оптимальных весов для функции ядра Акушского.
2. Разработка и модификация методов и алгоритмов выполнения вычислительно сложных операций в СОК, таких как обратное преобразование из СОК в позиционную систему счисления (ПСС), деление чисел, определение знака числа и сравнение чисел для выполнения обработки данных.
3. Разработка программного комплекса выполнения модульных и немодульных операций в системе остаточных классов для распределенной обработки данных в туманных вычислениях.

Соответствие паспорту научной специальности. Тема и основные результаты диссертации соответствуют следующим областям исследований паспорта специальности ВАК 2.3.5 — «Математическое и программное обеспечение вычислительных систем, комплексов и компьютерных сетей»:

8. Модели и методы создания программ и программных систем для параллельной и распределенной обработки данных, языки и инструментальные средства параллельного программирования.

9. Модели, методы, алгоритмы, облачные технологии и программная инфраструктура организации глобально распределенной обработки данных.

Научная новизна:

1. Разработаны алгоритмы выбора оптимальных параметров функции ядра Акушского для вычисленных компактных базисов СОК.
2. Модифицированы методы и алгоритмы перевода из СОК в позиционную систему счисления, деления, определения знака и сравнения чисел в СОК, отличающиеся от известных меньшей размерностью операндов и эффективной реализацией операций без необходимости нахождения остатка по большому модулю, за счет выбора оптимальных параметров функции ядра Акушского.
3. Разработан программный комплекс для выполнения немодульных операций в туманных узлах, позволяющий повысить скорость распределенной обработки данных в туманных вычислениях.

Моделирование и вычислительные эксперименты проведены на компьютере, оснащенным процессором Intel Core i7-7700HQ с тактовой частотой 2.80 ГГц, 8 ГБ оперативной памяти DDR4 с частотой 1196 МГц и твердотельным накопителем емкостью 512 ГБ, работающем под управлением Windows 10 Home Edition, с использованием языков программирования высокого уровня C++ и Python.

Практическая значимость результатов исследования заключается в разработке математической модели, методов и алгоритмов, позволяющих повысить скорость обработки данных в туманных вычислениях за счет использования модулярной арифметики. Результаты исследования могут быть применены в облачных и туманных структурах, работающих в условиях ограниченных вычислительных ресурсов. Кроме того, разработанные методы и программные средства могут быть применены в других областях, где требуется параллельная реализация арифметических операций с числами большой разрядности.

Методология и методы исследования включают использование математического аппарата высшей алгебры, теории чисел, модулярной арифметики, теории алгоритмов, численных методов, линейной алгебры, теории вероятно-

стей, а также методов математического моделирования и теории параллельных вычислений.

Основные положения, выносимые на защиту:

1. Метод перевода из системы остаточных классов в позиционную систему счисления на основе Китайской теоремы об остатках (КТО) и ранга числа функции ядра Акушского.
2. Алгоритмы определения знака числа для наборов модулей специального вида с использованием минимальной функции ядра Акушского с заданными свойствами.
3. Метод построения функций ядра Акушского для операции сравнения чисел.
4. Алгоритм построения компактных базисов специального вида для системы остаточных классов.
5. Алгоритмы поиска оптимальных весов функции ядра Акушского.

Достоверность полученных в диссертационной работе результатов обеспечивается строгостью математических доказательств и корректным использованием методологического аппарата исследований. Справедливость выводов относительно эффективности и корректности предложенных подходов проверена посредством компьютерного моделирования и экспериментальных исследований.

Личный вклад автора. Все изложенные в диссертационной работе результаты получены при непосредственном участии автора. Из результатов работ, выполненных коллективно, в диссертацию включены только полученные непосредственно автором. В работе [69] автором рассмотрена проблема критических ядер функции ядра Акушского и предложены методы их определения, необходимые для реализации операций сравнения чисел и определения их знака в системе остаточных классов. В работе [19] исследованы методы обратного преобразования из СОК в ПСС. В работе [73] автором предложен метод обратного преобразования из системы остаточных классов в позиционную систему счисления с использование КТО и ранга числа, который вычисляется на основе функции ядра Акушского. В работе [8] автором предложена модификация итерационного деления в СОК с использованием функции ядра Акушского. В работе [4] автором исследована эффективность специальных наборов модулей системы остаточных классов. В работе [28] автором предложен метод определения знака числа в СОК, основанный на использовании приближённо-

го ранга числа, вычисляемого с помощью функции ядра Акушского. В работе [10] автором предложено использование генетического алгоритма для поиска оптимальных весов функции ядра. Разработан программный комплекс вычислительных модулей туманной среды для оптимизации модульных операций СОК, а также выполнения модифицированных методов и алгоритмов вычисления немодульных операций в СОК, на который получены свидетельства о государственной регистрации программ для ЭВМ [11—18].

Апробация работы. Основные результаты диссертационного исследования докладывались на международных и всероссийских конференциях, среди которых «Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2025)» (г. Пятигорск, Россия), «Открытая конференция ИСП РАН им. В.П. Иванникова (ISPRASOPEN 2024)» (г. Москва, Россия), «CPAMCS-2024: Current Problems in Applied Mathematics and Computer Systems» (г. Ставрополь, Россия), «Spring/Summer Young Researchers' Colloquium on Software Engineering (SYRCoSE 2024)» (г. Ставрополь, Россия), «International Workshop on Advanced in Information Security Management and Applications (AISMA-2024)» (г. Алигарх, Индия, г. Ставрополь, г. Красноярск, Россия), «International Conference on Communication and Computational Technologies (ICCCT 2024)» (г. Джайпур, Индия), «Открытая конференция ИСП РАН им. В.П. Иванникова (ISPRASOPEN 2023)» (г. Москва, Россия), «Национальный Суперкомпьютерный Форум (НСКФ-2023)» (г. Переславль-Залесский, Россия), «Всероссийская научно-практическая конференция имени Жореса Алфёрова» (г. Санкт-Петербург, Россия), «CPAMCS-2023: Current Problems in Applied Mathematics and Computer Systems» (г. Ставрополь, Россия), «International Workshop on Advanced in Information Security Management and Applications (AISMA-2023)» (г. Алигарх, Индия, г. Ставрополь, г. Красноярск, Россия), «VII Всемирный Конгресс Математиков тюркского мира (TWMS Congress 2023)» (г. Туркестан, Казахстан), «International Conference on Mathematics and its Applications in New Computer Systems (MANCS 2021)» (г. Ставрополь, Россия).

Публикации. Основные результаты по теме диссертационного исследования изложены в 16 публикациях, 5 из которых изданы в журналах, рекомендованных ВАК [4; 8; 10; 45; 73], 9 — в тезисах докладов конференции [7; 19; 28; 52; 61; 68; 70—72], 9 — в публикациях, входящих в международные

базы цитирования Web of Science и Scopus [28; 52; 61; 68—72; 102]. Получено 8 свидетельств о государственной регистрации программ для ЭВМ [11—18].

Внедрение. Результаты диссертационной работы были использованы при выполнении проектов: гранта РНФ № 19-71-10033 «Эффективная, безопасная и отказоустойчивая система распределенного хранения и обработки конфиденциальных данных с регулируемой избыточностью для проектирования мобильных облаков на маломощных вычислительных устройствах», гранта РНФ № 22-71-10046 «Разработка новых методов и алгоритмов для повышения надежности и безопасности хранения, передачи и обработки данных в туманных вычислениях», гранта РНФ № 24-21-00149 «Разработка модульных искусственных нейронных сетей ориентированных на туманные вычисления», гранта Северо-Кавказского федерального университета «Интеллектуальный блок управления распределенной системой хранения данных в гетерогенных средах с регулируемой избыточностью и безопасностью», гранта РНФ № 25-71-30007 «Новые технологии для проектирования облачных сервисов машинного обучения, сохраняющих конфиденциальность» (глава 2, глава 3). Кроме того, ряд результатов работы использован в Северо-Кавказском центре математических исследований в рамках соглашения № 075-02-2024-1451 с Министерством науки и высшего образования Российской Федерации.

Объем и структура работы. Диссертация состоит из введения, 3 глав, заключения и 7 приложений.

В первой главе описана архитектура распределенной обработки данных, которая лежит в основе IoT, включая особенности передачи и обработки данных в распределенных системах и требования к их безопасности. Рассмотрены асимметричные алгоритмы шифрования, применимые для IoT. Исследованы методы повышения производительности алгоритмов шифрования в туманных вычислениях. Исследована возможность применения СОК для повышения скорости алгоритмов шифрования в туманных вычислениях. Рассмотрены проблемы арифметики системы остаточных классов. Таким образом, сформулирована научная задача исследования: исследование и разработка математических методов и алгоритмов выполнения вычислительно сложных операций СОК на основе функции ядра Акушского, способных повысить скорость при выполнении алгоритмов шифрования вычислительными узлами туманной среды, работающими в системе остаточных классов.

Во второй главе рассмотрены математические основы определения позиционной характеристики числа в СОК с использованием функции ядра Акушского. Рассмотрена проблема критических ядер функции ядра, ограничивающая возможность использования оптимальной формы функции ядра. Разработан метод определения критических ядер, основанный на Теореме 2.2.1, который в среднем на 99% быстрее алгоритма усеченного перебора.

Исследованы методы обратного преобразования чисел из СОК в позиционную систему счисления, включая Китайскую теорему об остатках, ее приближенный вариант, обобщенную позиционную систему счисления, диагональную функцию и функцию ядра. В результате предложен новый алгоритм, использующий КТО и ранг числа, вычисляемый с помощью функции ядра. Доказано равенство рангов числа, определенного на основе КТО и функции ядра Акушского, при отсутствии критических ядер.

Рассмотрен алгоритм итерационного деления чисел в СОК, а также предложена его модификация с использованием новых методов вычисления функции ядра от половины числа.

Исследованы методы определения знака числа в СОК, основанные на КТО, приближенном методе на базе КТО, обобщенной позиционной системе и диагональной функции. Разработаны минимальные функции ядра Акушского с заданными свойствами для специальных наборов модулей. Их применение позволяет сократить размер операндов и снизить вычислительную сложность операции вычисления остатка от деления с $O(n^2)$ до $O(n)$.

Проведен анализ алгоритма сравнения чисел с использованием функции ядра Акушского. Рассмотрена проблема монотонности функции ядра, необходимая для корректного сравнения чисел в СОК. Разработан метод построения функций ядра Акушского для операций сравнения чисел, у которых выполняется условие $C_P = 2^N$. Данный метод позволяет сократить размер операндов и заменить операцию нахождения остатка от деления взятием младших бит числа.

Таким образом, в главе представлены методы реализации всех ключевых немодульных операций, необходимых для построения системы обработки данных, работающей в системе остаточных классов.

В третьей главе продемонстрирована практическая реализация предложенных во второй главе методов и алгоритмов в виде программного обеспечения, предназначенного для работы в распределенных туманных средах.

Предложена архитектура программного комплекса для проектирования высокоскоростных вычислительных систем на основе модулярной арифметики. Архитектура программного комплекса включает модули выбор базиса СОК, поиска оптимальных весов для функции ядра Акушского, модулярных вычислений и вычисления позиционной характеристики для немодульных операций СОК.

Проведено исследование эффективности наиболее используемых наборов модулей специального вида СОК. Наибольшую эффективность в операциях сложения вычитания и умножения показали наборы $\{2^n - 1, 2^n, 2^n + 1\}$ и $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$. Предложен алгоритм построения компактных базисов с использованием теоремы Диемитко. Алгоритм построения компактных базисов позволяет снизить время выбора набора модулей СОК в среднем на 17% по сравнению с методом на основе чисел Мерсенна и на 73% — с методом общей фильтрации. В среднем, использование компактных наборов модулей позволяет ускорить модульные операции на 12% по сравнению с наборами специального вида.

Рассмотрено моделирование модулярного сложения и умножения в СОК с использованием компактных базисов. Операции в СОК сравнивались с реализациями модулярного сложения и умножения в двоичной системе счисления с использованием библиотек NTL и MIRACL. В результате, сложение в СОК оказалось быстрее на 36.5% по сравнению с NTL и на 51.2% по сравнению с MIRACL, а умножение — на 38.1% быстрее NTL и на 53.7% быстрее MIRACL.

Для вычисления позиционной характеристики числа в СОК была использована функция ядра Акушского. Предложены алгоритмы поиска оптимальных весов для функции ядра с использованием метода Монте-Карло и генетического алгоритма. В среднем, генетический алгоритм оказался на 71.2% быстрее метода Монте-Карло, что делает его более предпочтительным при работе с большим числом модулей.

Проведено моделирование методов обратного преобразования из СОК в позиционную систему счисления. Предложенный алгоритм, основанный на использовании КТО и ранга числа функции ядра Акушского, в среднем на 10.5% быстрее КТО при фиксированном количестве модулей и на 7.2% быстрее при динамическом изменении числа модулей.

Рассмотрен алгоритм итерационного деления в СОК. Для оптимизации итерационного деления использованы новые методы вычисления функции яд-

ра от половины числа. В результате, итерационное деление с вычислением функции ядра от половины числа оказалось на 12.2% быстрее классического итерационного деления. Однако данный метод применим только при использовании нечетных модулей в базисе СОК.

Разработаны алгоритмы определения знака числа в СОК на основе минимальной функции ядра для наборов модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ и $\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\}$. Алгоритм на основе использования минимальной функции ядра для набора модулей $\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\}$ в среднем на 20% быстрее КТО, на 23.4% быстрее приближенной КТО и на 17.6% быстрее функции Пирло.

Проведено моделирование методов сравнения чисел в СОК. Разработанный метод построения функций ядра Акушского для операции сравнения чисел в СОК позволил увеличить скорость сравнения в среднем на 15.1% по сравнению с классическими методами.

В третьей главе разработан программный комплекс для выполнения модульных и немодульных операций в СОК. Предложенный комплекс может быть использован в облачных и туманных структурах, работающих в условиях ограниченных вычислительных ресурсов. К направлениям внедрения результатов исследования можно отнести криптографию и искусственные нейронные сети.

Полный объем диссертации составляет 167 страниц, включая 18 рисунков и 42 таблицы. Список литературы содержит 119 наименований.

Глава 1. Исследование методов повышения скорости обработки данных Интернета вещей

В последние годы наблюдается тенденция к увеличению числа людей, переезжающих в города. По прогнозам ООН к 2030 году более 60% населения будет проживать в городской среде. Развитие умных городов, основанных на информационных технологиях, становится необходимостью для управления муниципалитетом и повышения качества жизни горожан. Умный город объединяет различные информационные системы для управления городской инфраструктурой: транспортом, промышленостью, образованием, здравоохранением, системами ЖКХ, безопасностью и финансами.

Для решения проблем, связанных с увеличением населения, существует потребность в интеграции и анализе информации из различных городских систем, что позволит предоставить более эффективные услуги. Эти данные собираются из различных источников, таких как беспроводные сенсорные сети и устройства IoT, установленные в зданиях, на улицах, на заводах, в транспортных средствах и так далее. Генерация больших данных и использование новых способов их сбора и анализа позволяют обрабатывать информацию в режиме реального времени, что улучшает качество и скорость принятия решений.

Как правило, обработка данных осуществляется с использованием облачных архитектур, однако их централизованная концепция накладывает ограничения, связанные с задержками, пропускной способностью сети и зависимостью от интернет-соединения. Внедрение туманных вычислений позволяет устранить эти недостатки, перемещая вычисления, хранение данных и принятие решений ближе к границе сети.

С развитием туманных вычислений, которые переносят обработку данных ближе к их источнику, возникает ряд проблем, связанных с безопасностью и защитой информации. В отличие от централизованных облачных систем, где безопасность обеспечивается за счет мощных защитных механизмов дата-центров, в распределенной среде туманных вычислений уязвимостей значительно больше.

1.1 Архитектура распределенной обработки данных

IoT — это концепция, при которой физические устройства, оснащенные сенсорами, программным обеспечением и сетевыми модулями, соединяются между собой через интернет для обмена данными и выполнения задач без участия человека. IoT применяется для автоматизации, мониторинга и управления устройствами в самых разных сферах, таких как: умный дом, промышленность, здравоохранение, транспорт и логистика, сельское хозяйство, энергетика [74].

IoT открывает перед обществом множество возможностей, но также создает серьезные угрозы безопасности [36]. Вот некоторые из основных рисков:

1. *Уязвимости в устройствах:* многие IoT-устройства имеют слабую защиту, устаревшие протоколы и редко обновляются, что делает их мишенью для хакеров [27].
2. *Недостаточная защита данных:* IoT-устройства собирают и передают огромные объемы данных, включая персональную информацию, которая может быть перехвачена злоумышленниками.
3. *Отсутствие стандартов безопасности:* разные производители применяют разные уровни защиты, что усложняет комплексную безопасность IoT-систем.
4. *Шпионские угрозы:* некоторые устройства могут использоваться для скрытого наблюдения за пользователями, собирая данные без их ведома.
5. *Компрометация критически важных систем:* IoT внедряется в промышленность, медицину, транспорт и другие критические сферы, что создает угрозу кибератак с серьезными последствиями [88].

Несмотря на значительный прогресс в области защиты данных в распределенных системах, таких как туманные, граничные и облачные вычисления, обеспечение безопасности в динамически изменяющихся условиях остается одной из самых сложных задач.

Для обеспечения безопасности и устойчивости в системе IoT рассмотрена архитектура, состоящая из пяти уровней (рис. 1.1).

Уровень 1. Устройства и сенсоры. Этот уровень включает в себя датчики, сенсоры и встроенные устройства, которые отличаются крайне низким энергопотреблением и ограниченными возможностями защиты данных. Такие

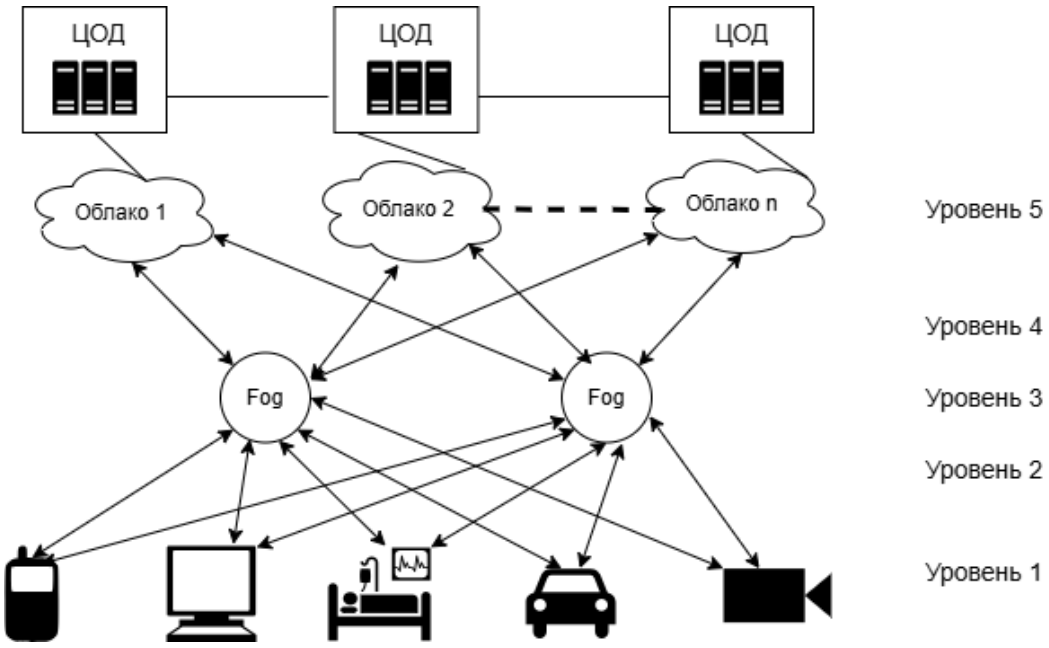

Рисунок 1.1 — Архитектура работы IoT

устройства широко используются в умных домах, носимой электронике, промышленных системах и других IoT-решениях [27]. Основная проблема на этом уровне — обеспечение безопасности при минимальных ресурсах, так как большинство устройств работают на батареях и не могут поддерживать сложные механизмы шифрования или защиты.

Уровень 2. Беспроводные сети передачи данных. На этом уровне данные передаются через беспроводные сети с низким энергопотреблением, такие как LoRaWAN, Zigbee [67]. Однако из-за ограниченной мощности передачи и высокой степени неопределенности в таких сетях возникают сложности с обеспечением конфиденциальности и целостности данных. Это делает их уязвимыми для атак, таких как перехват данных или подмена устройств.

Уровень 3. Туманные вычисления. Этот уровень представляет собой распределенную вычислительную инфраструктуру, которая объединяет миллионы устройств, расположенных ближе к источнику данных [34]. Туманные вычисления позволяют обрабатывать данные на границе сети, что снижает задержки и повышает скорость работы приложений. Однако управление такими гетерогенными системами, обеспечение их безопасности и координация между устройствами остаются сложными задачами.

Уровень 4. Глобальные сети. Этот уровень обеспечивает высокую надежность и безопасность передачи данных. Однако даже здесь возникают риски, связанные с масштабируемостью и увеличением нагрузки из-за роста числа подключенных устройств [58].

Уровень 5. Облачные и распределенные центры обработки данных. На этом уровне данные хранятся и обрабатываются в распределенных центрах, которые обеспечивают динамическое масштабирование и высокую отказоустойчивость. Основные задачи здесь — защита от DDoS-атак, минимизация утечек данных и обеспечение конфиденциальности информации [103].

Учитывая описанные выше риски и архитектуру IoT, становится очевидным, что обеспечение безопасности данных в таких системах требует применения современных алгоритмов шифрования. Однако, в отличие от традиционных вычислительных систем, IoT-устройства имеют ограниченные ресурсы (энергия, вычислительная мощность, память), что накладывает дополнительные требования на выбор криптографических методов. Рассмотрим, как алгоритмы шифрования могут быть применены на каждом уровне архитектуры IoT для обеспечения безопасности данных.

На уровне устройств и сенсоров, где ресурсы крайне ограничены, применяются легковесные алгоритмы шифрования. Например, AES-128 (Advanced Encryption Standard) [101] с оптимизированными раундами или специализированные алгоритмы, такие как PRESENT, SPECK и SIMON [112]. Эти методы обеспечивают достаточную защиту при минимальных затратах энергии и вычислительной мощности. Для аутентификации устройств и обеспечения целостности данных используются легковесные хэш-функции, такие как PHOTON или SPONGENT [29].

На уровне беспроводных сетей передачи данных важно защитить данные от перехвата и подмены устройств [42]. Здесь применяются как симметричные, так и асимметричные алгоритмы. Например, AES-128 или AES-256 для шифрования данных и эллиптическая криптография (Elliptic Curve Cryptography, ECC) для обмена ключами. Протоколы, такие как DTLS (Datagram Transport Layer Security), обеспечивают шифрование и аутентификацию [75].

На уровне туманных вычислений, где данные обрабатываются ближе к источнику, используются более сложные алгоритмы, но с учетом ограниченных ресурсов. Здесь часто применяются гибридные схемы: симметричные алгоритмы для шифрования данных и асимметричные для обмена ключами. Протоколы, такие как TLS (Transport Layer Security) [114], защищают данные при передаче между узлами, а OAuth 2.0 или JWT (JSON Web Tokens) обеспечивают аутентификацию и авторизацию [105].

На уровне глобальных сетей, где данные передаются через интернет, используются стандартные алгоритмы шифрования, такие как TLS/SSL и AES-256, для защиты данных от перехвата и атак. Асимметричные алгоритмы, такие как RSA или ECC, применяются для обмена ключами и аутентификации устройств [106]. Важным аспектом является масштабируемость, так как количество подключенных устройств постоянно растет.

На уровне облачных и распределенных центров обработки данных применяются наиболее мощные алгоритмы шифрования. AES-256 используется для шифрования данных на стороне сервера, а RSA или ECC — для управления ключами [106]. Для дополнительной защиты данных в облаке могут применяться методы гомоморфного шифрования, которые позволяют выполнять операции с зашифрованными данными без их дешифровки.

Таким образом, безопасность IoT-систем требует комплексного подхода, включающего использование различных алгоритмов шифрования на каждом уровне [57]. От легковесных методов для устройств с ограниченными ресурсами до мощных алгоритмов для облачных сервисов — каждый уровень архитектуры IoT требует своих решений для защиты данных.

1.2 Асимметричные алгоритмы шифрования для IoT

1.2.1 Алгоритм RSA

Алгоритм RSA (Rivest-Shamir-Adleman) [98] является одним из наиболее широко используемых методов асимметричного шифрования, который может быть эффективно применен в IoT для обеспечения конфиденциальности и целостности данных.

В основе RSA лежат следующие математические принципы. Пусть выбраны два различных больших простых числа p и q. Вычисляется их произведение $n = pq$, которое будет модулем как для открытого, так и для закрытого ключей. Функция Эйлера от числа n определяется как $\varphi(n) = (p-1)(q-1)$, поскольку p и q — простые. Затем выбирается целое число e, такое что $1 < e < \varphi(n)$ и $\gcd(e, \varphi(n)) = 1$, то есть e взаимно просто с $\varphi(n)$. Это число используется

как открытая экспонента. Далее вычисляется число d — мультипликативная обратная к e по модулю $\varphi(n)$, то есть такое, что $ed \equiv 1 \mod \varphi(n)$. Значение d образует закрытую экспоненту.

Пара (n, e) составляет открытый ключ, а (n, d) — закрытый. Шифрование сообщения M, представленного в виде числа, производится по формуле

$$C \equiv M^e \mod n,$$

а дешифрование — по формуле

$$M \equiv C^d \mod n.$$

Корректность дешифрования обеспечивается малой теоремой Ферма и свойствами модулярной арифметики: $(M^e)^d \equiv M^{ed} \equiv M \mod n$.

В таблице 1 представлены основные операции используемые в алгоритме RSA.

Таблица 1 — Операции в алгоритме RSA

Этап	Описание	Операция
Генерация ключей	Выбор двух больших простых чисел p и q	Выбор параметоров
	Вычисление $n = p \cdot q$	Умножение
	Вычисление функции Эйлера $\varphi(n) = (p-1)(q-1)$	Вычитание, умножение
	Выбор e, такого что $1 < e < \varphi(n)$ и $\gcd(e, \varphi(n)) = 1$	Алгоритм Евклида
	Вычисление d, такого что $d \cdot e \equiv 1 \mod \varphi(n)$	Расширенный алгоритм Евклида
Шифрование	Зашифрованное сообщение: $C = M^e \mod n$	Возведение в степень по модулю
Дешифрование	Дешифрованное сообщение: $M = C^d \mod n$	Возведение в степень по модулю

Рассмотрим пример работы алгоритма RSA.

Пример 1.2.1. *Пусть* $p = 61$, $q = 53$, *тогда* $n = 3233$, $\varphi(n) = (61-1)(53-1) = 3120$. *Выберем* $e = 17$, *так как* $\gcd(17, 3120) = 1$. *Теперь найдем* d, *обратное*

к e по модулю 3120, что дает $d = 2753$. *Таким образом, открытый ключ —* $(3233, 17)$, *закрытый —* $(3233, 2753)$.

Пусть зашифровывается сообщение $M = 123$. *Тогда шифротекст:*

$$C = 123^{17} \mod 3233 = 855.$$

А дешифровка даст:

$$M = 855^{2753} \mod 3233 = 123,$$

что подтверждает корректность алгоритма.

На практике используются числа длиной в сотни и тысячи битов, чтобы обеспечить криптостойкость, однако принцип остается неизменным. RSA служит основой для безопасной передачи ключей, цифровых подписей и иных протоколов защиты информации. В контексте IoT, где устройства часто обладают ограниченными вычислительными ресурсами, реализация RSA требует оптимизации для снижения нагрузки на процессор и энергопотребления. Современные подходы включают использование легковесных библиотек [38] и аппаратных ускорителей [39] для выполнения операций модулярной арифметики, таких как возведение в степень по модулю. Кроме того, для IoT-устройств рекомендуется применение RSA с меньшими длинами ключей (например, 1024 или 2048 бит), что позволяет сохранить баланс между безопасностью и производительностью.

В статье [53] используют туманные вычисления для управления медицинскими данными пациентов и цифровой идентификацией, которая обеспечивает безопасное хранение и управление медицинскими записями пациентов с использованием алгоритма RSA. Результаты оценки показывают, что предложенная технология обрабатывает данные для 100 пользователей за 631 секунду. Предложенный подход обеспечивает повышенную безопасность, конфиденциальность и эффективность в управлении медицинскими данными.

Алгоритм RSA остается широко принятым и проверенным временем алгоритмом с доказанной безопасностью, однако с учетом больших размеров ключей и ростом вычислительных мощностей можно использовать более устойчивые алгоритмы эллиптической криптографии. В статье [22] проведен анализ двух широко используемых алгоритмов шифрования ECC и RSA, с акцентом на их применение в облачных и туманных средах. В исследовании сравнивается размер ключей и уровень безопасности ECC и RSA, а также оценивается

их пригодность для использования в ресурсоограниченных туманных средах. Показано что ECC, обеспечивает тот же уровень безопасности, что и RSA, но использует меньшие размеры ключей. В статье [106] представлена высокозащищенная и энергоэффективная архитектура туманных вычислений с поддержкой ECC и алгоритма RSA. Проведено сравнение ECC и RSA по энергопотреблению и пропускной способности для туманного шлюза. Показано, что ECC демонстрирует превосходство над RSA как по энергопотреблению, так и по пропускной способности на всех уровнях безопасности.

1.2.2 Эллиптическая криптография

В 1985 году два математика, Нил Коблиц и Виктор Миллер, независимо друг от друга предложили использовать свойства эллиптических кривых для создания криптографических систем. Это открытие положило начало новому направлению в криптографии, получившему название «криптография на эллиптических кривых». Основой ECC является операция скалярного умножения точки на эллиптической кривой на целое число, которая реализуется через последовательные операции сложения и удвоения точек. Эти операции, в свою очередь, выполняются с использованием арифметики конечных полей, включающей сложение, умножение и инвертирование.

Интерес к криптографии на эллиптических кривых обусловлен ее уникальными преимуществами, такими как достаточно высокая скорость вычислений и компактность ключей. Это делает ECC особенно востребованной в системах с ограниченными ресурсами, таких как IoT-устройства. В отличие от традиционных асимметричных криптосистем, таких как RSA, которые полагаются на сложность факторизации больших чисел, ECC основывается на сложности задачи дискретного логарифмирования в группах точек эллиптической кривой.

В криптографии параметры и константы, используемые в уравнениях эллиптических кривых, тесно связаны с элементами конечного поля, что обеспечивает выполнение свойств абелевой группы [20]. Математическое описание эллиптической кривой основано на уравнении Вейерштрасса. Эллиптическая кривая над конечным простым полем $\mathbb{F}_p$, где p — простое число, задается сле-

дующим образом:

$$E : y^2 \equiv x^3 + ax + b \pmod{p}, \quad \text{где } 4a^3 + 27b^2 \not\equiv 0 \pmod{p}. \tag{1.1}$$

Условие $4a^3 + 27b^2 \not\equiv 0 \mod p$ гарантирует, что кривая не содержит сингулярностей и допускает корректные определения групповых операций над точками.

Множество точек $(x, y) \in \mathbb{F}_p^2$, удовлетворяющих уравнению (1.1), вместе со специальной точкой $\mathcal{O}$ (неподвижной точкой, аналогичной нулю в сложении) образуют конечную абелеву группу по операции сложения точек на кривой.

Основной криптографической операцией в ECC является скалярное умножение точки на число:

$$Q = dG,$$

где d — закрытый ключ, G — базовая (генерирующая) точка, а Q — соответствующий открытый ключ.

Операцию вычисления Q по заданным d и G можно реализовать эффективно, в то время как обратная задача — определение d по известным Q и G — считается вычислительно сложной. Эта проблема известна как задача дискретного логарифма на эллиптической кривой и составляет основу криптостойкости алгоритма ECC.

В таблице 2 представлены основные операции, используемые в алгоритме ECC.

Рассмотрим пример работы алгоритма ECC.

Пример 1.2.2. *Пусть эллиптическая кривая задана над полем* $\mathbb{F}_{17}$ *уравнением:*

$$E : y^2 \equiv x^3 + 2x + 2 \mod 17.$$

Базовая точка $G = (5, 1)$ *ее порядок равен* $n = 19$*. Закрытый ключ пользователя:* $d = 7$*. Тогда открытый ключ:*

$$Q = dG = 7 \cdot (5, 1) = (6, 3).$$

Для шифрования сообщения отправитель выбирает случайное число $k = 3$ *и вычисляет:*

$$C_1 = 3G = (10, 6), \quad S = 3Q = 3 \cdot (6, 3) = (13, 7).$$

Таблица 2 — Операции в алгоритме ECC

Этап	Описание	Операция
Генерация ключей	Выбор эллиптической кривой $E : y^2 = x^3 + ax + b \mod p$ и базовой точки G	Определение параметров кривой
	Выбор закрытого ключа $d \in [1, n-1]$	Случайный выбор числа
	Вычисление открытого ключа $Q = d \cdot G$	Скалярное произведение
Шифрование	Генерация случайного k, вычисление $C_1 = kG$	Скалярное произведение
	Вычисление общего секрета $S = kQ$, получение ключа шифрования, шифрование сообщения	Скалярное произведение
Дешифрование	Вычисление $S = d \cdot C_1$	Скалярное произведение
	Дешифрование сообщения с использованием ключа из S	Симметричное дешифрование

Пусть сообщение M представлено в виде числа и шифруется при помощи сложения по модулю два ($\oplus$) с x-координатой общей точки S. Пусть $M = 9$, тогда шифротекст:

$$C_2 = M \oplus 13 = 9 \oplus 13 = 4.$$

Получатель, имея $C_1 = (10, 6)$, вычисляет:

$$S = d \cdot C_1 = 7 \cdot (10, 6) = (13, 7),$$

и восстанавливает сообщение:

$$M = C_2 \oplus 13 = 4 \oplus 13 = 9.$$

Эллиптическая криптография привлекает за счет своей стойкости при малом размере ключей. Например, 256-битный ECC-ключ обеспечивает уровень безопасности, сопоставимый с 3072-битным RSA, что критично в условиях ограниченных вычислительных ресурсов.

ECC активно применяется в туманных вычислениях для обеспечения безопасности, аутентификации и защиты данных. В [30] рассматривается протокол SAKA-FC, который использует ECC для трехфакторной аутентификации с сохранением конфиденциальности удаленных пользователей. ECC применяется в сочетании с хеш-функциями, симметричными полиномиальными функциями для обеспечения безопасности. В [107] предложен метод который сочетает ECC и ДНК-кодирование для повышения безопасности данных в туманных вычислениях. ECC используется для шифрования данных, собранных кластерными узлами, что делает его легковесным и эффективным решением для IoT-устройств с ограниченными ресурсами. В статье [90] предложен подход биометрической криптографии, где ECC используется для аутентификации в туманных вычислениях.

1.3 Повышение производительности алгоритмов шифрования в туманных вычислениях

Современные исследования в области криптографии для распределенных вычислительных систем, включая туманные вычисления, демонстрируют устойчивую тенденцию к поиску методов ускорения алгоритмов шифрования. Это обусловлено возрастающими требованиями к производительности в условиях ограниченных ресурсов туманных узлов, которые, будучи промежуточным звеном между облачными вычислениями и конечными устройствами IoT, должны обеспечивать как высокую скорость криптографических операций, так и энергетическую эффективность. Анализ научных публикаций [21; 48; 51; 77; 81] показывает, что существующие подходы к оптимизации производительности криптографических алгоритмов можно условно разделить на три основные категории: аппаратные ускорители, программная оптимизация и гибридные методы.

Основным направлением в настоящее время остается аппаратное ускорение, что подтверждается многочисленными исследованиями, посвященными использованию специализированных интегральных схем, таких как FPGA (Field-Programmable Gate Arrays) и ASIC (Application-Specific Integrated Circuits), для выполнения ресурсоемких операций модульной арифметики

[23; 109]. В работе [26] продемонстрировано, что применение FPGA позволяет достичь ускорения алгоритма RSA в 6-9 раз по сравнению с реализацией на центральном процессоре (CPU). Аналогичные результаты получены для операций скалярного умножения на эллиптических кривых [54]. Однако, несмотря на впечатляющие показатели производительности, аппаратные решения имеют ряд существенных ограничений, включая высокую стоимость разработки и внедрения, повышенное энергопотребление, а также недостаточную гибкость при необходимости модификации криптографических алгоритмов [43]. Не все устройства на периферии сети поддерживают аппаратное ускорение. Например, туманные узлы, представленные в виде недорогих микрокомпьютеров, микроконтроллеров или мобильных устройств, не всегда имеют возможность интеграции криптографических сопроцессоров. Эти факторы делают FPGA и ASIC малопригодными для массового развертывания в гетерогенных туманных средах, где критически важными являются масштабируемость и адаптивность.

Альтернативным подходом к использованию интегральных схем является использование графических процессоров (GPU) для параллелизации криптографических вычислений. Исследования [48] показывают, что современные GPU, благодаря их архитектуре, могут эффективно ускорять как RSA, так и ECC. В частности, в работе [65] демонстрируется ускорение RSA в 1197.5 раз в среднем при использовании CUDA-реализации по сравнению с алгоритмом на базе CPU. Однако GPU-ориентированные решения также обладают существенными недостатками, главными из которых являются высокая стоимость и высокое энергопотребление, что критично для автономных туманных узлов, а также необходимость переноса данных между CPU и GPU, что создает дополнительные задержки в распределенных системах [51].

Второе направление исследований связано с программной реализацией, не требующей специализированного аппаратного обеспечения. Сюда относятся: модификации самих криптографических алгоритмов [81], применение эффективных методов модульной арифметики [9; 77], оптимизация на уровне программного кода [25]. Эти методы, безусловно, обеспечивают определенный прирост производительности, но их потенциал ограничен фундаментальной сложностью самих криптографических алгоритмов, особенно при работе с длинной арифметикой [25].

Третье, гибридное направление, пытается объединить преимущества аппаратного и программного подходов. Примером может служить использование

инструкций Intel AVX-512 для ускорения модульной арифметики [47] или применение тензорных ядер в современных GPU для матричных операций в криптографии на решетках [111].

Таким образом, анализ современного состояния исследований выявляет следующую проблему: существующие методы ускорения либо требуют дорогостоящего и энергозатратного аппаратного обеспечения, либо предлагают лишь умеренное улучшение производительности за счет программной оптимизации. Эта проблема может быть решена за счет применения иного математического аппарата — системы остаточных классов, которая, благодаря своим свойствам параллелизма и отсутствия межразрядных переносов, открывает возможности для ускорения криптографических вычислений без существенного увеличения аппаратных затрат, что соответствует распределенной природе туманных вычислений.

1.4 Применение системы остаточных классов для повышения скорости алгоритмов шифрования в туманных вычислениях

1.4.1 Математические основы системы остаточных классов

Определение 1.4.1. *Базисом системы остаточных классов называется множество модулей* $\{p_1, p_2, ..., p_n\}$*, где каждый модуль* $p_i \geqslant 2$ $(i = 1, 2, \ldots, n)$ *и* $\gcd(p_i, p_j) = 1$*. Здесь и далее по тексту* gcd *обозначает наибольший общий делитель (англ. greatest common divisor, GCD). По умолчанию модули упорядочены по возрастанию, то есть* $p_1 < p_2 < \cdots < p_n$*.*

Определение 1.4.2. *Произведение модулей* $P = \prod_{i=1}^{n} p_i$ *называется динамическим диапазоном системы остаточных классов.*

Определение 1.4.3. *Целое число* $X \in [0, P)$ *представимо в виде* n*-мерного вектора, составленного из наименьших неотрицательных остатков от деления соответствующего числа на* p_i*:*

$$X = (x_1, x_2, \ldots, x_n),$$

где $x_i \equiv X \mod p_i$, *что также обозначается как* $x_i = |X|_{p_i}$.

Для введения отрицательных чисел необходимо разделить диапазон на два интервала. Тогда при введении отрицательных чисел число X удовлетворяет следующему соотношению:

$$\begin{cases} -\frac{P-1}{2} \leqslant X \leqslant \frac{P-1}{2}, & \text{если } P \equiv 1 \mod 2, \\ \frac{-P}{2} \leqslant X \leqslant \frac{P}{2} - 1, & \text{если } P \equiv 0 \mod 2. \end{cases}$$

Рассмотрим СОК с базисом $\{3, 4\}$. В этом базисе можно взаимно-однозначно представить числа из полуинтервала $[-6, 6)$, так как $P = 3 \cdot 4 = 12$. Если $X = (x_1, x_2, \ldots, x_n)$, то отрицательное число $-X = (\overline{x_1}, \overline{x_2}, \ldots, \overline{x_n})$, где $\overline{x_i}$ является дополнением x_i до модуля p_i. Для СОК $\{3, 4\}$ и числа $X = (1, 1)$ получим $-X = (3 - 1, 4 - 1) = (2, 3)$.

В таблице 3 представлены соответствия чисел из позиционной системы счисления и системы остаточных классов.

Таблица 3 — Представление чисел для СОК с базисом $\{3, 4\}$

$-6 \xrightarrow{\text{СОК}} (0, 2)$	$-5 \xrightarrow{\text{СОК}} (1, 3)$	$-4 \xrightarrow{\text{СОК}} (2, 0)$	$-3 \xrightarrow{\text{СОК}} (0, 1)$
$-2 \xrightarrow{\text{СОК}} (1, 2)$	$-1 \xrightarrow{\text{СОК}} (2, 3)$	$0 \xrightarrow{\text{СОК}} (0, 0)$	$1 \xrightarrow{\text{СОК}} (1, 1)$
$2 \xrightarrow{\text{СОК}} (2, 2)$	$3 \xrightarrow{\text{СОК}} (0, 3)$	$4 \xrightarrow{\text{СОК}} (1, 0)$	$5 \xrightarrow{\text{СОК}} (2, 1)$

Система остаточных классов обладает особенностью выполнять операции сложения, вычитания и умножения параллельно и независимо для каждого модуля. Для чисел $X = (x_1, x_2, \ldots, x_n)$ и $Y = (y_1, y_2, \ldots, y_n)$ выразим арифметические операции следующим образом

$$X * Y = (x_1 * y_1, x_2 * y_2, \ldots, x_n * y_n), \text{где } * \in \{+, -, \times\}. \tag{1.2}$$

Запишем процесс выполнения модульных операция в СОК в виде алгоритма 1.

В Примере 1.4.1 представлено сложение двух чисел в СОК.

Пример 1.4.1. *Сложим два числа* $X = -2$ *и* $Y = 3$ *в базисе* $\{3, 4\}$. *Их представление в заданном базисе указано в таблице 3. Воспользуемся уравнением* (1.2) *для сложения:*

$$X + Y = (|1 + 0|_3, |2 + 3|_4) = (1, 1).$$

Алгоритм 1: Выполнение модульных операций в СОК

Input: $X \xrightarrow{\text{СОК}} (x_1, x_2, \ldots, x_n)$, $Y \xrightarrow{\text{СОК}} (y_1, y_2, \ldots, y_n)$, $* \in \{+, -, \times\}$

Data: $\{p_1, p_2, \ldots, p_n\}$

Output: $R = (r_1, r_2, \ldots, r_n)$

1 **for** $i = 1$, $i \leqslant n$, $i++$ **do**

2 $\quad r_i = |x_i * y_i|_{p_i}$

Result: R.

Этот пример наглядно демонстрирует преимущества модулярной арифметики. Рассматриваемый диапазон позволяет работать с числами до 4 бит. Однако сами модули 3 и 4 требуют 3 бита для кодирования, что позволяет параллельно выполнять вычисления меньшей размерности по каждому модулю. Особенно это свойство проявляется в криптографии: вместо обработки 2048-битных чисел, как в современных алгоритмах шифрования, можно использовать 33 параллельных канала СОК с модулями длиной 32 бита [35; 63].

Первоочередная задача при задании СОК — определить набор модулей системы или же другими словами базис системы. Начиная с пятидесятых годов двадцатого века было предложено множество различных специальных наборов модулей [4]. Для классических систем эффективным считается набор модулей $\{2^n - 1, 2^n, 2^n + 1\}$, который может значительно снизить сложность операций, описанных ранее. В таблице 6 параграфа 3.2.1 представлены наиболее распространенные наборы модулей специального вида и их характеристики.

Одним из фундаментальных свойств арифметики в системе остаточных классов является отсутствие необходимости в учете межразрядных переносов при выполнении операций сложения и умножения [108]. В отличие от традиционных позиционных систем счисления, где переносы могут распространяться на несколько разрядов, в СОК результат каждой операции остается строго в пределах диапазона, определяемого выбранным модулем. Это свойство существенно снижает вычислительную сложность данных операций, обеспечивая их высокую эффективность.

Важным следствием данного подхода является возможность оптимизации вычислительных процессов, включающих последовательные модульные арифметические операции. При этом можно выбирать базис СОК, таким образом, чтобы конечные результаты всегда находились в допустимом диапазоне зна-

чений для любых входных операндов. Таким образом, СОК предоставляет преимущества в задачах, требующих параллельной обработки данных.

1.4.2 Китайская теорема об остатках

Китайская теорема об остатках позволяет восстановить число в СОК по его остаткам [115].

Теорема 1.4.1. *Пусть даны натуральные попарно взаимно простые модули* $p_1, p_2, \ldots, p_n$ *и* $P = \prod_{i=1}^{n} p_i$. *Тогда существует единственное число* X *в диапазоне* $0 \leqslant X < P$, *которое может быть однозначно представлено в виде последовательности* $(x_1, x_2, \ldots, x_n)$, *где* $x_i = X \bmod p_i$, *при этом*

$$X = \left| \sum_{i=1}^{n} x_i \cdot P_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_P, \tag{1.3}$$

где $P_i = \frac{P}{p_i}$, $\left| P_i^{-1} \right|_{p_i}$ — *обратный элемент* P_i *по модулю* p_i.

Определение 1.4.4. *Значения* $B_i = \left| P_i^{-1} \right|_{p_i} \cdot P_i$ *называются ортогональными базисами системы остаточных классов.*

Запишем реализацию метода обратного преобразования на основе КТО в виде Алгоритма 2.

Пример 1.4.2. *Возьмем СОК с набором модулей* $\{3, 4\}$. *Рассмотрим процесс перевода числа* $X = (2, 1)$ *в позиционную систему счисления.*

Найдем значения P_i *и* $\left| P_i^{-1} \right|_{p_i}$:

$$P_1 = \frac{P}{p_1} = 4, \quad P_2 = \frac{P}{p_2} = 3.$$
$$\left| P_1^{-1} \right|_{p_1} = 1, \quad \left| P_2^{-1} \right|_{p_2} = 3.$$

По формуле 1.3 найдем X

$$X = |4 \cdot 1 \cdot 2 + 3 \cdot 3 \cdot 1|_{12} = 5.$$

Таким образом, получено число в позиционной системе.

Алгоритм 2: Обратное преобразование из СОК в ПСС на основе КТО

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,
$P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$B_i = \left|P_i^{-1}\right|_{p_i} \cdot P_i, i = 1, 2, \ldots, n$

Output: X

1 $sum = 0$

2 **for** $i = 1$, $i \leqslant n$, $i++$ **do**

3 $\quad sum = sum + x_i \cdot B_i$

4 $X = sum \mod P$

Result: X.

Входное преобразование и основной вычислительный блок работают независимо и полностью параллельно, что является одним из главных преимуществ СОК по сравнению с традиционными позиционными системами счисления. Однако выходное преобразование, которое собирает данные для восстановления итогового результата, не может быть выполнено параллельно. Это делает его критическим местом, определяющим общую эффективность системы, работающей в СОК.

Тем не менее, в задачах, где основная часть вычислений выполняется в СОК, а преобразование результата требуется лишь один раз в конце (например, в криптографических алгоритмах), эта проблема не является критичной. Однако в некоторых случаях, таких как расширение базиса системы остаточных классов, используемое в криптографии, процесс обратного преобразования становится неотъемлемой частью вычислений.

1.4.3 Проблемы арифметики системы остаточных классов

Несмотря на преимущества системы остаточных классов в некоторых аспектах, она имеет ряд существенных ограничений в других. Например, для чисел $(2, 3)$ и $(1, 0)$ из таблицы 3, невозможно сразу определить, какое из них больше, что требует использования специальных методов для нахожде-

ния позиционной характеристики (ПХ), что может быть крайне ресурсоемким процессом, особенно при использовании методов, подобных Китайской теореме об остатках. Поэтому при разработке приложений, полностью основанных на СОК, следует минимизировать использование условных операций, которые зависят от сравнения чисел. Аналогичные сложности возникают при выполнении операций определения знака числа или деления [95; 110]. Однако есть исключения: в случае если известно, что число делиться нацело, то операцию деления можно свести к умножению на мультипликативную инверсию делителя, что делает ее эффективной. Это свойство активно используется в криптографических системах с открытым ключом.

Кроме того, преобразование данных в СОК и обратно неизбежно при взаимодействии с системами, не использующими модулярную арифметику. Однако из-за высокой вычислительной сложности таких преобразований рекомендуется минимизировать их количество. Идеальным подходом является проектирование системы, в которой данные преобразуются в СОК только один раз на входе, все вычисления выполняются в СОК, а обратное преобразование происходит только на выходе, когда результат должен быть передан в систему, не поддерживающую СОК. Если же преобразования неизбежны (например, при расширении базиса СОК, что необходимо для умножения Монтгомери), важно тщательно оценить вычислительные затраты и выбрать оптимальный набор модулей, чтобы достичь баланса между производительностью и сложностью реализации [99].

Таким образом, немодульные операции, такие как обратное преобразование числа, общее деление, определение четности (при отсутствии четного модуля), сравнение чисел, определение знака числа являются вычислительно сложными и требуют большого количества вычислений, что ограничивает применение СОК и ее использование для быстрых вычислений.

1.5 Выводы по первой главе

В первой главе описана архитектура распределенной обработки данных, которая лежит в основе IoT, включая особенности передачи и обработки данных в распределенных системах и требования к их безопасности. Рассмотрены асимметричные алгоритмы шифрования, и их применение для IoT. Исследованы

методы повышения производительности алгоритмов шифрования в туманных вычислениях.

Исследована возможность применения СОК для повышения скорости алгоритмов шифрования в туманных вычислениях. Рассмотрены проблемы арифметики системы остаточных классов. Это позволило сформулировать научную задачу исследования: исследование и разработка математических методов и алгоритмов выполнения вычислительно сложных операций СОК на основе функции ядра Акушского, способных повысить скорость при выполнении алгоритмов шифрования вычислительными узлами туманной среды, работающими в системе остаточных классов.

Во второй главе исследуется функция ядра Акушского, которая позволяет определить позиционные свойства чисел в системе остаточных классов. Будет рассмотрена проблема критических ядер функции ядра и предложены алгоритмы их поиска. На основе функции ядра Акушского будут предложены эффективные методы и алгоритмы ускорения немодульных операций, включая преобразование из СОК в позиционную систему счисления, выполнение общего деления, определение знака числа и сравнения чисел.

Глава 2. Разработка методов и алгоритмов повышения скорости немодульных операций с использованием функции ядра Акушского

2.1 Определение позиционной характеристики на основе функции ядра Акушского

Исследования, проведенные Акушским И.Я., Бурцевым В.М. и Паком И.Т. [1], были направлены на нахождение позиционной характеристики числа в СОК. В результате этих исследований была предложена новая математическая конструкция, получившая название функции ядра Акушского. Функция ядра Акушского определяется следующей формулой:

$$C\left(X\right) = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{p_i} \right\rfloor = \sum_{i=1}^{n} \left(X - x_i\right) \cdot \frac{w_i}{p_i}, \tag{2.1}$$

где целые числа w_i выступают в роли постоянных величин, которые определяются выбором точки для интерполяции. Константы w_i задают вес каждого из частных $\left\lfloor \frac{X}{p_i} \right\rfloor$ в формуле (2.1), тем самым задавая функцию ядра и придавая ей различные свойства.

Подставив $X = P$ в (2.1) получим:

$$C\left(P\right) = C_P = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{P}{p_i} \right\rfloor = \sum_{i=1}^{n} w_i \cdot P_i. \tag{2.2}$$

Разделив $C\left(P\right)$ на P получим:

$$\frac{C\left(P\right)}{P} = \sum_{i=1}^{n} \frac{w_i}{p_i}. \tag{2.3}$$

Отсюда следует, что для получения малых значений $C(P)$ необходимо, чтобы некоторые значения w_i были отрицательными. Подставив (2.3) в (2.1), получим:

$$C\left(X\right) = X \cdot \sum_{i=1}^{n} \frac{w_i}{p_i} - \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i} = X \cdot \frac{C\left(P\right)}{P} - \sum_{i=1}^{n} \frac{x_i \cdot w_i}{p_i}. \tag{2.4}$$

Используя функцию ядра можно получить информацию о позиционной характеристики числа как показано на рисунке 2.1. Функция ядра позволяет

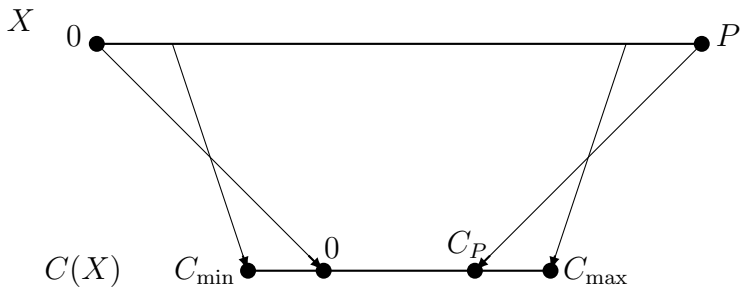

Рисунок 2.1 — Отображение $0 \leqslant X < P$ на $C_{min} \leqslant C(X) \leqslant C_{max}$

отобразить число в СОК на координатную прямую, где C_{min} и C_{max} минимальное и максимальное значение функции ядра для заданного набора весов.

Функция ядра Акушского является нелинейной функцией. Величина нелинейности определяется величиной весов, в свою очередь связанной с конкретным значением $C(P)$ для заданного набора модулей СОК.

Учитывая, что $P_j \equiv 0 \mod p_i$ для любого $i \neq j$, веса w_i функции $C(X)$ могут быть определены соотношением:

$$w_i \equiv \left| C(P) \cdot P_i^{-1} \right|_{p_i}. \tag{2.5}$$

Таким образом, веса могут быть определены после выбора $C(P)$, но с условием выполнения (2.2).

Пример 2.1.1. *Рассмотрим систему с модулями* $p_1 = 3$, $p_2 = 4$ *и весами* $w_1 = 1$, $w_2 = 2$. *Динамический диапазон* $P = 12$. *Функция ядра от динамического диапазона равна* $C(P) = 10$. *Для данных параметров график функции ядра изображена на рисунке 2.2.*

Формула (2.1) имеет ограниченную применимость в практических вычислениях, поскольку требует знания позиционного представления числа X. Данную проблему можно решить путем применения Китайской теоремы об остатках к формуле (2.1). Тогда значение функции ядра $C(X)$ с весами $w_1, w_2, \ldots, w_n$ может быть определено на основе остатков числа X. При соблюдении условий $0 \leqslant C(X) < C_P$ для $X \in [0, P)$, расчет функции ядра может быть определен следующим образом:

$$C(X) \equiv \left| \sum_{i=1}^{n} k_i \cdot x_i \right|_{C_P}, \tag{2.6}$$

где $k_i = C(B_i)$.

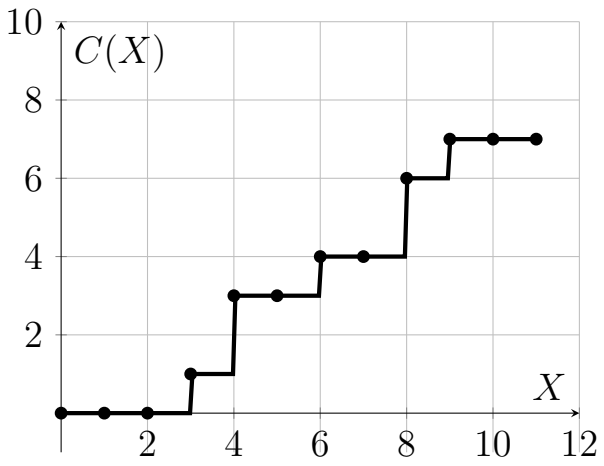

Рисунок 2.2 — График функции ядра с $w_1 = 1$, $w_2 = 2$ для СОК с основаниями $p_1 = 3$, $p_2 = 4$

Следует отметить, что при произвольном выборе весов w_i значение функции ядра $C(X)$ может выходить за границы интервала $[0, C_P)$, что приводит к возникновению проблемы критических ядер, ограничивающей применение формулы (2.6). В таких случаях диапазон возможных значений функции ядра расширяется до интервала $[-\delta_1, C_P + \delta_2]$, где δ_1 и δ_2 характеризуют величину отклонения от стандартного диапазона.

Пример 2.1.2. *Рассмотрим систему с модулями* $p_1 = 5, p_2 = 6$, *весами* $w_1 = -3, w_2 = 5$ *и параметрами* $P = 30$ *и* $C(P) = 7$. *График функции ядра для данной системы изображен на рисунке 2.3.*

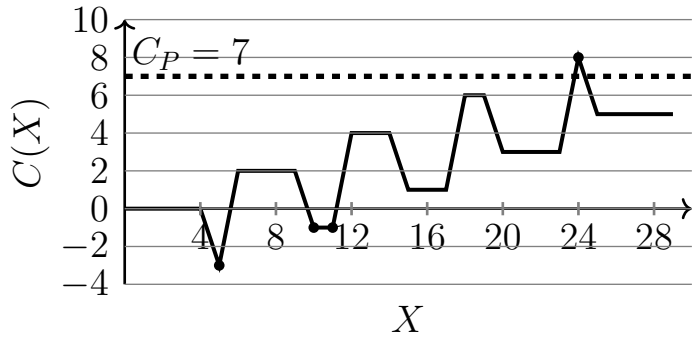

Рисунок 2.3 — График функции ядра Акушского с весами $w_1 = -3$, $w_2 = 5$ для СОК с основаниями $p_1 = 5$, $p_2 = 6$

В данном примере критические ядра возникают при $X = 5$, $X = 10$ *и* $X = 11$, *где функция ядра принимает отрицательные значения, что сви-*

детельствует о выходе за нижнюю границу допустимого диапазона, а при $X = 24$ *значение функции ядра превышает верхнюю границу* C_P.

При наличии критических ядер точное вычисление функции ядра становится невозможным с применением формулы (2.6). Рассмотрим в пункте 2.2 способы определения критических ядер для эффективной реализации различных немодульных операций в СОК [69].

2.2 Разработка методов определения критических ядер функции ядра Акушского

2.2.1 Метод определения критических ядер с использованием алгоритма усеченного перебора

Определение 2.2.1. *Пусть задано значение функции ядра* $C(X)$ *с весами* $w_1, w_2, \ldots, w_n$ *для* $X \in [0, P)$ *и* $C_P > 0$. *Тогда:*

1. *Нижнее критическое ядро — значение* $C(X)$, *удовлетворяющее условию:*
$$C(X) \in [-\delta_1, 0), \quad \text{где } \delta_1 > 0. \tag{2.7}$$
2. *Верхнее критическое ядро — значение* $C(X)$, *удовлетворяющее условию:*
$$C(X) \in [C_P, C_P + \delta_2], \quad \text{где } \delta_2 > 0. \tag{2.8}$$

Здесь δ_1 *и* δ_2 *— величины, характеризующие максимальное отклонение функции ядра Акушского от диапазона* $[0, C_P)$.

Находить нижние и верхние критические ядра можно предварительно используя формулу (2.4) и перебирая X в промежутке $[0, P)$. Однако при больших значения P данный метод будет невозможен в силу огромных временных затрат. Поэтому мы предлагаем перебирать X в промежутке $[0, p_n)$ для поиска нижних критических ядер, то есть

$$lower\ critical\ cores = \begin{cases} 1, \text{если } C(X) < 0, \\ 0, \text{если } 0 \leqslant C(X) < P. \end{cases} \tag{2.9}$$

А для поиска верхних критических ядер будем перебирать X в промежутке $[P - p_n, P)$. Тогда

$$upper\ critical\ cores = \begin{cases} 1, \text{если } C(X) \geqslant C(P), \\ 0, \text{если } 0 \leqslant C(X) < P. \end{cases} \tag{2.10}$$

Запишем выражения (2.9) и (2.10) в виде Алгоритма 3.

Алгоритм 3: Обнаружение критических ядер с помощью усеченного перебора

Input: $\{w_1, w_2, \ldots, w_n\}, \{p_1, p_2, \ldots, p_n\}$
Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$C(P) = \sum_{i=1}^{n} w_i \cdot P_i, i = 1, 2, \ldots, n$
Output: $lower\_critical\_cores$, $upper\_critical\_cores$

```
1  lower_critical_cores = 0, upper_critical_cores = 0
2  for X = 0, X < p_n, X++ do
3      X --COK--> (x_1, x_2, ..., x_n)
4      C(X) = X · Σ_{i=1}^{n} w_i/p_i − Σ_{i=1}^{n} x_i·w_i/p_i
5      if C(X) < 0 then
6          lower_critical_cores = 1
7          break
8  for X = P − p_n, X < P, X++ do
9      X --COK--> (x_1, x_2, ..., x_n)
10     C(X) = X · Σ_{i=1}^{n} w_i/p_i − Σ_{i=1}^{n} x_i·w_i/p_i
11     if C(X) ⩾ C(P) then
12         upper_critical_cores = 1
13         break
```

Result: $lower\_critical\_cores$, $upper\_critical\_cores$.

Пример 2.2.1. *Пусть дана система с модулями* $p_1 = 5, p_2 = 6$. *И веса* $w_1 = -3, w_2 = 5$. *Параметры* $P, C(P)$ *равны 30 и 7 соответственно.*

Перебирая числа в промежутке $[0, p_n)$ *мы дойдем до* $X = 5$, $X \xrightarrow{COK} (0, 5)$ *для данного числа*

$$C(X) = 5 \cdot \left(\frac{-3}{5} + \frac{5}{6}\right) - \left(\frac{0 \cdot (-3)}{5} + \frac{5 \cdot 5}{6}\right) = -3.$$

Поскольку $C(X) < 0$, *система с заданными весами содержит нижние критические ядра.*

Перебирая X *в промежутке* $[P - p_n, P)$ *и дойдя до* $X = 24,\ X \xrightarrow{COK} (4, 0)$ *для данного числа*

$$C(X) = 24 \cdot \left(\frac{-3}{5} + \frac{5}{6}\right) - \left(\frac{4 \cdot (-3)}{5} + \frac{0 \cdot 5}{6}\right) = 8.$$

Так как $C(X) > C(P)$ *значит при данных весах система имеет верхние критические ядра.*

2.2.2 Оптимизация поиска критических ядер функции ядра Акушского

В Теореме 2.2.1 сформулированы критерии, которые позволят определить имеет ли функция $C(X)$ критические ядра [69].

Теорема 2.2.1. *Функция ядра* $C(X)$, *заданная весами* $w_1, w_2, \ldots, w_n$, *не имеет критических ядер тогда и только тогда, когда для каждого* $k = 1, 2, \ldots, n$ *выполняются условия:*

1. $\sum_{i=1}^{n} w_i > 0$.
2. *Отсутствие нижних критических ядер:* $C(p_k) = \sum_{i=1}^{n} w_i \left\lfloor \frac{p_k}{p_i} \right\rfloor \geqslant 0$.
3. *Отсутствие верхних критических ядер:* $\sum_{i=1}^{n} \left(\left\lfloor \frac{p_k}{p_i} \right\rfloor + 1 \right) \cdot w_i - w_k > 0$.

Запишем условия теоремы в виде Алгоритма 4.

В следующем примере проиллюстрирована работа Алгоритма 4.

Пример 2.2.2. *Продолжим рассматривать систему с модулями* $p_1 = 5, p_2 = 6$ *и весами* $w_1 = -3, w_2 = 5$.

Для начала определим, есть ли у системы нижние критические ядра

$$C(p_1) = -3 \cdot \left\lfloor \frac{5}{5} \right\rfloor + 5 \cdot \left\lfloor \frac{5}{6} \right\rfloor = -3.$$

Так как $C(p_1) < 0$, *следовательно система с данными весами будет иметь нижние критические ядра Далее определим существуют ли верхние критические ядра. Для* $k = 1$ *получим*

$$\left(\left\lfloor \frac{5}{5} \right\rfloor + 1 \right) \cdot (-3) + \left(\left\lfloor \frac{5}{6} \right\rfloor + 1 \right) \cdot 5 - (-3) = 5 > 0.$$

Алгоритм 4: Обнаружение критических ядер на основе Теоремы 2.2.1

Input: $\{w_1, w_2, \ldots, w_n\}$, $\{p_1, p_2, \ldots, p_n\}$

Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$C(P) = \sum_{i=1}^{n} w_i \cdot P_i, i = 1, 2, \ldots, n$

Output: *lower_critical_cores*, *upper_critical_cores*

```
lower_critical_cores = 0, upper_critical_cores = 0
for k = 1, k ⩽ n, k++ do
    C(p_k) = Σ_{i=1}^{n} w_i · ⌊p_k / p_i⌋
    if C(p_k) < 0 then
        lower_critical_cores = 1
        break
for k = 1, k ⩽ n, k++ do
    sum = 0
    for i = 1, i ⩽ n, i++ do
        sum = sum + (⌊p_k / p_i⌋ + 1) · w_i
    sum = sum − w_k
    if sum ⩽ 0 then
        upper_critical_cores = 1
        break
```

Result: *lower_critical_cores*, *upper_critical_cores*.

Таким образом, система с данными весами имеет верхние критические ядра.

Предложенные методы были исследованы на примере обнаружения критических ядер для различных наборов модулей (таблицы 17, 18 приложение А). Чтобы избежать погрешностей при измерении времени, было проведено 10^6 измерений. В таблицах 19, 20 приложения А представлено среднее время выполнения методов определения критических ядер.

Метод на основе алгоритма усеченного перебора демонстрирует экспоненциальный рост временных затрат, что делает его непрактичным при увеличении числа модулей. Метод на основе Теоремы 2.2.1 в среднем на 99% быстрее предложенного усеченного алгоритма перебора.

2.3 Методы обратного преобразования перевода чисел из СОК в позиционную систему счисления

Обратное преобразование из системы остаточных классов в позиционную систему счисления является наиболее частой задачей. Существует несколько методов для выполнения этого преобразования, каждый из которых имеет свои преимущества и недостатки. Базовыми методами являются: метод на основе КТО и метод перевода в обобщенную позиционную систему счисления [82]. Другие методы представляют собой модификацию или комбинацию базовых подходов. К ним относятся, приближенная КТО и диагональная функция.

Метод, основанный на КТО, рассмотрен в параграфе 1.4.

В работе [104] предложено получать дробное представление X, которое получается путем деления (1.3) на P:

$$\frac{X}{P} = \left| \sum_{i=1}^{n} x_i \cdot \ c_i \right|_1, \tag{2.11}$$

где $c_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}$, дробная часть суммы дает результат в интервале $[0, 1)$.

При умножении (2.11) на P мы получим представление числа X в позиционной системе счисления. Метод обратного преобразования на основе приближенной КТО представлен Алгоритмом 5.

Рассмотрим пример применения приближенного метода.

Пример 2.3.1. *Дана система модулей* $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$ *и число* $X = (1, 2, 1, 4, 7)$. *Найдем представление* X *в ПСС. Вычислим константы* c_i*:*

$$c_1 = \frac{1}{2}, c_2 = \frac{2}{3}, c_3 = \frac{3}{5}, c_4 = \frac{1}{7}, c_5 = \frac{1}{11}.$$

С использованием (2.11) *найдем выражение* $\frac{X}{P}$

$$\frac{X}{P} = \left| 1 \cdot \frac{1}{2} + 2 \cdot \frac{2}{3} + 1 \cdot \frac{3}{5} + 4 \cdot \frac{1}{7} + 7 \cdot \frac{1}{11} \right|_1 = \left| 1\frac{52}{105} \right|_1 = \frac{52}{105}.$$

Тогда X *в ПСС*

$$X = \frac{52}{105} \cdot 2310 = 1481.$$

Основная сложность приближенной КТО состоит в том, что слагаемые вида $x_i \cdot c_i$, представляют собой рациональные величины, которые не могут

Алгоритм 5: Обратное преобразование из СОК в ПСС на основе приближенной КТО

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,
$P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$c_i = \frac{\left|P_i^{-1}\right|_{p_i}}{p_i}, i = 1, 2, \ldots, n$

Output: X

1 $sum = 0$
2 **for** $i = 1$, $i < n$, $i++$ **do**
3 $\quad sum = sum + x_i \cdot c_i$
4 $X = |sum|_1 \cdot P$

Result: X.

быть точно вычислены в рамках арифметики с фиксированной разрядностью. Как следствие, неизбежно возникает вопрос о точности округления, который при отсутствии дополнительных мер контроля способен привести к ошибочным результатам при выполнении немодульных операций.

Другим методом, является метод перевода из СОК в обобщенную позиционную систему счисления преобразования [118]. В ОПСС число имеет следующий вид:

$$X = d_1 + d_2 p_1 + d_3 p_1 p_2 + \ldots + d_n p_1 p_2 \ldots p_{n-1}, \tag{2.12}$$

где $0 \leqslant d_i < p_i$. Параметры d_i известны как цифры ОПСС.

Цифры d_i ОПСС могут быть вычислены следующим образом:

$$\begin{aligned}
&d_1 \equiv x_1 \pmod{p_1},\\
&d_2 \equiv (x_2 - d_1)\tau_{12} \pmod{p_2},\\
&d_3 \equiv ((x_3 - d_1)\tau_{13} - d_2)\tau_{23} \pmod{p_3},\\
&\quad\vdots\\
&d_n \equiv ((\ldots(x_n - d_1)\tau_{1n} - \ldots d_{n-1})\tau_{n-1n} \pmod{p_n}.
\end{aligned} \tag{2.13}$$

Предварительно вычисляются константы τ_{kj}, удовлетворяющие условию

$$\tau_{kj} p_k \equiv 1 \pmod{p_j}, (1 \leqslant k < j \leqslant n).$$

Рассмотрим перевод в ОПСС на Примере 2.3.2.

Таблица 4 — Метод перевода в ОПСС

Действия	Модули					Цифры
	$p_1 = 2$	$p_2 = 3$	$p_3 = 5$	$p_4 = 7$	$p_5 = 11$	
$X - d_1$	1 1	2 1	1 1	4 1	7 1	$d_1 = 1$
$(X - d_1) \cdot \tau_{1j}$	0	1 2	0 3	3 4	6 6	
$X_1 - d_2$		2 2	0 2	5 2	3 2	$d_2 = 2$
$(X_1 - d_2) \cdot \tau_{2j}$		0	3 2	3 5	1 4	
$X_2 - d_3$			1 1	1 1	4 1	$d_3 = 1$
$(X_2 - d_3) \cdot \tau_{3j}$			0	0 3	3 9	
$X_3 - d_4$				0 0	5 0	$d_4 = 0$
$(X_3 - d_4) \cdot \tau_{4j}$				0	5 8	
X_4					7	$d_5 = 7$

Пример 2.3.2. *Пусть дана система базисов* $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$. *Объем динамического диапазона* $P = 2 \cdot 3 \cdot 5 \cdot 7 \cdot 11 = 2310$. *Преобразуем число* $X = (1, 2, 1, 4, 7)$ *в позиционную систему счисления используя ОПСС.*

Сначала найдем константы τ_{kj}, *которые являются мультипликативными инверсиями* p_k *по модулю* p_j. *Для удобства запишем константы* τ_{kj} *в виде матрицы* $k \times j$:

$$\begin{pmatrix} 0 & 2 & 3 & 4 & 6 \\ 0 & 0 & 2 & 5 & 4 \\ 0 & 0 & 0 & 3 & 9 \\ 0 & 0 & 0 & 0 & 8 \end{pmatrix}$$

Используя (2.13) *запишем результаты в таблицу 4.*

Таким образом,

$$X = d_5p_1p_2p_3p_4 + d_4p_1p_2p_3 + d_3p_1p_2 + d_2p_1 + d_1 =$$

$$= 7 \cdot 2 \cdot 3 \cdot 5 \cdot 7 + 0 \cdot 2 \cdot 3 \cdot 5 + 1 \cdot 2 \cdot 3 + 2 \cdot 2 + 1 = 1481.$$

В литературе встречается другой способ обратного преобразования чисел с использованием, так называемой диагональной функции [46; 96]. Для числа X в СОК с базисом $\{p_1, p_2, \ldots, p_n\}$ диагональная функция определяется следующим образом:

$$D(X) = \left| x_1\check{k}_1 + x_2\check{k}_2 + \ldots + x_n\check{k}_n \right|_{SQ}. \tag{2.14}$$

где $SQ = \sum_{i=1}^{n} P_i$, $\check{k}_i = \left|-p_i^{-1}\right|_{SQ}$ — аддитивная мультипликативная инверсия p_i по модулю SQ.

На основе диагональной функции можно получить значение X в ПСС, с использованием следующего выражения

$$X = \frac{P}{SQ} \cdot \left| \sum_{i=1}^{n} x_i \cdot \left(\check{k}_i + \frac{1}{p_i} \right) \right|_{SQ} = \frac{P \cdot D(X) + x_1 \cdot P_1 + \ldots + x_n \cdot P_n}{SQ}. \tag{2.15}$$

Рассмотрим этот метод на примере.

Пример 2.3.3. *Продолжим рассматривать СОК с базисом* $\{2, 3, 5, 7, 11\}$ *и число* $X = 1481 = (1, 2, 1, 4, 7)$. *Из предыдущих примеров мы знаем, что* $P = 2310, P_1 = 1155, P_2 = 770, P_3 = 462, P_4 = 330, P_5 = 210$. *Тогда* $SQ = 2927$, *коэффициенты* $\check{k}_i$ *раны* $\check{k}_1 = 1463, \check{k}_2 = 1951, \check{k}_3 = 1756, \check{k}_4 = 418, \check{k}_5 = 266$. *Найдем диагональную функцию*

$$D(X) = |10655|_{2927} = 1874,$$

Используя (2.15) *находим искомое значение:*

$$X = \frac{4334887}{2927} = 1481.$$

Обратное преобразования на основе диагональной функции проигрывает аналогам по быстродействию, однако диагональная функция широко применяется для других немодульных операций в СОК.

2.3.1 Метод обратного преобразования с использованием КТО и ранга числа функции ядра Акушского

Для обратного преобразования из СОК в ПСС можно использовать функцию ядра Акушского [1]. Из выражения (2.4), число X может быть вычислено следующим образом

$$X = \left| \frac{P}{C(P)} \cdot \left(C(X) + \sum_{i=1}^{n} \frac{w_i}{p_i} \cdot x_i \right) \right|_P . \tag{2.16}$$

Обратное преобразование из СОК в ПСС с применением функции ядра Акушского возможно реализовать с помощью Алгоритма 6.

Алгоритм 6: Обратное преобразование из СОК в ПСС с использованием функции ядра

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,
$P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$B_i = \left| P_i^{-1} \right|_{p_i} \cdot P_i$,
$k_i = C\left(B_i\right)$,
$C\left(P\right) = \sum_{i=1}^{n} w_i \cdot P_i, i = 1, 2, \ldots, n$

Output: X

1 $S_1 = 0$
2 $S_2 = 0$
3 **for** $i = 1$, $i \leqslant n$, $i++$ **do**
4 $\quad S_1 = S_1 + x_i \cdot k_i$
5 $\quad S_2 = S_2 + \frac{w_i}{p_i} \cdot x_i$
6 $X = \left| \frac{P}{C(P)} \cdot (S_1 + S_2) \right|_P$

Result: X.

Рассмотрим пример, схожий с предыдущим.

Пример 2.3.4. *Рассмотрим набор модулей* $p_1 = 2, p_2 = 3, p_3 = 5, p_4 = 7, p_5 = 11$. *Выберем для функции ядра веса* $w_1 = 1, w_2 = 0, w_3 = 0, w_4 = 0, w_5 = 1$.

Рассчитаем следующие значения:

$$P = 2310, P_1 = \frac{P}{p_1} = 1155, P_2 = \frac{P}{p_2} = 550, P_3 = \frac{P}{p_3} = 462,$$
$$P_4 = \frac{P}{p_4} = 330, P_5 = \frac{P}{p_5} = 210, \left|P_1^{-1}\right|_{p_1} = 1, \left|P_2^{-1}\right|_{p_2} = 2,$$
$$\left|P_3^{-1}\right|_{p_3} = 3, \left|P_4^{-1}\right|_{p_4} = 1, \left|P_5^{-1}\right|_{p_5} = 1.$$

Отсюда найдем ортогональные базисы:

$$B_1 = 1155, B_2 = 1540, B_3 = 1386, B_4 = 330, B_5 = 210.$$

Функция ядра от динамического диапазона равна

$$C_P = P_1 + P_5 = 1365.$$

Далее найдем значения функции ядра от ортогональных базисов:

$$C(B_1) = 682, C(B_2) = 910, C(B_3) = 819, C(B_4) = 195, C(B_5) = 124.$$

Рассмотрим число в СОК равное $(1, 2, 1, 4, 7)$. *Используя* (2.6) *находим*

$$C(X) = (1 \cdot 682 + 2 \cdot 910 + 1 \cdot 819 + 4 \cdot 195 + 7 \cdot 124) \bmod 1365 = 874.$$

По формуле (2.16) *находим*

$$X = \left|\frac{2310}{1365} \cdot \left(874 + \frac{1 \cdot 1}{2} + \frac{7 \cdot 1}{11}\right)\right|_{2310} = 1481.$$

Однако обратное преобразование с использованием функции ядра является неэффективным. Мы предлагаем высокоскоростной метод перевода чисел из СОК в ПСС с использованием КТО и ранга числа функции ядра. Для этой цели необходимо рассмотреть понятие ранга числа СОК. Уравнение (1.3) может быть переписано в следующем виде

$$X = \sum_{i=1}^{n} P_i \cdot x_i \cdot \left|P_i^{-1}\right|_{p_i} - r(X) \cdot P, \tag{2.17}$$

где $r(X)$ – ранг числа вычисляемый как

$$r(X) = \left\lfloor \sum_{i=1}^{n} \frac{1}{p_i} \cdot \left|P_i^{-1}\right|_{p_i} \cdot x_i \right\rfloor. \tag{2.18}$$

На основе (2.6) и (2.17) можно определить ранг числа функции ядра Акушского

$$\check{r}(X) = \left\lfloor \frac{\sum_{i=1}^{n} k_i \cdot x_i}{C_P} \right\rfloor. \tag{2.19}$$

Рассмотрим теорему, которая связывают данные позиционные характеристики [3].

Теорема 2.3.1. *Пусть* $p_1 < p_2 < \ldots < p_n$, *число* $X \xrightarrow{COK} (x_1, x_2, \ldots, x_n)$ *и веса функции ядра Акушского* $w_1, w_2, \ldots, w_n$, *удовлетворяющие условию* $0 \leqslant X < P$, *тогда* $\check{r}(X) = r(X) + \left\lfloor \frac{C(X)}{C_P} \right\rfloor$.

Для использования обратного преобразования на основе КТО и ранга числа вычисляемого с помощью функции ядра Акушского доказана Теорема 2.3.2.

Теорема 2.3.2. *Для функции ядра Акушского, у которой отсутствуют критические ядра, выполняется*

$$\check{r}(X) = r(X). \tag{2.20}$$

Доказательство. При отсутствии критических ядер $C(X) \in [0; C_P)$. Согласно Теореме 2.3.1, $\check{r}(X) = r(X) + \left\lfloor \frac{C(X)}{C_P} \right\rfloor$. Отсюда $\left\lfloor \frac{C(X)}{C_P} \right\rfloor = 0$, следовательно $\check{r}(X) = r(X)$.

Теорема доказана. □

Таким образом, в случае отсутствия критических ядер ранг числа СОК на основе КТО равен рангу числа вычисляемого на основе функции ядра Акушского.

С использованием КТО и ранга функции ядра Акушского, формулу (2.17) можно записать в виде

$$X = \sum_{i=1}^{n} P_i \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} - \check{r}(X) \cdot P. \tag{2.21}$$

Учитывая вышеперечисленное разработан Алгоритм 7, использующий КТО и ранг числа функции ядра Акушского.

Рассмотрим предложенный нами метод на примере.

Пример 2.3.5. *Аналогично, дана СОК* $\{2, 3, 5, 7, 11\}$ *и число* $X = (1, 2, 1, 4, 7)$. $P = 2310, P_1 = 1155, P_2 = 770, P_3 = 462, P_4 = 330, P_5 = 210$. *Используем набор весов* $w_1 = 0, w_2 = 0, w_3 = 0, w_4 = 0, w_5 = 1$.

Алгоритм 7: Обратное преобразование из СОК в ПСС с использованием КТО и ранга числа функции ядра Акушского

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,
$P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}$,
$B_i = \left|P_i^{-1}\right|_{p_i} \cdot P_i$,
$k_i = C\left(B_i\right)$,
$C\left(P\right) = \sum_{i=1}^{n} w_i \cdot P_i, i = 1, 2, \ldots, n$

Output: X

1 $S = 0$
2 $r = 0$
3 **for** $i = 1$, $i \leqslant n$, $i++$ **do**
4 $\quad S = S + x_i \cdot B_i$
5 $\quad r = r + x_i \cdot k_i$
6 $X = S - \left\lfloor \frac{r}{C_P} \right\rfloor$

Result: X.

Вычислим значения B_i:

$$B_1 = P_1 \cdot \left|P_1^{-1}\right| = 1155, B_2 = P_2 \cdot \left|P_2^{-1}\right| = 1540,$$

$$B_3 = P_3 \cdot \left|P_3^{-1}\right| = 1386, B_4 = P_4 \cdot \left|P_4^{-1}\right| = 330,$$

$$B_5 = P_5 \cdot \left|P_5^{-1}\right| = 210.$$

Затем находим значение диапазона функции ядра

$$C\left(P\right) = C_P = 210.$$

Найдем значение коэффициентов k_i:

$$k_1 = 105, k_2 = 140, k_3 = 126, k_4 = 30, k_5 = 19.$$

Используя (2.19) *ранг функции ядра равен*

$$\check{r}\left(X\right) = \left\lfloor \frac{105 \cdot 1 + 140 \cdot 2 + 126 \cdot 1 + 30 \cdot 4 + 19 \cdot 7}{210} \right\rfloor = 3.$$

Таким образом,

$$X = 1155 \cdot 1 + 1540 \cdot 2 + 1386 \cdot 1 + 330 \cdot 4 + 210 \cdot 7 - 3 \cdot 2310 = 1481.$$

Предложенный метод позволяет уйти от вычислительно сложных операций деления и взятия по модулю, присутствующих в формуле (2.16), что позволяет сократить время обратного преобразования из СОК в ПСС.

2.4 Модификация итерационного деления в системе остаточных классов

Наиболее сложной операцией в СОК является деление чисел [108]. Акушский предложил алгоритм итерационного деления, способный обеспечивать быстрые результаты при выполнении деления [1]. Рассмотрим применение итерационного деления с использованием функции ядра.

Алгоритм деления базируется на особой простоте выполнения элементарных операций деления на 2 и на основание СОК, а также на определении четности числа. Рассмотрим процесс итерационного деления для СОК с базисом $\{p_1, p_2, \ldots, p_n\}$ для делимого $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ с ядром $C(A)$ на делитель $B = (\beta_1, \beta_2, \ldots, \beta_n)$ с ядром $C(B)$ поэтапно.

Первый этап — если $\beta_1 = 0$, то делим B на p_1 в противном случае делим B на 2, получаем B_1. Делим B_1 на p_1, если $\beta_1' = 0$, и на 2, в противном случае получаем B_2 и т.д. до $B_k = 1$.

Параллельно с этим делим A на p_1, если $\beta_1 = 0$, и на 2, в противном случае получаем A_1. Делим A_1 на p_1, если $\beta_1' = 0$, и на 2, в противном случае получаем A_2 и т.д. до A_k. Данный процесс представлен в Алгоритме 8.

Второй этап — вычисляется первая невязка:

$$A - BA_k = Q^{(1)}.$$

Третий этап в соответствии с первым этапом деления делителя B вычисляется так:

$$Q_1, Q_2, \ldots, Q_k^{(1)}.$$

При этом если имеется возможность запомнить соответствующие делители $d_1, d_2, \ldots, d_k$ делителя B, который является как бы ведущим в рассматриваем процессе деления, то не надо повторять деление B, и содержание этапа сводится лишь к делению $Q^{(1)}$.

Четвертый этап — вычисляется вторая невязка:

$$Q_1 - Q_k^{(1)} \cdot B = Q^{(2)}. \tag{2.22}$$

Эти этапы повторяются до получения нулевого промежуточного частного $Q_i^{(l+1)} = 0, (i = 1, 2, \ldots, k)$. Тогда искомое частное:

$$Z = \frac{A}{B} = A_k + Q_k^{(1)} + Q_k^{(2)} + \ldots + Q_k^{(l)}, \tag{2.23}$$

где

$$Q_k^{(l+1)} = 0. \tag{2.24}$$

Рассмотрим пример итерационного деления в системе остаточных классов.

Пример 2.4.1. *Рассмотрим СОК с основаниями* $p_1 = 7, p_2 = 9, p_3 = 11$, *с системой весов* $w_1 = -1, w_2 = -1, w_3 = 3$, *ядром диапазона* $C(P) = C_P = 13$.

Пусть дано делимое $A = (5, 7, 10)$ *с ядром* $C(A) = C_A = 7$ *и делитель* $B = (3, 6, 10)$ *с ядром* $C(B) = C_B = 0$.

Первый этап:

$$B_1 = (3, 6, 10)/2 \approx (2, 5, 9)/2 = (1, 7, 10), C_{B_1} = -1;$$

$$B_2 = (1, 7, 10)/2 \approx (0, 6, 9)/2 = (0, 3, 10), C_{B_2} = -2;$$

$$B_3 = (0, 6, 9)/p_1 = (0, 6, 9)/7 = (3, 3, 3), C_{B_3} = 0;$$

$$B_4 = (3, 3, 3)/2 \approx (2, 2, 2)/2 = (1, 1, 1), C_{B_4} = 0.$$

Параллельно выполняем операции над делимым A:

$$A_1 = (5, 7, 10)/2 \approx (4, 6, 2)/2 = (2, 3, 10), C_{A_1} = 2;$$

$$A_2 = (2, 3, 10)/2 \approx (1, 2, 9)/2 = (4, 1, 10), C_{A_2} = 0;$$

$$A_3 = (4, 1, 10)/7 = (0, 6, 6)/7 = (1, 6, 4), C_{A_3} = 0;$$

$$A_4 = (1, 6, 4)/2 \approx (0, 5, 3)/2 = (0, 7, 7), C_{A_4} = -1.$$

Первое промежуточное частное $A_4 = (0, 7, 7)$ *с* $C_{A_4} = -1$.

Второй этап — вычисляется первая невязка.

Для этого найдем $A_4 \cdot B = (0, 7, 7) \cdot (3, 6, 10) = (0, 6, 4)$ *с истинным ядром* $C_{A_4 B} = 11$.

Вычисляем функции четности A_4, B *и* $A_4 \cdot B$. *Получаем* $\varphi(A_4) = 1$, $\varphi(B) = 1$, $\varphi(A_4 \cdot B) = 1$.

Так как $\varphi(A_4) \cdot \varphi(B) = \varphi(A_4B)$, *то следовательно,* $A_4B < P$. *Найдем невязку* $Q^{(1)} = A - A_4B = (5,7,10) - (0,6,4) = (5,1,6)$ *с расчетным ядром* $C_{Q^{(1)}} = 9$.

Так как $\overline{C}_{Q^{(1)}} \neq C_{Q^{(1)}}$, *то истинная невязка равна* $Q_n^{(1)} = P - Q^{(1)} = (2,8,5)$ *с ядром* $C_{Q_n^{(1)}} = 3$ *и знаком минус.*

Третий этап — повторение первого этапа для $Q_n^{(1)}$:

$$Q_1^{(1)} = (2,8,5)/2 = (1,4,8), C_{Q_1^{(1)}} = 0;$$

$$Q_2^{(1)} = (1,4,8)/2 \approx (0,3,7)/2 = (0,6,9), C_{Q_2^{(1)}} = -1;$$

$$Q_3^{(1)} = (0,6,9)/7 = (6,6,6), C_{Q_3^{(1)}} = 0;$$

$$Q_4^{(1)} = (6,6,6)/2 = (3,3,3), C_{Q_4^{(1)}} = 0.$$

Второе промежуточное частное $Q_4^{(1)} = (3,3,3)$ *с* $C_{Q_4^{(1)}} = 0$.

Четвертый этап — вычисляется вторая невязка.

Следуя второму этапу, получим $Q_4^{(1)} \cdot B = (3,3,3) \cdot (3,6,10) = (2,0,8)$ *с истинным ядром* $C_{Q_4^{(1)}} = 3$. *Здесь* $\varphi(Q_4^{(1)}) = 1$, $\varphi(Q_4^{(1)} \cdot B) = 1$ *и* $\varphi(Q_4^{(1)}) \cdot \varphi(B) = \varphi(Q_4^{(1)} \cdot B)$, *т. е.* $Q_4^{(1)} \cdot B < P$.

Найдем невязку с учетом знака минус у $Q_n^{(1)} Q^{(2)} = Q_n^{(1)} \cdot B - Q_4^{(1)} = (2,0,8) - (2,8,5) = (0,1,3)$ *с расчетным ядром* $C_{Q_4^{(2)}} = 1$ *и истинным ядром* $C_{Q_4^{(2)}} = 1$.

Так как $C_{Q_4^{(2)}} = C_{Q_4^{(2)}}$, *то получена истинная невязка.*

Пятый этап — повторение первого этапа для $C^{(2)}$:

$$Q_1^{(2)} = (0,1,3)/2 \approx (6,0,2)/2 = (3,0,1), C_{Q_1^{(2)}} = 1;$$

$$Q_2^{(2)} = (3,0,1)/2 \approx (2,8,0)/2 = (1,4,0), C_{Q_2^{(2)}} = 1;$$

$$Q_3^{(2)} = (0,6,9)/7 \approx (0,3,10)/7 = (3,3,3), C_{Q_3^{(2)}} = 0;$$

$$Q_4^{(2)} = (6,6,6)/2 \approx (2,2,2)/2 = (1,1,1), C_{Q_4^{(2)}} = 0.$$

Третье промежуточное частное $Q_4 = (1,1,1)$ *с* $C_{Q_4^{(2)}} = 0$.

На этом деление можно закончить, так как Q_4 *будет обязательно равно нулю.*

Составим частное:

$$\frac{A}{B} = \frac{(5,7,10)}{(3,6,10)} = (0,7,7) - (3,3,3) + (1,1,1) = (5,5,5).$$

с расчетным ядром $C_{A/B} = -1 - 0 + 0 - (-1) = 0$.

Действительно,

$$\frac{A}{B} = \frac{(5,7,10)}{(3,6,10)} = \frac{439}{87} = 5\frac{4}{84}.$$

В Алгоритме 9 представлено итерационное деление чисел.

Алгоритм 8: Вспомогательная функция для итерационного деления (basicDivision)

Input: $X = (x_1, x_2, \ldots, x_n)$, k

Output: X_k, C_{X_k}

Инициализировать X_k, C_{X_k}
if $x_1 = 0$ **then**
    $X_k = X_k \cup \left\{\frac{(x_1,x_2,\ldots,x_n)}{p_1}\right\}$
    $C_{X_k} = C_{X_k} \cup \{C(X_1)\}$
else
    $X_k = X_k \cup \left\{\frac{(x_1,x_2,\ldots,x_n)}{2}\right\}$
    $C_{X_k} = C_{X_k} \cup \{C(X_1)\}$
for $i = 2$, $i \leqslant k$, $i{+}{+}$ **do**
    if $x_1' = 0$ **then**
        $X_k = X_k \cup \left\{\frac{X_i}{p_1}\right\}$
        $C_{X_k} = C_{X_k} \cup \{C(X_i)\}$
    else
        $X_k = X_k \cup \left\{\frac{X_i}{2}\right\}$
        $C_{X_k} = C_{X_k} \cup \{C(X_i)\}$

Result: X_k, C_{X_k}.

Как видно из Алгоритма 9, функция ядра играет важную роль при выполнении итерационного деления. На каждом шаге возникает необходимость вычисления функции ядра. Для оптимизации итерационного деления докажем следующие теоремы.

Теорема 2.4.1. *Функцию ядра от числа, деленного пополам, можно вычислить следующим образом*

$$C\left(\frac{X}{2}\right) = \frac{C(X) - \sum_{i=1,|x_i|_2=1}^{n} w_i}{2}. \tag{2.25}$$

Алгоритм 9: Итерационное деление чисел

Input: $A = (\alpha_1, \alpha_2, \ldots, \alpha_n)$, $B = (\beta_1, \beta_2, \ldots, \beta_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$, $\{w_1, w_2, \ldots, w_n\}$,
$P = \prod_{i=1}^{n} p_i$,$P_i = \frac{P}{p_i}$,$C_P = \sum_{i=1}^{n} P_i \cdot w_i$,
$B_i = \left|P_i^{-1}\right|_{p_i} \cdot P_i$, $k_i = C\left(B_i\right)$, $i = 1, 2, \ldots, n$

Output: Z

Инициализировать A_k, C_{A_k}, B_k, C_{B_k}
if $\beta_1 = 0$ **then**
    $B_k = B_k \cup \left\{\frac{(\beta_1, \beta_2, \ldots, \beta_n)}{p_1}\right\}$
    $C_{B_k} = C_{B_k} \cup \{C(B_1)\}$
    $A_k = A_k \cup \left\{\frac{(\alpha_1, \alpha_2, \ldots, \alpha_n)}{p_1}\right\}$
    $C_{A_k} = C_{A_k} \cup \{C(A_1)\}$
else
    $B_k = B_k \cup \left\{\frac{(\beta_1, \beta_2, \ldots, \beta_n)}{2}\right\}$
    $C_{B_k} = C_{B_k} \cup \{C(B_1)\}$
    $A_k = A_k \cup \left\{\frac{(\alpha_1, \alpha_2, \ldots, \alpha_n)}{2}\right\}$
    $C_{A_k} = C_{A_k} \cup \{C(A_1)\}$
while $B_k \neq 1$ **do**
    if $\beta_1' = 0$ **then**
        $B_k = B_k \cup \left\{\frac{B_i}{p_1}\right\}$
        $C_{B_k} = C_{B_k} \cup \{C(B_i)\}$
        $A_k = A_k \cup \left\{\frac{A_i}{p_1}\right\}$
        $C_{A_k} = C_{A_k} \cup \{C(A_i)\}$
    else
        $B_k = B_k \cup \left\{\frac{B_i}{2}\right\}$
        $C_{B_k} = C_{B_k} \cup \{C(B_i)\}$
        $A_k = A_k \cup \left\{\frac{A_i}{2}\right\}$
        $C_{A_k} = C_{A_k} \cup \{C(A_i)\}$
Инициализировать $Q^{(k)}$, $Q_k^{(1)}$
$Q^{(k)} = Q^{(k)} \cup \left\{Q^{(1)} = A - BA_k\right\}$;
$Q_k^{(1)}$, $C_{Q_k^{(1)}} = \text{basicDivision}(Q^{(1)},k)$
$Q_k^{(1)} = Q_k^{(1)} \cup \left\{Q_k^{(1)}\right\}$
$Q^{(k)} = Q^{(k)} \cup \left\{Q^{(2)} = Q_1 - Q_k^{(1)} \cdot B\right\}$
while $Q_i^{(l)} \neq 0$ **do**
    $Q_k^{(i)}$, $C_{Q_k^{(i)}} = \text{basicDivision}(Q^{(i)},k)$
    $Q_k^{(1)} = Q_k^{(1)} \cup \left\{Q_k^{(i)}\right\}$
    $Q^{(k)} = Q^{(k)} \cup \left\{Q^{(i)} = Q_i - Q_k^{(i)} \cdot B\right\}$
$Z = A_k + \sum_{i=1}^{l} Q_k^{(i)}$;

Result: $Z = \frac{A}{B}$.

Доказательство. Так как значение функции ядра Акушского вычисляется по формуле (2.1), тогда:

$$C\left(\frac{X}{2}\right) = \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{2p_i} \right\rfloor.$$

Рассмотрим функцию ядра в следующем виде:

$$C(X) = \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \frac{X}{2p_i} \right\rfloor = \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \left(\left\lfloor \frac{X}{2p_i} \right\rfloor + \left\{ \frac{X}{2p_i} \right\} \right) \right\rfloor$$

$$= \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \left\lfloor \frac{X}{2p_i} \right\rfloor + 2 \cdot \left\{ \frac{X}{2p_i} \right\} \right\rfloor = 2 \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{X}{2p_i} \right\rfloor + \sum_{i=1}^{n} \left\lfloor 2 \cdot \left\{ \frac{X}{2p_i} \right\} \right\rfloor$$

$$= 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor 2 \cdot \frac{|X|_{2p_i}}{2p_i} \right\rfloor = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{|X|_{2p_i}}{p_i} \right\rfloor,$$

так как $|X|_2 = 0$ и $|X|_{p_i} = x_i$, тогда:

$$C(X) = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1}^{n} w_i \cdot \left\lfloor \frac{|x_i|_2 \, p_i + x_i}{p_i} \right\rfloor = 2 \cdot C\left(\frac{X}{2}\right) + \sum_{i=1, |x_i|_2 = 1}^{n} w_i.$$

Отсюда следует, что:

$$C\left(\frac{X}{2}\right) = \frac{C(X) - \sum_{i=1, |x_i|_2 = 1}^{n} w_i}{2}.$$

Теорема доказана. □

Если в базисе СОК отсутствуют четные модули, то функцию ядра от числа, деленного пополам можно вычислить в соответствии с Теоремой 2.4.2.

Теорема 2.4.2. *Для СОК* $\{p_1, p_2, \ldots, p_n\}$, *где* p_i — *нечетные числа и* $Y = \frac{X}{2}$, *значение функции ядра от* Y *равно*

$$C(Y) = \left| \frac{1}{2} C(X) + \frac{1}{2} \sum_{|x_i|_2 = 1} C(B_i) \cdot x_i \right|_{C_P}. \tag{2.26}$$

Доказательство. Так как

$$C(X) = \left| \sum_{i=1}^{n} C(B_i) \cdot x_i \right|_{C_P}.$$

Тогда

$$C(Y) = \left| \sum_{i=1}^{n} C(B_i) \cdot y_i \right|_{C_P} = \left| \sum_{|x_i|_2=0} C(B_i) \cdot \frac{x_i}{2} + \sum_{|x_i|_2=1} C(B_i) \cdot \frac{x_i + p_i}{2} \right|_{C_P}$$

$$= \left| \sum_{|x_i|_2=0} C(B_i) \cdot \frac{x_i}{2} + \sum_{|x_i|_2=1} C(B_i) \cdot \frac{x_i}{2} + \sum_{|x_i|_2=1} C(B_i) \cdot \frac{p_i}{2} \right|_{C_P}$$

$$= \left| \sum_{i=1}^{n} C(B_i) \cdot \frac{x_i}{2} + \sum_{|x_i|_2=1} C(B_i) \frac{p_i}{2} \right|_{C_P} = \left| \frac{1}{2} \sum C(B_i) \cdot x_i + \frac{1}{2} \sum_{|x_i|_2=1} C(B_i) \cdot p_i \right|_{C_P}$$

$$= \left| \frac{1}{2} \left| \sum C(B_i) \cdot x_i \right|_{C_P} + \frac{1}{2} \sum_{|x_i|_2=1} C(B_i) p_i \right|_{C_P} = \left| \frac{1}{2} C(X) + \frac{1}{2} \sum_{|x_i|_2=1} C(B_i) \cdot x_i \right|_{C_P}.$$

Теорема доказана. □

Предложенные теоремы позволяют сократить время расчета функции ядра Акушского, что позволит снизить общее время итерационного деления в СОК. Применение данных теорем рассмотрим в параграфе 3.5.

2.5 Методы определения знака числа в системе остаточных классов

В СОК с базисом $\{p_1, p_2, \ldots, p_n\}$ и динамическим диапозоном $P = \prod_{i=1}^{n} p_i$, общий интервал значений $[0, P)$ может быть разбит на непересекающиеся поддиапазоны, соответствующие положительным и отрицательным числам. В качестве примера рассмотрим число X, для которого выполняются следующие условия:

$$-\frac{P+1}{2} \leqslant X < \frac{P+1}{2}, \text{ если } P \text{ нечетное,} \tag{2.27}$$

$$-\frac{P}{2} \leqslant X < \frac{P}{2}, \text{ если } P \text{ четное.} \tag{2.28}$$

Для произвольного натурального числа $X \in \mathbb{N}$, удовлетворяющего условию $X \leqslant \frac{P}{2}$, выполняется $P - X \equiv -X \pmod{P}$. Это свойство позволяет

установить правило определения знака числа в системе остаточных классов при четном динамическом диапазоне P в рамках полной системы наименьших неотрицательных вычетов:

$$Sign\,(X) = \begin{cases} 0, & \text{если } 0 \leqslant X < \frac{P}{2}, \\ 1, & \text{если } \frac{P}{2} \leqslant X < P. \end{cases} \tag{2.29}$$

Для нечетного динамического диапазона знак числа определим по формуле

$$Sign\,(X) = \begin{cases} 0, & \text{если } 0 \leqslant X < \frac{P+1}{2}, \\ 1, & \text{если } \frac{P+1}{2} \leqslant X < P. \end{cases} \tag{2.30}$$

Определить знак числа можно используя методы обратного преобразования. С использованием КТО определение знака числа можно записать в виде Алгоритма 10.

Алгоритм 10: Определение знака числа с использованием КТО

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,

$P = \prod_{i=1}^{n} p_i$,

$K = \begin{cases} \frac{P}{2}, & \text{если } P \text{ четное} \\ \frac{P+1}{2}, & \text{если } P \text{ нечетное.} \end{cases}$

$P_i = \frac{P}{p_i}$,

$B_i = \left|P_i^{-1}\right|_{p_i} \cdot P_i, i = 1, 2, \ldots, n$

Output: $sign$

1 $sum = 0$

2 **for** $i = 1$, $i \leqslant n$, $i{+}{+}$ **do**

3 $\quad sum = sum + x_i \cdot B_i$

4 $X = sum \mod P$

5 **if** $X < K$ **then**

6 $\quad sign = 0$

7 **else**

8 $\quad sign = 1$

Result: $sign$.

Приведем Пример 2.5.1 определения знака числа с использованием КТО.

Пример 2.5.1. *Рассмотрим СОК с основаниями* $p_1 = 3, p_2 = 7, p_3 = 8$ *объем динамического диапазона* $P = 3 \cdot 7 \cdot 8 = 168$. *Середина динамического диапазона* $K = \frac{P}{2} = 84$ *Определим знак числа* $X = (2, 0, 1)$ *с использованием КТО.*

Значения P_i *равны:*

$$P_1 = \frac{P}{p_1} = 56, P_2 = 24, P_3 = 21.$$

Значения мультипликативных инверсий равны:

$$\left|P_1^{-1}\right|_{p_1} = 2, \left|P_2^{-1}\right|_{p_2} = 5, \left|P_2^{-1}\right|_{p_2} = 5.$$

Имея данные значения можно вычислить X, *используя* (1.3)

$$X = |56 \cdot 2 \cdot 2 + 24 \cdot 0 \cdot 5 + 21 \cdot 1 \cdot 5|_{168} = 161.$$

Поскольку $161 > 84$, *число* (2,0,1) *отрицательное.*

С учетом работы алгоритмов обратного преобразования с использованием приближенного метода на основе КТО и метода на основе ОПСС описных в параграфе 2.3, определение знака числа в СОК с использованием данных методов будет аналогично Алгоритму 10.

Для определения знака можно использовать диагональную функцию, не проводя обратного преобразования в ПСС. Определение знака числа с использованием диагональной функции представлено Алгоритмом 11.

Рассмотрим пример определения знака числа с использованием диагональной функции.

Пример 2.5.2. *Рассмотрим СОК с основаниями* $p_1 = 3, p_2 = 7, p_3 = 8$ *объем динамического диапазона* $P = 168, K = \frac{P}{2} = 84, D(K) = 50$. *Определим знак числа* $X = (2, 0, 1)$ *с использованием диагональной функции.*

Значения P_i *равны:*

$$P_1 = \frac{P}{p_1} = 56, P_2 = 24, P_3 = 21,$$
$$SQ = P_1 + P_2 + P_3 = 101.$$

Коэффициенты диагональной функции равны

$$\check{k}_1 = \left|-p_1^{-1}\right|_{SQ} = 67, \check{k}_2 = 72, \check{k}_3 = 63.$$

Посчитав $D(X)$, *мы получим:*

$$D(X) = |67 \cdot 2 + 72 \cdot 0 + 63 \cdot 1|_{101} = 96.$$

Так как $96 > 50$, *число* $(2, 0, 1)$ *отрицательное.*

Алгоритм 11: Определение знака числа с использованием диагональной функции

Input: $(x_1, x_2, \ldots, x_n)$

Data: $\{p_1, p_2, \ldots, p_n\}$,

$P = \prod_{i=1}^{n} p_i$,

$K = \begin{cases} \frac{P}{2}, & \text{если } P \text{ четное} \\ \frac{P+1}{2}, & \text{если } P \text{ нечетное.} \end{cases}$

$D(K) = \sum_{i=1}^{n} \left\lfloor \frac{K}{p_i} \right\rfloor$,

$P_i = \frac{P}{p_i}$,

$SQ = \sum_{i=1}^{n} P_i$,

$\check{k}_i = \left| -p_i^{-1} \right|_{SQ}, i = 1, 2, \ldots, n$

Output: $sign$

```
1 sum = 0
2 for i = 1, i ⩽ n, i++ do
3 |   sum = sum + x_i · k̆_i
4 D(X) = sum mod SQ
5 if D(X) < D(K) then
6 |   sign = 0
7 else
8 |   sign = 1
```

Result: $sign$.

2.5.1 Алгоритм определения знака числа на основе функции ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$

Используя соотношение (2.31) можно определить знак числа в СОК, на основе функции ядра.

$$Sign\,(X) = \begin{cases} 0, & \text{если } C\,(X) = \left|\sum_{i=1}^{n} k_i \cdot x_i\right|_{C(P)} < C\,(K)\,, \\ 1, & \text{если } C\,(X) = \left|\sum_{i=1}^{n} k_i \cdot x_i\right|_{C(P)} \geqslant C\,(K)\,, \end{cases} \tag{2.31}$$

где $K = \frac{P}{2}$ если P четное, либо $K = \frac{P+1}{2}$ если P нечетное.

Выбирая веса так чтобы $C_P = 2^N$ можно построить функцию ядра, наиболее подходящую для определения знака числа в СОК.

Мы предлагаем минимальную функцию с заданными свойствами для определения знака числа на основе функции ядра Акушского для специального набора модулей $\{2^n-1, 2^{n+a}, 2^n+1\}$, где $0 \leqslant a < n$. Определим функцию ядра следующим образом:

$$C(X) = w_1 \left\lfloor \frac{X}{2^n-1} \right\rfloor + w_2 \left\lfloor \frac{X}{2^{n+a}} \right\rfloor + w_3 \left\lfloor \frac{X}{2^n+1} \right\rfloor, \tag{2.32}$$

и $C_P = C(P) = w_1P_1 + w_2P_2 + w_3P_3$, где $P = (2^n-1)\,2^{n+a}\,(2^n+1)$ и $P_1 = 2^{n+a}\,(2^n+1)\,, P_2 = (2^n-1)\,(2^n+1)\,, P_3 = (2^n-1)2^{n+a}$ и $w_1, w_2, w_3 \in \mathbb{Z}$.

Исследуем, какие ограничения накладываются на коэффициенты w_1, w_2 и w_3 функции $C(X)$, чтобы функция удовлетворяла условию $\forall X \in [0, P)$: $0 \leqslant C(X) \leqslant C_P$. Для этого докажем леммы 2.5.1, 2.5.2 и 2.5.3.

Лемма 2.5.1. *Если* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$, *тогда*

$$\begin{cases} 0 \leqslant w_1 \leqslant C_P, \\ 0 \leqslant w_1 + w_2 + w_3 \leqslant C_P. \end{cases}$$

Доказательство. Вычислим $C(2^n-1)$, $C(P-1)$:

$$C(2^n-1) = w_1 \left\lfloor \frac{2^n-1}{2^n-1} \right\rfloor + w_2 \left\lfloor \frac{2^n-1}{2^{n+a}} \right\rfloor + w_3 \left\lfloor \frac{2^n-1}{2^n+1} \right\rfloor = w_1,$$

$$C(P-1) = w_1(P_1-1) + w_2(P_2-1) + w_3(P_3-1) = C_P - w_1 - w_2 - w_3.$$

Так как $0 \leqslant C(2^n-1) \leqslant C_P$ и $C(P-1) \leqslant C_P$, следовательно $0 \leqslant w_1 \leqslant C_P$ и $0 \leqslant C_P - w_1 - w_2 - w_3 \leqslant C_P$, поэтому $0 \leqslant w_1 \leqslant C_P$ и $0 \leqslant w_1 + w_2 + w_3 \leqslant C_P$.

Лемма доказана. □

Лемма 2.5.2. *Если* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$ *в COK с набором модулей* $\{2^n-1, 2^{n+a}, 2^n+1\}$ *верны следующие утверждения:*

Если $a = 0$, *то* $0 \leqslant w_1 + w_2 \leqslant C_P$ *и* $0 \leqslant w_1 + w_2 + w_3 \leqslant C_P$.

Если $a \geqslant 1$, *то* $0 \leqslant w_1 + w_3 \leqslant C_P$ *и* $0 \leqslant 2^a w_1 + w_2 + (2^a-1)\,w_3 \leqslant C_P$.

Доказательство. Если $a = 0$, то $2^{n+a} = 2^n$. Посчитаем $C(2^n)$ и $C(2^n+1)$, получим:

$$C(2^n) = w_1 \left\lfloor \frac{2^n}{2^n-1} \right\rfloor + w_2 \left\lfloor \frac{2^n}{2^n} \right\rfloor + w_3 \left\lfloor \frac{2^n}{2^n+1} \right\rfloor = w_1 + w_2,$$

$$C(2^n+1) = w_1 \left\lfloor \frac{2^n+1}{2^n-1} \right\rfloor + w_2 \left\lfloor \frac{2^n+1}{2^n} \right\rfloor + w_3 \left\lfloor \frac{2^n+1}{2^n+1} \right\rfloor = w_1 + w_2 + w_3.$$

Учитывая, что $a \geqslant 1$ мы вычислим $C\left(2^n+1\right)$ и $C\left(2^{n+a}\right)$, получаем:

$$C\left(2^n+1\right) = w_1\left\lfloor\frac{2^n+1}{2^n-1}\right\rfloor + w_2\left\lfloor\frac{2^n+1}{2^{n+a}}\right\rfloor + w_3\left\lfloor\frac{2^n+1}{2^n+1}\right\rfloor = w_1+w_3,$$

$$C\left(2^{n+a}\right) = w_1\left\lfloor\frac{2^{n+a}}{2^n-1}\right\rfloor + w_2 + w_3\left\lfloor\frac{2^{n+a}}{2^n+1}\right\rfloor = 2^a w_1 + w_2 + \left(2^a-1\right)w_3.$$

Так как $\forall X \in [0,P) : 0 \leqslant C(X) \leqslant C_P$, следовательно $0 \leqslant w_1+w_2 \leqslant C_P, 0 \leqslant w_1+w_2+w_3 \leqslant C_P$, если $a=0$ и $0 \leqslant w_1+w_3 \leqslant C_P, 0 \leqslant 2^a w_1+w_2+\left(2^a-1\right)w_3 \leqslant C_P$, если $a \geqslant 1$.

Лемма доказана. □

Лемма 2.5.3. *Если* $\forall X \in [0,P) : 0 \leqslant C(X) \leqslant C_P$ *и* $C_P = 2^b$ *тогда* $w_1 \geqslant 1$.

Доказательство. Поскольку $C_P = 2^b$, то

$$C_P = w_1P_1 + w_2P_2 + w_3P_3 = 2^b.$$

Предположим, что $w_1 = 0$, тогда $\left(2^n-1\right)|2^b$, что неверно для всех $n \geqslant 2$. Следовательно, предположение неверно и $w_1 \neq 0$. Из Леммы 2.5.1 следует, что $0 \leqslant w_1 \leqslant C_P$ и $w_1 \neq 0$, поэтому $w_1 \geqslant 1$.

Лемма доказана. □

Теорема 2.5.1. *Если* $\forall X \in [0,P) : 0 \leqslant C(X) \leqslant C_P < P$ *и* $C_P = 2^b$, *то* $b \geqslant n+a+1$ *и* $w_1 = 2^{|b-a-1|_n}, w_2 = 0$ *и* $w_3 = -2^{|b+n-a-1|_n}$.

Доказательство. Из условия теоремы следует, что выполняется равенство:

$$C_P = w_1\left\lfloor\frac{P}{2^n-1}\right\rfloor + w_2\left\lfloor\frac{P}{2^{n+a}}\right\rfloor + w_3\left\lfloor\frac{P}{2^n+1}\right\rfloor = 2^b,$$

Следовательно, $w_1P_1 + w_2P_2 + w_3P_3 = 2^b$.

Случай 1. Если $a=0$, то C_P можно представить в следующем виде:

$$C_P = P_3\left(w_1+w_2+w_3\right) + \left(P_2-P_3\right)\left(w_1+w_2\right) + \left(P_1-P_2\right)w_1$$
$$= \left(2^n-1\right)2^n\left(w_1+w_2+w_3\right) + \left(2^n-1\right)\left(w_1+w_2\right) + \left(2^n+1\right)w_1.$$

Из Лемм 2.5.1, 2.5.2 и 2.5.3 следует: $w_1+w_2+w_3 \geqslant 0, w_1+w_2 \geqslant 0$ и $w_1 \geqslant 1$, поэтому: $C_P \geqslant 2^n+1$ и $b \geqslant n$. Обратим внимание, что

$$\begin{cases} w_1 \equiv 2^{b-1} \pmod{2^n-1}, \\ w_2 \equiv 0 \pmod{2^n}, \\ w_3 \equiv -2^{b+n-1} \pmod{2^n+1}. \end{cases}$$

Заметим, что при $b = n + 1$ и $w_1 = 1, w_2 = 0$ и $w_3 = -1$ условия Лемм 2.5.1, 2.5.2 и 2.5.3 выполняются.

$$\begin{aligned} C_P &= \left\lfloor \frac{(2^n - 1)\, 2^n\, (2^n + 1)}{2^n - 1} \right\rfloor - \left\lfloor \frac{(2^n - 1)\, 2^n\, (2^n + 1)}{2^n + 1} \right\rfloor \\ &= 2^n\, (2^n + 1) - (2^n - 1)\, 2^n = 2^{n+1}. \end{aligned}$$

Следовательно, $b = n + 1$.

Случай 2. Если $a \geqslant 1$, то C_P можно представить в следующем виде:

$$C_P = P_2\, (w_1 + w_2 + w_3) + (P_3 - P_2)\, (w_1 + w_3) + (P_1 - P_3)\, w_1 = \left(2^{2n} - 1\right) (w_1 + w_2 + w_3) + (2^n - 1)\left(2^{n+a} - 2^n - 1\right) (w_1 + w_3) + 2^{n+a+1} w_1.$$

Из Лемм 2.5.1, 2.5.2 и 2.5.3 следует, что: $w_1 + w_2 + w_3 \geqslant 0$, $w_1 + w_3 \geqslant 0$ и $w_1 \geqslant 1$, значит:

$$C_P \geqslant 2^{n+a+1}.$$

Следовательно, $b > n + a$.

Обратим внимание, что

$$\begin{cases} w_1 \equiv 2^{b-a-1} \pmod{2^n - 1}, \\ w_2 \equiv 0 \pmod{2^{n+a}}, \\ w_3 \equiv -2^{b+n-a-1} \pmod{2^n + 1}. \end{cases}$$

Заметим также, что при $b = n + a + 1, w_1 = 1, w_2 = 0$ и $w_3 = -1$ выполняются условия Лемм 2.5.1, 2.5.2 и 2.5.3.

$$\begin{aligned} C_P &= \left\lfloor \frac{(2^n - 1)\, 2^{n+a}\, (2^n + 1)}{2^n - 1} \right\rfloor - \left\lfloor \frac{(2^n - 1)\, 2^{n+a}\, (2^n + 1)}{2^n + 1} \right\rfloor \\ &= 2^{n+a}\, (2^n + 1) - (2^n - 1)\, 2^{n+a} = 2^{n+a+1}. \end{aligned}$$

Следовательно, $b = n + a + 1$.

Теорема доказана. □

Рассмотрим минимальную функцию ядра, удовлетворяющую Теореме 2.5.1 и покажем, что ее можно использовать для эффективной реализации определения знака числа в СОК.

Функция ядра и ее параметры примут вид:

$$\begin{aligned} C(X) &= \left\lfloor \frac{X}{2^n-1} \right\rfloor - \left\lfloor \frac{X}{2^n+1} \right\rfloor, \\ P &= (2^n-1)\,2^{n+a}\,(2^n+1), \\ C(P) &= 2^{n+a+1}, \\ \frac{P}{2} &= (2^n-1)\,2^{n+a-1}\,(2^n+1). \end{aligned}$$

Утверждение 2.5.1. *В COK* $\{2^n-1, 2^{n+a}, 2^n+1\}$, $\forall X \in [0,P) : C(X) \geqslant 0$.

Доказательство. Так как $\forall n : 2^n - 1 < 2^n + 1$, то

$$\frac{X}{2^n-1} > \frac{X}{2^n+1},$$

следовательно

$$\left\lfloor \frac{X}{2^n-1} \right\rfloor \geqslant \left\lfloor \frac{X}{2^n+1} \right\rfloor,$$

значит $\forall X \in [0,P] : C(X) \geqslant 0$.

Утверждение доказано.

□

Утверждение 2.5.2. $\forall X \in [0,P) : C(X) \leqslant 2^{n+a+1}$.

Доказательство. Рассмотрим функцию $G(P-X) = C(P) - C(X) = 2^{n+a+1} - C(X)$, мы имеем:

$$\begin{aligned} G(P-X) &= \left\lceil \frac{P-X}{2^n-1} \right\rceil - \left\lceil \frac{P-X}{2^n+1} \right\rceil \\ &= \left\lceil \frac{(2^{2n}-1)2^{n+a} - X}{2^n-1} \right\rceil - \left\lceil \frac{(2^{2n}-1)2^{n+a} - X}{2^n+1} \right\rceil \\ &= \left\lceil 2^{n+a}(2^n+1) - \frac{X}{2^n-1} \right\rceil - \left\lceil 2^{n+a}(2^n-1) - \frac{X}{2^n+1} \right\rceil \\ &= 2^{n+a}(2^n+1) + \left\lceil -\frac{X}{2^n-1} \right\rceil - 2^{n+a}(2^n-1) - \left\lceil -\frac{X}{2^n+1} \right\rceil \\ &= 2^{n+a+1} + \left\lfloor \frac{X}{2^n+1} \right\rfloor - \left\lfloor \frac{X}{2^n-1} \right\rfloor = C(P) - C(X). \end{aligned}$$

Пусть $P - X = Y$, тогда $Y \in (0,P]$ и функция $G(Y)$ примет вид:

$$G(Y) = \left\lceil \frac{Y}{2^n-1} \right\rceil - \left\lceil \frac{Y}{2^n+1} \right\rceil.$$

Так как $\forall n : 2^n - 1 < 2^n + 1$, то

$$\frac{Y}{2^n - 1} > \frac{Y}{2^n + 1},$$

следовательно

$$\left\lceil \frac{Y}{2^n - 1} \right\rceil \geqslant \left\lceil \frac{Y}{2^n + 1} \right\rceil.$$

значит $\forall Y \in (0, P] : G(Y) \geqslant 0$. Таким образом, $C(X) \leqslant 2^{n+a+1}$.

Утверждение доказано. □

Исследуем для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ свойства функции $C(X)$. Для этого докажем Теорему 2.5.2.

Теорема 2.5.2. $\forall X \in [0, P)$ *выполняются следующие условия:*

1. Если $C(X) > 2^{n+a}$, *тогда* $X > (2^n - 1)\, 2^{n+a-1} (2^n + 1)$.

2. Если $C(X) < 2^{n+a}$, *тогда* $X < (2^n - 1)\, 2^{n+a-1} (2^n + 1)$.

Доказательство. Случай 1. Если $X \geqslant \frac{P}{2} + 2^n - 1$ тогда $C(X) > 2^{n+a}$. Рассмотрим числа вида $Y = (2^n - 1)\, 2^{n+a-1} (2^n + 1) + 2^n - 1 + X$, где $0 \leqslant X < \frac{P}{2} - 2^n + 1$ мы получим:

$$\begin{aligned} C(Y) &= \left\lfloor \frac{(2^n - 1)\, 2^{n+a-1} (2^n + 1) + 2^n - 1 + X}{2^n - 1} \right\rfloor \\ &\quad - \left\lfloor \frac{(2^n - 1)\, 2^{n+a-1} (2^n + 1) + 2^n - 1 + X}{2^n + 1} \right\rfloor \\ &= 2^{n+a} + 1 + \left\lfloor \frac{X}{2^n - 1} \right\rfloor - \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor \\ &= 2^{n+a} + 1 + \left\lfloor \frac{X}{2^n - 1} \right\rfloor - \left\lfloor \frac{X}{2^n + 1} \right\rfloor + \left\lfloor \frac{X}{2^n + 1} \right\rfloor - \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor \\ &= 2^{n+a} + 1 + C(X) + \left\lfloor \frac{X}{2^n + 1} \right\rfloor - \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor. \end{aligned}$$

Если $\left\lfloor \frac{X}{2^n+1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^n+1} \right\rfloor$, тогда $C(Y) = 2^{n+a} + 1 + C(X)$. Из Утверждения 2.5.1 следует, что $C(X) \geqslant 0$, следовательно: $C(Y) > 2^{n+a}$.

Если $\left\lfloor \frac{X}{2^n+1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^n+1} \right\rfloor - 1$, то $C(Y) = 2^{n+a} + C(X), C(X) \geqslant 0$, следовательно: $C(Y) \geqslant 2^{n+a}$.

Случай 2. Если $X \leqslant \frac{P}{2} - 2^n + 1$ то $C(X) < 2^{n+a}$. Рассмотрим числа вида $G = (2^n - 1)\,2^{n+a-1}\,(2^n + 1) - 2^n + 1 - X$, где $0 \leqslant X \leqslant \frac{P}{2} - 2^n + 1$ мы получим:

$$\begin{aligned} C(G) &= \left\lfloor \frac{(2^n-1)\,2^{n+a-1}\,(2^n+1) - 2^n + 1 - X}{2^n - 1} \right\rfloor \\ &\quad - \left\lfloor \frac{(2^n-1)\,2^{n+a-1}\,(2^n+1) - 2^n + 1 - X}{2^n + 1} \right\rfloor \\ &= 2^{n+a} - 1 - \left\lfloor \frac{X}{2^n-1} \right\rfloor + \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor \\ &= 2^{n+a} - 1 - \left\lfloor \frac{X}{2^n-1} \right\rfloor + \left\lfloor \frac{X}{2^n+1} \right\rfloor - \left\lfloor \frac{X}{2^n+1} \right\rfloor + \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor \\ &= 2^{n+a} - 1 - C(X) - \left\lfloor \frac{X}{2^n+1} \right\rfloor + \left\lfloor \frac{X + 2^n - 1}{2^n + 1} \right\rfloor. \end{aligned}$$

Если $\left\lfloor \frac{X}{2^n+1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^n+1} \right\rfloor$, то $C(Y) = 2^{n+a} - 1 - C(X)$, следовательно: $C(G) < 2^{n+a}$.

Если $\left\lfloor \frac{X}{2^n+1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^n+1} \right\rfloor - 1$, то $C(Y) = 2^{n+a} - C(X)$, следовательно: $C(G) \leqslant 2^{n+a}$.

Теорема доказана. □

Согласно методам, предложенным в параграфе 2.2 представленная функция ядра будет иметь верхние критические ядра. Для того чтобы использовать формулу (2.6) рассмотрим следующую теорему.

Теорема 2.5.3. *В COK с основаниями* $\{2^n - 1, 2^{n+a}, 2^n + 1\}$, $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$ *с весами* $w_1 = 1, w_2 = 0, w_3 = -1$, *если при вычислении ядра числа* $X = (x_1, x_2, x_3)$ *по формуле* (2.6) *получится критическое значение* 0, *то в случае* $x_1 \geqslant x_3$ *имеем* $C(X) = 0$, *а в противном случае* $C(X) = C_P$.

Доказательство. Пусть $p_1 = t + q_1 = 2^n - 1, p_2 = t + q_2 = 2^{n+a}, p_3 = t + q_1 + r = 2^n + 1$ где $(q_1, q_2, r > 0)$, тогда для числа с нулевыми значениями функции ядра имеем

$$\frac{X - x_1}{t + q_1} - \frac{X - x_3}{t + q_1 + r} = 0,$$

или

$$X(t + q_1 + r) - x_1(t + q_1 + r) - Xt - Xq_1 + x_3t + x_3q_1 = 0,$$

$$Xr - x_1(t + q_1) - x_1r + x_3(t + q_1) = 0,$$

$$Xr = (t + q_1)(x_1 - x_3) + x_1r.$$

Так как $Xr > 0$, то заключаем что $x_1 - x_3 \geqslant 0$, откуда $x_1 \geqslant x_3$.

Теорема доказана. □

Таким образом, остатки числа с нулевыми значениями функции ядра должны удовлетворять условию $x_1 \geqslant x_3$.

Вычислим константы СОК с набором модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$.

$$\begin{aligned} C_P &= C(P) = 2^{n+a}(2^n+1) - (2^n-1)2^{n+a} = 2^{n+a+1}. \\ P_1 &= 2^{n+a}(2^n+1). \\ P_2 &= (2^n-1)(2^n+1). \\ P_3 &= (2^n-1)\,2^{n+a}. \end{aligned}$$

$$\begin{aligned} \left|P_1^{-1}\right|_{p_1} &= \left|\frac{1}{2^{n+a}(2^n+1)}\right|_{2^n-1} = \left|\frac{1}{2^{a+1}}\right|_{2^n-1} = \left|2^{n+a+1}\right|_{2^n-1} \\ &= \begin{cases} 2^{n-1} \ \text{ если } a = 0, \\ 2^{n-a-1} \text{ если } 1 \leqslant a < n. \end{cases} \end{aligned}$$

$$\begin{aligned} \left|P_2^{-1}\right|_{p_2} &= \left|\frac{1}{(2^n-1)(2^n+1)}\right|_{2^{n+a}} = \left|\frac{1}{2^{2n}-1}\right|_{2^{n+a}} = \left|2^{n+a}-1\right|_{2^{n+a}} \\ &= \begin{cases} 2^n - 1 \text{ если } a = 0, \\ 2^{n+a} - 1 \text{ если } 1 \leqslant a < n. \end{cases} \\ \left|P_3^{-1}\right|_{p_3} &= \left|\frac{1}{(2^n-1)\,2^{n+a}}\right|_{2^n+1} = \left|2^{n+1}+2^n+1\right|_{2^n+1} \\ &= \begin{cases} 2^n - 2^{n-1} + 1 \text{ если } a = 0, \\ 2^n - 2^{n-a-1} + 1 \text{ если } 1 \leqslant a < n. \end{cases} \end{aligned}$$

Если $a = 0$, тогда $\left|P_1^{-1}\right|_{p_1} = 2^{n-1}, \left|P_2^{-1}\right|_{p_2} = 2^n - 1, \left|P_3^{-1}\right|_{p_3} = 2^n - 2^{n-1} + 1$, следовательно,

$$
\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{2^{n-1}2^n(2^n+1)}{2^n-1} \right\rfloor - \left\lfloor \frac{2^{n-1}2^n\,(2^n+1)}{2^n+1} \right\rfloor \\
&= 2^{2n-1} + 2^n + 1 - 2^{n-1}2^n = 2^n + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^n-1)\,(2^n-1)\,(2^n+1)}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{(2^n-1)\,(2^n-1)\,(2^n+1)}{2^n+1} \right\rfloor = (2^n-1)\,(2^n+1) \\
&- (2^n-1)\,(2^n-1) = 2\,(2^n-1) = 2^{n+1} - 2, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{\left(2^n - 2^{n-1} + 1\right)(2^n-1)2^n}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{\left(2^n - 2^{n-1} + 1\right)(2^n-1)2^n}{2^n+1} \right\rfloor = \left(2^n - 2^{n-1} + 1\right)2^n - \left(2^{2n-1} - 1\right) \\
&= 2^{2n} - 2^{2n-1} + 2^n - 2^{2n-1} + 1 = 2^n + 1.
\end{aligned}
$$

Если $a \geqslant 1$, тогда $\left|P_1^{-1}\right|_{p_1} = 2^{n-a-1}, \left|P_2^{-1}\right|_{p_2} = 2^{n+a} - 1, \left|P_3^{-1}\right|_{p_3} = 2^n - 2^{n-a-1} + 1$, следовательно,

$$
\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{2^{n-a-1}2^{n+a}(2^n+1)}{2^n-1} \right\rfloor - \left\lfloor \frac{2^{n-a-1}2^{n+a}\,(2^n+1)}{2^n+1} \right\rfloor \\
&= 2^{2n} + 2^n + 1 - 2^{n-a-1}2^{n+a} = 2^n + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^{n+a}-1)\,(2^n-1)\,(2^n+1)}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{(2^{n+a}-1)\,(2^n-1)\,(2^n+1)}{2^n+1} \right\rfloor = 2^{n+a+1} - 2, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{\left(2^n - 2^{n-a-1} + 1\right)(2^n-1)2^{n+a}}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{\left(2^n - 2^{n-a-1} + 1\right)(2^n-1)2^{n+a}}{2^n+1} \right\rfloor \\
&= \left(2^{2n+a} - 2^{2n-1} + 2^{n+a}\right) - \left(2^{2n+a} - 2^{n+a} - 2^{2n-1} + 2^n - 1\right) \\
&= 2^{n+a+1} - 2^n + 1.
\end{aligned}
$$

Для определения знака числа с использованием минимальной функции ядра с заданными свойствами для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ предложен Алгоритм 12.

Алгоритм 12: Определение знака числа с использованием функции ядра для набора модулей $\{2^n-1, 2^{n+a}, 2^n+1\}$

Input: (x_1, x_2, x_3)
Data: $\{p_1, p_2, p_3\}$,
$P = (2^n-1)\,2^{n+a}\,(2^n+1)$,
$C_P = 2^{n+a+1}$,
$C\left(\frac{P}{2}\right) = 2^{n+a}$,
$N = n+1+a$,
$\{k_1, k_2, k_3\}$
Output: $sign$

```
C(X) = (k1 · x1 + k2 · x2 + k3 · x3) ∧ (2^N − 1)
if C(X) = 0 and x1 < x3 then
    C(X) = C_P
else
    C(X) = 0
if C(X) < C(P/2) then
    sign = 0
else
    sign = 1
```

Result: $sign$.

Рассмотрим Пример 2.5.3 определения знака числа для предложенной функции ядра.

Пример 2.5.3. *Для $n = 3$ и $a = 0$ имеем систему модулей $p_1 = 7, p_2 = 8, p_3 = 9$. Объем динамического диапазона $P = 7 \cdot 8 \cdot 9 = 504$. Функция ядра от динамического диапазона равна*

$$C_P = 2^{n+a+1} = 16 = 2^4.$$

Функция ядра от половины диапазона равна

$$C\left(\frac{P}{2}\right) = 2^{n+a} = 8.$$

Далее найдем коэффициенты k_i:

$$k_1 = 9, k_2 = 14, k_3 = 9.$$

Определим знак числа $(6,3,7)$. *Для этого найдем значение* $C(X)$:

$$C(X) = |9 \cdot 6 + 14 \cdot 3 + 9 \cdot 7|_{2^4} = |159|_{2^4} = |[10011111]_2|_{2^4} = [1111]_2 = 15.$$

Так как $15 > 8$, *следовательно число* $(6,3,7)$ *отрицательное.*

Таким образом, предложенный подход позволяет сократить размеры операндов, а также вычислительно сложную операцию нахождения остатка от деления можно заменить взятием N младших бит числа или же поразрядной конъюнкцией.

2.5.2 Алгоритм определения знака числа на основе функции ядра для набора модулей $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$

Рассмотрим минимальную функцию ядра Акушского с заданными свойствами для набора модулей $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$, где $0 \leqslant a < n$. Определим функцию ядра следующим образом:

$$C(X) = w_1 \left\lfloor \frac{X}{2^n - 1} \right\rfloor + w_2 \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor + w_3 \left\lfloor \frac{X}{2^{n+a}} \right\rfloor,$$

и $C_P = C(P) = w_1 P_1 + w_2 P_2 + w_3 P_3$, где $P = (2^n - 1)\left(2^{n+1} - 1\right) 2^{n+a}$ и $P_1 = \left(2^{n+1} - 1\right) 2^{n+a}$, $P_2 = (2^n - 1)\, 2^{n+a}$, $P_3 = (2^n - 1)(2^{n+1} - 1)$ и $w_1, w_2, w_3 \in \mathbb{Z}$.

Исследуем какие ограничения накладываются на коэффициенты w_1, w_2 и w_3 функции ядра $C(X)$, так чтобы функция удовлетворяла условию $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$. Для этого докажем Леммы 2.5.4 и 2.5.5.

Лемма 2.5.4. *Если* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$, *то* $\begin{cases} 0 \leqslant w_1 \leqslant C_P, \\ 0 \leqslant w_1 + w_2 + w_3 \leqslant C_P. \end{cases}$

Доказательство. Доказательство проводится аналогично Лемме 2.5.1.

Лемма доказана. □

Лемма 2.5.5. *Если* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$, *в КОК с набором модулей* $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$ *верны следующие утверждения:*

Если $a = 0$, *то* $0 \leqslant w_1 + w_3 \leqslant C_P$ *и* $0 \leqslant 2w_1 + w_2 + w_3 \leqslant C_P$.

Если $a \geqslant 1$, *то* $0 \leqslant 2w_1 + w_2 \leqslant C_P$ *и* $0 \leqslant 2^a w_1 + 2^{a-1} w_2 + w_3 \leqslant C_P$.

Доказательство. Если $a = 0$, то $2^{n+a} = 2^n$. Вычислим $C(2^n)$ и $C(2^{n+1} - 1)$, получим:

$$C(2^n) = w_1 + w_3,$$

$$C(2^{n+1} - 1) = 2w_1 + w_2 + w_3.$$

При условии, что $a \geqslant 1$ вычислим $C(2^{n+1} - 1)$ и $C(2^{n+a})$, получим:

$$C(2^{n+1} - 1) = 2w_1 + w_2,$$

$$C(2^{n+a}) = w_1 \left\lfloor \frac{2^{n+a}}{2^n - 1} \right\rfloor + w_2 \left\lfloor \frac{2^{n+a}}{2^{n+1} - 1} \right\rfloor + w_3 = 2^a w_1 + 2^{a-1} w_2 + w_3.$$

Так как $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$, следовательно $0 \leqslant w_1 + w_3 \leqslant C_P$, $0 \leqslant 2w_1 + w_2 + w_3 \leqslant C_P$, если $a = 0$ и $0 \leqslant 2w_1 + w_2 \leqslant C_P$, $0 \leqslant 2^a w_1 + 2^{a-1} w_2 + w_3 \leqslant C_P$, если $a \geqslant 1$.

Лемма доказана. □

Лемма 2.5.6. *Если* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P$ *и* $C_P = 2^b$, *то* $w_1 \geqslant 1$.

Доказательство. Так как $C_P = 2^b$, то

$$w_1 P_1 + w_2 P_2 + w_3 P_3 = 2^b.$$

Предположим, что $w_1 = 0$, то $(2^n - 1) | 2^b$, что неверно для всех $n \geqslant 2$. Следовательно, предположение неверно и $w_1 \neq 0$. Из Леммы 2.5.4 следует, что $0 \leqslant w_1 \leqslant C_P$ и $w_1 \neq 0$, значит $w_1 \geqslant 1$.

Лемма доказана. □

Теорема 2.5.4. *Если для функции* $C(X) = w_1 \left\lfloor \frac{X}{2^n - 1} \right\rfloor + w_2 \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor + w_3 \left\lfloor \frac{X}{2^{n+a}} \right\rfloor$ *выполняются следующие условия* $\forall X \in [0, P) : 0 \leqslant C(X) \leqslant C_P < P$ *и* $C_P = 2^b$ *то* $b \geqslant 2n + a$.

Доказательство. Рассмотрим два случая.

Случай 1. Если $a = 0$ то C_P можно представить в следующем виде:

$$C_P = 2^n(2^n - 1)(w_1 + w_2 + w_3) + (2^n - 1)^2(w_1 + w_3) + (2^{n+1} - 1)w_1. \quad (2.33)$$

Из Лемм 2.5.4, 2.5.5 и 2.5.6 следует, что: $w_1 + w_2 + w_3 \geqslant 0$, $w_1 + w_3 \geqslant 0$ и $w_1 \geqslant 1$, значит: $C_P \geqslant 2^{n+1} - 1$ и $b \geqslant n + 1$.

Учитывая, что

$$\begin{cases} w_1 \equiv 2^b \pmod{2^n - 1}, \\ w_2 \equiv -2^{b+2} \pmod{2^{n+1} - 1}, \\ w_3 \equiv 0 \pmod{2^n}. \end{cases}$$

то $w_1 = q_1 (2^n - 1) + 2^{|b|_n}$, $w_2 = q_2 \left(2^{n+1} - 1\right) - 2^{|b+2|_{n+1}}$ и $w_3 = q_3 2^n$.

Обратим внимание, что при $b = 2n$ и $w_1 = 1, w_2 = -1$ и $w_3 = 0$ выполняются условия Лемм 2.5.4, 2.5.5 и 2.5.6.

Предположим, что существуют w_1, w_2 и w_3, удовлетворяющие условиям Лемм 2.5.4, 2.5.5 и 2.5.6 и $n + 1 \leqslant b < 2n$. Тогда $b = n + |b|_n$ и $|b + 2|_{n+1} = |b|_n$, следовательно: $w_1 = q_1(2^n - 1) + 2^{|b|_n}$, $w_2 = q_2(2^{n+1} - 1) - 2^{|b|_n}$ и $w_3 = q_3 2^n$.

Из (2.33) следует: что $(2^{n+1} - 1)w_1 \leqslant 2^{2n}$ значит:

$$0 \leqslant (2^{n+1} - 1)(q_1(2^n - 1) + 2^{|b|_n}) \leqslant 2^{2n},$$

$$-2^{|b|_n} \leqslant q_1(2^n - 1) \leqslant \frac{2^{2n}}{2^{n+1} - 1} - 2^{|b|_n} < 2^{n-1}.$$

Следовательно, $q_1 = 0$ и $w_1 = 2^{|b|_n}$.

Из (2.33) следует, что если $w_1 + w_3 \geqslant 1$ то $C_P \geqslant 2^{2n}$, что противоречит предположению $n+1 \leqslant b < 2n$, следовательно $w_1 + w_3 = 0$ и $2^{|b|_n} + q_3 2^n = 0$. Так как $2^{|b|_n} \neq 0$ и $1 + q_3 2^{n-|b|_n} \neq 0$ то $2^{|b|_n} \left(1 + q_3 2^{n-|b|_n}\right) = 2^{|b|_n} + q_3 2^n \neq 0$, пришли к противоречию. Следовательно, не существует w_1, w_2 и w_3 удовлетворяющим условиям Лемм 2.5.4, 2.5.5, 2.5.6 и $n + 1 \leqslant b < 2n$. Значит, если $\forall X \in [0, P)$: $0 \leqslant C(X) \leqslant C_P < P$ и $C_P = 2^b$ то $b \geqslant 2n$.

Случай 2. Если $a \geqslant 1$ то C_P можно представить в следующем виде:

$$C_P = (2^n - 1)(2^{n+1} - 1)(w_1 + w_2 + w_3) + \\ + (2^n - 1)\left(2^{n+a} - 2^{n+1} + 1\right)(2w_1 + w_2) + \left(2^{2n+1} + (2^a - 3)\, 2^n + 1\right) w_1. \quad (2.34)$$

Из Лемм 2.5.4, 2.5.5 и 2.5.6 следует, что: $w_1 + w_2 + w_3 \geqslant 0$, $2w_1 + w_2 \geqslant 0$ и $w_1 \geqslant 1$, значит:

$$C_P \geqslant 2^{2n+1} + (2^a - 3)\, 2^n + 1 \geqslant 2^{2n+1} - 2^n + 1 > 2^{2n}.$$

Следовательно, $b > 2n$.

Учитывая, что

$$\begin{cases} w_1 \equiv 2^{b-a} \pmod{2^n - 1}, \\ w_2 \equiv -2^{b-a+2} \pmod{2^{n+1} - 1}, \\ w_3 \equiv 0 \pmod{2^{n+a}}. \end{cases}$$

то $w_1 = q_1(2^n - 1) + 2^{|b-a|_n}$, $w_2 = q_2(2^{n+1} - 1) - 2^{|b-a+2|_{n+1}}$ и $w_3 = q_3 2^{n+a}$.

Обратим внимание, что при $b = 2n + a$, $w_1 = 1$, $w_2 = -1$ и $w_3 = 0$ выполняются условия Лемм 2.5.4, 2.5.5 и 2.5.6. Предположим, что существуют w_1, w_2 и w_3 удовлетворяющим условиям Лемм 2.5.4, 2.5.5, 2.5.6 и $2n + 1 \leqslant b < 2n + a$. Следовательно, b можно представить в виде: $b = n + a + |b - a|_n$ и $|b - a + 2|_{n+1} = |b - a|_n$, $w_2 = q_2(2^{n+1} - 1) - 2^{|b-a|_n}$.

Из (2.34) следует, что:

$$0 \leqslant \left(2^{2n+1} + (2^a - 3)2^n + 1\right) w_1 < 2^{2n+a}.$$

Следовательно, $q_1 = 0$ и $w_1 = 2^{|b-a|_n}$. Значит

$$0 \leqslant (2^n - 1)\left(2^{n+a} - 2^{n+1} + 1\right)(2w_1 + w_2) + \left(2^{2n+1} + (2^a - 3)\,2^n + 1\right) w_1 \leqslant C_P.$$

Так как $2w_1 + w_2 = q_2\left(2^{n+1} - 1\right) + 2^{|b-a+2|_{n+1}}$ и $w_1 = 2^{|b-a|_n}$, то

$$\begin{gathered} (2^n - 1)\left(2^{n+a} - 2^{n+1} + 1\right)(2w_1 + w_2) + \left(2^{2n+1} + (2^a - 3)\,2^n + 1\right) w_1 \\ = q_2 P + 2^{|b-a|_n + 2n + a}, \\ 0 \leqslant q_2 P + 2^{|b-a|_n + 2n + a} < 2^{2n+a}, \end{gathered}$$

и $-1 < q_2 < 0$ в целых числах решения нет.

Теорема доказана. □

Рассмотрим функцию ядра, удовлетворяющую Теореме 2.5.4 и покажем, что ее можно использовать для эффективной реализации функции знака числа в СОК.

Функция ядра примет вид

$$C(X) = \left\lfloor \frac{X}{2^n - 1} \right\rfloor - \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor.$$

И будет иметь следующие параметры

$$P = (2^n - 1)(2^{n+1} - 1)2^{n+a},$$

$$C(P) = 2^{2n+a},$$
$$\frac{P}{2} = (2^n - 1)\left(2^{n+1} - 1\right) 2^{n+a-1}.$$

Утверждение 2.5.3. $\forall X{\in}[0, P) : C(X) \geqslant 0$.

Доказательство. Доказательство проводится аналогично Утверждению 2.5.1. Утверждение доказано. □

Утверждение 2.5.4. $\forall X{\in}[0, P) : C(X) \leqslant 2^{2n+a}$.

Доказательство. Доказательство проводится аналогично Утверждению 2.5.2. Утверждение доказано. □

Теорема 2.5.5. $\forall X{\in}[0, P)$ *выполняются следующие условия:*

1. *Если* $C(X) > 2^{2n+a-1}$, *то* $X > (2^n - 1)(2^{n+1} - 1)2^{n+a-1}$.
2. *Если* $C(X) < 2^{2n+a-1}$, *то* $X < (2^n - 1)(2^{n+1} - 1)2^{n+a-1}$.

Доказательство. Случай 1. (Если $X \geqslant \frac{P}{2}+2^n-1$, то $C(X) > 2^{2n+a-1}$). Рассмотрим числа вида $Y = (2^n-1)(2^{n+1}-1)2^{n+a-1}+2^n-1+X$, где $0 \leqslant X < \frac{P}{2}-2^n+1$ мы получим:

$$C(Y) = \left\lfloor \frac{(2^n - 1)\left(2^{n+1} - 1\right) 2^{n+a-1} + 2^n - 1 + X}{2^n - 1} \right\rfloor$$
$$- \left\lfloor \frac{(2^n - 1)\left(2^{n+1} - 1\right) 2^{n+a-1} + 2^n - 1 + X}{2^{n+1} - 1} \right\rfloor$$
$$= 2^{2n+a-1} + 1 + \left\lfloor \frac{X}{2^n - 1} \right\rfloor - \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor + \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor - \left\lfloor \frac{X + 2^n - 1}{2^{n+1} - 1} \right\rfloor$$
$$= 2^{2n+a-1} + 1 + C(X) + \left\lfloor \frac{X}{2^{n+1} - 1} \right\rfloor - \left\lfloor \frac{X + 2^n - 1}{2^{n+1} - 1} \right\rfloor.$$

Если $\left\lfloor \frac{X}{2^{n+1}-1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^{n+1}-1} \right\rfloor$, то $C\left(Y\right) = 2^{2n+a-1}+1+C(X)$. Из Утверждения 2.5.4 следует, что $C(X) \geqslant 0$, следовательно: $C\left(Y\right) > 2^{2n+a-1}$.

Если $\left\lfloor \frac{X}{2^{n+1}-1} \right\rfloor = \left\lfloor \frac{X+2^n-1}{2^{n+1}-1} \right\rfloor - 1$, то $2^n \leqslant |Xt|_{2^{n+1}-1} < 2^{n+1} - 1$. Покажем, что если $2^n \leqslant |X|_{2^{n+1}-1} < 2^{n+1} - 1$ то $C(X) \geqslant 1$. Так как $2^n \leqslant |X|_{2^{n+1}-1} < 2^{n+1} - 1$ то $X = \xi\left(2^{n+1} - 1\right) + \delta$, где $\xi \leqslant 0$ и $2^n \leqslant \delta < 2^{n+1} - 1$. Вычислим $C(X)$, имеем:

$$C(X) = \left\lfloor \frac{\xi\left(2^{n+1} - 1\right) + \delta}{2^n - 1} \right\rfloor - \left\lfloor \frac{\xi\left(2^{n+1} - 1\right) + \delta}{2^{n+1} - 1} \right\rfloor.$$

Так как $\left|\frac{\xi\left(2^{n+1}-1\right)+\delta}{2^n-1}\right| = 2\xi + \left|\frac{\xi+\delta}{2^n-1}\right|$ и $\left|\frac{\xi\left(2^{n+1}-1\right)+\delta}{2^{n+1}-1}\right| = \xi$, то

$$C(X) = \xi + \left\lfloor \frac{\xi+\delta}{2^n-1} \right\rfloor.$$

Учитывая, что $\xi+\delta \geqslant 2^n$, то $\left\lfloor \frac{\xi+\delta}{2^n-1} \right\rfloor \geqslant 1$, следовательно, $C(X) \geqslant 1$, значит, $C(Y) > 2^{2n+a-1}$.

Случай 2. Если $X \leqslant \frac{P}{2} - 2^n + 1$ то $C(X) < 2^{2n+a-1}$. Рассмотрим числа вида $G = (2^n-1)\left(2^{n+1}-1\right)2^{n+a-1} - 2^n + 1 - X$, где $0 \leqslant X \leqslant \frac{P}{2} - 2^n + 1$ мы получим:

$$\begin{aligned} C(G) &= \left\lfloor \frac{(2^n-1)(2^{n+1}-1)2^{n+a-1} - 2^n + 1 - X}{2^n-1} \right\rfloor \\ &- \left\lfloor \frac{(2^n-1)(2^{n+1}-1)2^{n+a-1} - 2^n + 1 - X}{2^{n+1}-1} \right\rfloor \\ &= 2^{2n+a-1} - 1 - \left\lfloor \frac{X}{2^n-1} \right\rfloor + \left\lfloor \frac{X+2^n-1}{2^{n+1}-1} \right\rfloor. \end{aligned}$$

Так как $\forall X \geqslant 2^n - 1$ выполняется неравенство

$$\frac{X}{2^n-1} > \frac{X+2^n-1}{2^{n+1}-1},$$

то

$$\left\lceil \frac{X}{2^n-1} \right\rceil \geqslant \left\lceil \frac{X+2^n-1}{2^{n+1}-1} \right\rceil,$$

следовательно,$C(G) < 2^{2n+a-1}$.

Если $0 \leqslant X < 2^n - 1$, то $\left\lceil \frac{X}{2^n-1} \right\rceil = 0$ и $\left\lceil \frac{X+2^n-1}{2^{n+1}-1} \right\rceil = 0$, следовательно, $C(G) = 2^{2n+a-1} - 1 < 2^{2n+a-1}$.

Случай 3. Если $\frac{P}{2} - 2^n + 1 < X < \frac{P}{2} + 2^n - 1$ то $C(X) = 2^{2n+a-1}$. Рассмотрим числа вида $U = (2^n-1)(2^{n+1}-1)2^{n+a-1} + X$, мы получим:

$$\begin{aligned} C(U) &= \left\lfloor \frac{(2^n-1)(2^{n+1}-1)2^{n+a-1} + X}{2^n-1} \right\rfloor - \left\lfloor \frac{(2^n-1)(2^{n+1}-1)2^{n+a-1} + X}{2^{n+1}-1} \right\rfloor \\ &= 2^{2n+a-1} + \left\lfloor \frac{X}{2^n-1} \right\rfloor - \left\lfloor \frac{X}{2^{n+1}-1} \right\rfloor. \end{aligned}$$

Если $0 \leqslant X < 2^n - 1$, то $\left\lfloor \frac{X}{2^n-1} \right\rfloor = 0$ и $\left\lfloor \frac{X}{2^{n+1}-1} \right\rfloor = 0$, следовательно, $C(U) = 2^{2n+a-1}$.

Если $-2^n + 1 < X < 0$, то $\left\lfloor \frac{X}{2^n-1} \right\rfloor = -1$ и $\left\lfloor \frac{X}{2^{n+1}-1} \right\rfloor = -1$, следовательно, $C(U) = 2^{2n+a-1}$. $\square$

Для вычисления функции ядра в случае возникновения верхних критических ядер рассмотрим следующую теорему.

Теорема 2.5.6. *В COK с основаниями* $\left\{2^n-1, 2^{n+1}-1, 2^{n+a}\right\}$ *и весами* $w_1 = 1, w_2 = -1, w_3 = 0$, *если при вычислении ядра числа* $X = (x_1, x_2, x_3)$ *по формуле* (2.6) *получится критическое значение* 0, *то в случае* $x_1 \geqslant x_2$ *имеем* $C(X) = 0$, *а в противном случае* $C(X) = C_P$.

Доказательство. Доказательство проводится аналогично Теореме 2.5.3.

Теорема доказана. □

Вычислим константы COK с набором модулей $\left\{2^n-1, 2^{n+1}-1, 2^{n+a}\right\}$.

$$C_P = C(P) = \left(2^{n+1}-1\right)2^{n+a} - \left(2^n-1\right)2^{n+a} = 2^{2n+a},$$

$$P_1 = (2^{n+1}-1)2^{n+a}, P_2 = (2^n-1)2^{n+a},$$
$$P_3 = (2^n-1)(2^{n+1}-1),$$

$$\left|P_1^{-1}\right|_{p_1} = \left|\frac{1}{(2^{n+1}-1)2^{n+a}}\right|_{2^n-1} = \left|\frac{1}{2^a}\right|_{2^n-1} = \left|2^{n-a}\right|_{2^n-1} = \begin{cases} 1 \text{ если } a = 0, \\ 2^{n-a} \text{ если } 1 \leqslant a \leqslant n. \end{cases}$$

$$\left|P_2^{-1}\right|_{p_2} = \left|\frac{1}{(2^n-1)2^{n+a}}\right|_{2^{n+1}-1} = \left|\frac{-1}{2^{n+a-1}}\right|_{2^{n+1}-1} = \left|\frac{-2^{2n+2}}{2^{n+a-1}}\right|_{2^{n+1}-1}$$

$$= \left|2^{n+1} - 2^{n-a+3} - 1\right|_{2^{n+1}-1} = \begin{cases} 2^{n+1} - 2^2 - 1 \text{ если } a = 0, \\ 2^{n+1} - 2 - 1 \text{ если } a = 1, \\ 2^{n+1} - 2 \text{ если } a = 2, \\ 2^{n+1} - 2^{n-a+3} - 1 \text{ если } 3 \leqslant a \leqslant n. \end{cases}$$

$$\left|P_3^{-1}\right|_{p_3} = \left|\frac{1}{\left(2^n-1\right)\left(2^{n+1}-1\right)}\right|_{2^{n+a}} = \left|2^{n+1} + 2^n + 1\right|_{2^{n+a}}$$

$$= \begin{cases} 1 \text{ если } a = 0, \\ 2^n + 1 \text{ если } a = 1, \\ 2^{n+1} + 2^n + 1 \text{ если } 2 \leqslant a \leqslant n. \end{cases}$$

Рассчитаем константы k_i необходимые для расчета функции ядра.

Если $a > 2$, то $|P_1^{-1}|_{p_1} = 2^{n-a}$, $|P_2^{-1}|_{p_2} = 2^{n+1} - 2^{n-a+3} - 1$, $|P_3^{-1}|_{p_3} = 2^{n+1} + 2^n + 1$, следовательно,

$$
\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{(2^{n+1}-1)2^{n+a}2^{n-a}}{2^n-1} \right\rfloor - \left\lfloor \frac{(2^{n+1}-1)2^{n+a}2^{n-a}}{2^{n+1}-1} \right\rfloor \\
&= \left\lfloor \frac{(2^{n+1}-1)2^{2n}}{2^n-1} \right\rfloor - 2^{2n} = 2^{2n+1} + 2^n + 1 - 2^{2n} \\
&= (2^n-1)(2^{2n+1}+2^n+1) + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^{n+1}-2^{n-a+3}-1)(2^n-1)2^{n+a}}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{(2^{n+1}-2^{n-a+3}-1)(2^n-1)2^{n+a}}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1}-2^{n-a+3}-1)2^{n+a} - \left\lfloor \frac{(2^{n+1}-2^{n-a+3}-1)(2^n-1)2^{n+a}}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1}-2^{n-a+3}-1)2^{n+a} - (2^n-1)2^{n+a} - \left\lfloor -\frac{(2^{n+1}-2)2^{2n+1}}{2^{n+1}-1} \right\rfloor \\
&= \left(2^{n+1}-2^{n-a+3}-1\right)2^{n+a} - (2^n-1)2^{n+a} - (2^{n+1}+1-2^{2n+2}) \\
&= 2^{2n+a+1} - 2^{2n+3} - 2^{n+a} - 2^{2n+a} + 2^{n+a} - 2^{n+1} - 1 + 2^{2n+2} \\
&= 2^{2n+a} - 2^{2n+2} - 2^{n+1} - 1, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{(2^n-1)(2^{n+1}-1)(2^{n+1}+2^n+1)}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{(2^n-1)\left(2^{n+1}-1\right)(2^{n+1}+2^n+1)}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1}-1)\left(2^{n+1}+2^n+1\right) - (2^n-1)(2^{n+1}+2^n+1) \\
&= 2^{2n+1} + 2^{2n} + 2^n.
\end{aligned}
$$

Если $a = 2$, то $|P_1^{-1}|_{p_1} = 2^{n-2}$, $|P_2^{-1}|_{p_2} = 2^{n+1} - 2$, $|P_3^{-1}|_{p_3} = 2^{n+1} + 2^n + 1$, следовательно,

$$\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{(2^{n+1}-1)2^{2n}}{2^n - 1} \right\rfloor - \left\lfloor \frac{(2^{n+1}-1)2^{2n}}{2^{n+1}-1} \right\rfloor \\
&= 2^{2n} + 2^n + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^{n+1}-2)(2^n-1)2^{n+2}}{2^n - 1} \right\rfloor \\
&- \left\lfloor \frac{(2^{n+1}-2)(2^n-1)2^{n+2}}{2^{n+1}-1} \right\rfloor \\
&= \left(2^{n+1}-2\right)2^{n+2} - (2^n-1)\,2^{n+2} + 2^{n+1} - 1 \\
&= 2^{2n+2} - 2^{n+1} - 1, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{(2^n-1)(2^{n+1}-1)(2^{n+1}+2^n+1)}{2^n - 1} \right\rfloor \\
&- \left| \frac{(2^n-1)\left(2^{n+1}-1\right)(2^{n+1}+2^n+1)}{2^{n+1}-1} \right| \\
&= (2^{n+1}-1)(2^{n+1}+2^n+1) - (2^n-1)(2^{n+1}+2^n+1) \\
&= 2^{2n+1} + 2^{2n} + 2^n.
\end{aligned}$$

Если $a = 1$, то $|P_1^{-1}|_{p_1} = 2^{n-1}$, $|P_2^{-1}|_{p_2} = 2^{n+1} - 2 - 1$, $|P_3^{-1}|_{p_3} = 2^n + 1$, следовательно,

$$\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{(2^{n+1}-1)2^{2n}}{2^n - 1} \right\rfloor - \left\lfloor \frac{(2^{n+1}-1)2^{2n}}{2^{n+1}-1} \right\rfloor \\
&= 2^{2n} + 2^n + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^{n+1}-2-1)(2^n-1)2^{n+1}}{2^n - 1} \right\rfloor \\
&- \left\lfloor \frac{(2^{n+1}-2-1)(2^n-1)2^{n+1}}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1}-2-1)2^{n+1} - (2^n-1)2^{n+1} + 2^{n+1} - 1 \\
&= 2^{2n+1} - 2^{n+1} - 1, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{(2^n-1)(2^{n+1}-1)(2^n+1)}{2^n - 1} \right\rfloor \\
&- \left\lfloor \frac{(2^n-1)\left(2^{n+1}-1\right)(2^n+1)}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1}-1)(2^n+1) - (2^n-1)(2^n+1) = (2^n+1)2^n = 2^{2n} + 2^n.
\end{aligned}$$

Если $a = 0$, то $|P_1^{-1}|_{p_1} = 1$, $|P_2^{-1}|_{p_2} = 2^{n+1} - 2^2 - 1$, $|P_3^{-1}|_{p_3} = 1$, следовательно,

$$
\begin{aligned}
k_1 &= C\left(\left|P_1^{-1}\right|_{p_1} \cdot P_1\right) = \left\lfloor \frac{(2^{n+1}-1)2^n}{2^n-1} \right\rfloor - \left\lfloor \frac{(2^{n+1}-1)2^n}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1} - 2 + 1)2^n = (2^n - 1)(2^{n+1} + 1) + 1, \\
k_2 &= C\left(\left|P_2^{-1}\right|_{p_2} \cdot P_2\right) = \left\lfloor \frac{(2^{n+1}-2^2-1)(2^n-1)2^n}{2^n-1} \right\rfloor \\
&- \left\lfloor \frac{(2^{n+1}-2^2-1)(2^n-1)2^n}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1} - 2^2 - 1)2^n - (2^n - 1)2^n - \left\lfloor -\frac{(2^{n+1}-2)2^{n+1}}{2^{n+1}-1} \right\rfloor \\
&= (2^{n+1} - 2^2 - 1)2^n - (2^n - 1)2^n + 2^{n+1} - 1 \\
&= 2^{2n} - 2^{n+1} - 1, \\
k_3 &= C\left(\left|P_3^{-1}\right|_{p_3} \cdot P_3\right) = \left\lfloor \frac{(2^n-1)(2^{n+1}-1)}{2^n-1} \right\rfloor - \left\lfloor \frac{(2^n-1)(2^{n+1}-1)}{2^{n+1}-1} \right\rfloor \\
&= 2^{n+1} - 1 - 2^n + 1 = 2^n.
\end{aligned}
$$

Алгоритм определения знака числа для с использованием функции ядра для представленных модулей подобен Алгоритму 12.

Алгоритмы определения знака числа, основанные на предложенных функциях ядра, обладают преимуществом перед классическими методами благодаря уменьшению размера операндов, а также снижению вычислительной сложности операции нахождения остатка от деления по модулю C_P с $O(n^2)$ до $O(n)$.

2.6 Построение функций ядра Акушского для операции сравнения чисел в СОК

В различных практических приложениях часто требуется сравнивать числовые значения. Операция сравнения чисел играет ключевую роль в реализации алгоритмов для защиты информации [95], цифровой обработки сигналов [33], облачных вычислений [113].

Алгоритм сравнения чисел в системе остаточных классов включает два основных этапа. На первом этапе производится расчет позиционной характери-

стики для каждого числа в модулярном представлении. Затем, на втором этапе, выполняется сравнение полученных позиционных характеристик этих чисел.

Рассмотрим систему модулей $p_1, p_2, \ldots, p_n$, упорядоченных по возрастанию: $p_1 < p_2 < \cdots < p_n$. Функция ядра Акушского определена формулой (2.1).

При необходимости использования функции ядра для сравнения чисел в системе остаточных классов, необходимо чтобы она была монотонно возрастающей. Монотонность обеспечивается за счет выбора весов функции ядра. Исследуем условия, обеспечивающие монотонность функции ядра. Для этого рассмотрим следующую теорему, приведенную в работе [3].

Теорема 2.6.1. *Функция $C(X)$ — монотонно возрастающая тогда и только тогда, когда $\forall i \in [1;n] : w_i \geqslant 0$.*

Алгоритм 13 демонстрирует применение монотонной функции ядра Акушского для сравнения чисел в системе остаточных классов.

Далее сосредоточимся на исследовании свойств функции ядра Акушского необходимых для построения минимальной функции для операции сравнения чисел, у которой выполняется условие $C(P) = C_P = 2^N$, где $C(P) = \sum_i^n w_i P_i$.

Если $w_1 = w_2 = \cdots = w_n = 0$, то $C(X) = 0$ и ее значение не несет никакой информации о позиции числа X на числовой прямой. Следовательно, для того чтобы $C(X)$ можно было использовать для операции сравнения чисел необходимо, чтобы хотя бы одно значение $w_j > 0$.

Так как хотя бы одно значение $w_j > 0$, то $C_P \geqslant P_n = \frac{P}{p_n}$, следовательно $2^N \geqslant P_n$, значит $N \geqslant \lceil \log_2 P_n \rceil$.

Утверждение 2.6.1. *Если $n \geqslant 3$, то P_n — не является степенью двойки.*

Доказательство. Так как $n \geqslant 3$, то P_n можно представить в следующем виде $P_n = p_1 \cdot Q$, где $p_1 \geqslant 2$, $p_2 | Q$, $Q \geqslant p_2 > p_1$ и $\gcd(p_1, Q) = 1$. Предположим, что $n \geqslant 3$ и P_n — степенью двойки, тогда P_n представляется в виде $P_n = 2^a$, где $a \in N$. Так как $p_1 | P_n$, $p_1 \geqslant 2$ и $P_n = 2^a$ то, следовательно, $p_1 = 2^b$, $b \in N$, $1 \leqslant b \leqslant a$ и $Q = 2^{a-b}$. Если $a = b$, то $Q = 1$ и пришли к противоречию, что $Q \geqslant p_2 > p_1 \geqslant 2$. Если $1 \leqslant b < a$ то $2 | Q$ и $2 | p_1$ следовательно $2 | \gcd(p_1, Q)$. Так как 2 не делит 1, то пришли к противоречию. Следовательно, если $n \geqslant 3$, то P_n — не является степенью двойки.

Утверждение доказано. □

Алгоритм 13: Сравнение чисел, представленных в СОК, с использованием монотонной функции ядра

Input: $X = (x_1, x_2, \ldots, x_n), Y = (y_1, y_2, \ldots, y_n)$
Data: $\{p_1, p_2, \ldots, p_n\}$,
$\{w_1, w_2, \ldots, w_n\}$,
$B_i = P_i \cdot \left|P_i^{-1}\right|_P$,
$k_i = C(B_i)$,
$C_P = \sum_{i=1}^{n} w_i \cdot P_i, i = 1, 2, \ldots, n$
Output: *sign*

```
C(X) = |Σ_{i=1}^{n} k_i · x_i|_{C_P}, C(Y) = |Σ_{i=1}^{n} k_i · y_i|_{C_P}
if C(X) < C(Y) then
    return «01»
else if C(X) > C(Y) then
    return «10»
else
    if x_k < y_k then
        return «01»
    else if x_k > y_k then
        return «10»
    else
        return «00»
```

Из Утверждения 2.6.1 следует, что если $n \geqslant 3$ и $C_P = 2^N$, то $C_P > P_n$ и существуют такие $w_j \geqslant 1$ и $w_k \geqslant 1$, где $j \neq k$.

Теорема 2.6.2. *Условие* $r(2^N) = 0$ *является необходимым и достаточным для выполнения неравенства:*

$$\sum_{i=1}^{n} \left| \left|2^N\right|_{p_i} \cdot \left|P_i^{-1}\right|_{p_i} \right|_{p_i} P_i < P.$$

Доказательство. Из предположения о выполнении указанного неравенства, вычислим $r(2^N)$, получаем:

$$r\left(2^N\right) = \left\lfloor \frac{1}{P} \sum_{i=1}^{n} \left| \left|2^N\right|_{p_i} \cdot \left|P_i^{-1}\right|_{p_i} \right|_{p_i} P_i \right\rfloor = 0.$$

Учитывая что $r(2^N)=0$, тогда, используя свойства модулярной арифметики, можем записать:

$$\left|\sum_{i=1}^{n}\left|\left|2^N\right|_{p_i}\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i\right|_P=\sum_{i=1}^{n}\left|\left|2^N\right|_{p_i}\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i-r\left(2^N\right)P$$

$$=\sum_{i=1}^{n}\left|\left|2^N\right|_{p_i}\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i<P.$$

Теорема доказана. □

Лемма 2.6.1. $r\left(2X\right)=2r\left(X\right)-\sum_{i=1}^{n}\left\lfloor\frac{2}{p_i}\left|x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}\right\rfloor$.

Доказательство. Исходя из определения функции $r\left(X\right)$, вычислим:

$$r\left(2X\right)=\left\lfloor\frac{1}{P}\sum_{i=1}^{n}\left|\left|2x_i\right|_{p_i}\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i\right\rfloor$$

$$=\left\lfloor\frac{1}{P}\left(\sum_{2x_i<p_i}\left|2x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i+\sum_{2x_i\geqslant p_i}\left|(2x_i-p_i)\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i\right)\right\rfloor$$

$$=\left\lfloor\frac{1}{P}\sum_{i=1}^{n}\left|2x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i\right\rfloor.$$

Так как

$$\left|2x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}=2x_i\cdot\left|P_i^{-1}\right|_{p_i}-\left\lfloor\frac{2}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor p_i$$

$$=2\left|x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}+p_i\left(2\left\lfloor\frac{1}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor-\left\lfloor\frac{2}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor\right),$$

мы получим

$$r\left(2X\right)=\left\lfloor\frac{2}{P}\sum_{i=1}^{n}\left|x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i+\sum_{i=1}^{n}\left(2\left\lfloor\frac{1}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor-\left\lfloor\frac{2}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor\right)\right\rfloor.$$

Так как

$$\sum_{i=1}^{n}\left|x_i\cdot\left|P_i^{-1}\right|_{p_i}\right|_{p_i}P_i=X+r(X)P,$$

следовательно

$$r(2X)=\left\lfloor\frac{2X}{P}+2r(X)+\sum_{i=1}^{n}\left(2\left\lfloor\frac{1}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor-\left\lfloor\frac{2}{p_i}\cdot x_i\cdot\left|P_i^{-1}\right|_{p_i}\right\rfloor\right)\right\rfloor.$$

Учитывая условие $2X < P$, мы имеем:

$$r(2X) = 2r(X) + \sum_{i=1}^{n} \left(2 \left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor - \left\lfloor \frac{2}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor \right).$$

Так как $\left\lfloor \frac{2}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor = \left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor + \left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} + \frac{1}{2} \right\rfloor$ получим

$$r(2X) = 2r(X) + \sum_{i=1}^{n} \left(\left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor - \left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} + \frac{1}{2} \right\rfloor \right).$$

Представим $\frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i}$ в виде сумы целой и дробной части числа $\frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} = \left\lfloor \frac{1}{p_i} \cdot x_i \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor + \frac{1}{p_i} \cdot \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i}$, мы получим:

$$r(2X) = 2r(X) - \sum_{i=1}^{n} \left(\left\lfloor \frac{1}{p_i} \cdot \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} + \frac{1}{2} \right\rfloor \right).$$

Обратим внимание, что

$$\left\lfloor \frac{1}{p_i} \cdot \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} + \frac{1}{2} \right\rfloor = \begin{cases} 0, \text{ если } 2 \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} < p_i, \\ 1, \text{ если } 2 \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \geqslant p_i. \end{cases}$$

Таким образом,

$$r(2X) = 2r(X) - \sum_{2\left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \geqslant p_i} 1 = 2r\left(X\right) - \sum_{i=1}^{n} \left\lfloor \frac{2}{p_i} \left| x_i \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor.$$

Лемма доказана. □

Теорема 2.6.3. $r(2^N) = 2^N r(1) - \sum_{i=1}^{n} \left\lfloor \frac{2^N}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor$.

Доказательство. Для доказательства используем принцип математической индукции. Для этого покажем, что равенство выполняется, при $N = 1$. Из Леммы 2.6.1 следует, что

$$r(2) = 2r(1) - \sum_{i=1}^{n} \left\lfloor \frac{2}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor$$

Предположим, что равенство верно при $N = K$

$$r(2^K) = 2^K r(1) - \sum_{i=1}^{n} \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor$$

Докажем, что верно при $N = K+1$. Используя Лемму 2.6.1, мы получим

$$r(2^{K+1}) = 2r(2^K) - \sum_{i=1}^{n} \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor$$

Подставляя результат предположения, получим:

$$r(2^{K+1}) = 2^{K+1} r(1) - 2\sum_{i=1}^{n} \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor - \sum_{i=1}^{n} \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor$$

$$= 2^{K+1} r(1) - \sum_{i=1}^{n} \left(2 \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor \right)$$

Упростим выражение

$$2 \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor$$

$$= 2 \cdot \frac{2^K \left| P_i^{-1} \right|_{p_i} - \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i}}{p_i} + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor$$

$$= \frac{2^{K+1} \left| P_i^{-1} \right|_{p_i}}{p_i} - \frac{2 \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i}}{p_i} + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor .$$

Учитывая, что $\frac{2}{p_i} \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i} = \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor + \frac{1}{p_i} \left| 2 \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right|_{p_i}$ мы получим:

$$2 \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor = \frac{2^{K+1} \left| P_i^{-1} \right|_{p_i}}{p_i} - \frac{1}{p_i} \left| 2 \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right|_{p_i} .$$

Используя свойства сравнения чисел $\left| 2 \left| 2^K \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right|_{p_i} = \left| 2^{K+1} \left| P_i^{-1} \right|_{p_i} \right|_{p_i}$, мы получим:

$$2 \left\lfloor \frac{2^K}{p_i} \left| P_i^{-1} \right|_{p_i} \right\rfloor + \left\lfloor \frac{2}{p_i} \left| 2^K \cdot \left| P_i^{-1} \right|_{p_i} \right|_{p_i} \right\rfloor = \frac{2^{K+1} \left| P_i^{-1} \right|_{p_i}}{p_i} - \frac{1}{p_i} \left| 2^{K+1} \left| P_i^{-1} \right|_{p_i} \right|_{p_i}$$

$$= \left\lfloor \frac{2^{K+1} \left| P_i^{-1} \right|_{p_i}}{p_i} \right\rfloor .$$

Значит

$$r(2^{K+1}) = 2^{K+1} r\,(1) - \sum_{i=1}^{n} \left\lfloor \frac{2^{K+1}}{p_i} \cdot \left| P_i^{-1} \right|_{p_i} \right\rfloor .$$

Теорема доказана. □

Сформулируем Теорему 2.6.4 необходимую для метода поиска функций ядра Акушского для операции сравнения чисел.

Теорема 2.6.4. *Если* $r\left(2^N\right) = 0$, *то* $N_{min} = N - \log_2 \gcd(2^N, w_1, w_2, \ldots, w_n)$, $w_i^* = \frac{w_i}{\gcd(2^N, w_1, w_2, \ldots, w_n)}$ *и* $2^{N_{min}} = \sum_{i=1}^{n} w_i^* P_i$.

Доказательство. Из условия $r(2^N) = 0$ и Теоремы 2.6.2, следует, что существуют такие $w_i \geqslant 0$, что выполняется равенство

$$2^N = \sum_{i=1}^{n} w_i P_i.$$

Разделим, левую и правую часть равенства на $\gcd(2^N, w_1, w_2, \ldots, w_n)$ получим, что

$$\frac{2^N}{\gcd(2^N, w_1, w_2, \ldots, w_n)} = \sum_{i=1}^{n} \frac{w_i}{\gcd(2^N, w_1, w_2, \ldots, w_n)} P_i.$$

Так как $\gcd(2^N, w_1, w_2, \ldots, w_n)$ — есть степень двойки, то его можно представить в виде:

$$gcd(2^N, w_1, w_2, \ldots, w_n) = 2^{\log_2 \gcd(2^N, w_1, w_2, \ldots, w_n)},$$

и формула примет вид:

$$2^{N - \log_2 \gcd(2^N, w_1, w_2, \ldots, w_n)} = \sum_{i=1}^{n} \frac{w_i}{\gcd(2^N, w_1, w_2, \ldots, w_n)} P_i.$$

Пусть $w_i^* = \frac{w_i}{\gcd(2^N, w_1, w_2, \ldots, w_n)}$. Заметим, что $\frac{w_i}{\gcd(2^N, w_1, w_2, \ldots, w_n)} \in N$. Значит, существуют такие константы w_i^*, для которых выполняется равенство

$$2^{N_{min}} = \sum_{i=1}^{n} w_i^* P_i.$$

Докажем минимальность. Предположим, что существует $N < N_{min}$ для которого выполняется равенство:

$$2^N = \sum_{i=1}^{n} w_i P_i.$$

Следовательно,

$$2^{N_{min}} = \sum_{i=1}^{n} 2^{N - N_{min}} w_i P_i.$$

то есть $w_i^* = 2^{N-N_{min}} w_i P_i$ и $\gcd\left(2^N, w_1^*, w_2^*, \ldots, w_n^*\right) \geqslant 2^{N-N_{min}} > 1$, с другой стороны $\gcd\left(2^N, w_1^*, w_2^*, \ldots, w_n^*\right) = 1$, следовательно пришли к противоречию.

Теорема доказана. □

Рассмотрим пример поиска минимальной функции ядра, подходящей для сравнения чисел в СОК, для которой выполняется условие $C_P = 2^N$.

Пример 2.6.1. *Найдем N_{min} для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$.*

Вычислим константы P, C_P, $\left|P_i^{-1}\right|_{p_i}$, мы имеем

$$P = (2^n - 1)2^{n+a}(2^n + 1) = 2^{3n+a} - 2^{n+a},$$

$$C_P = (2^n + 1)2^{n+a} + (2^n - 1)2^{n+a} = 2^{2n+a+1},$$

$$\left|P_1^{-1}\right|_{p_1} = \left|\frac{1}{(2^n + 1)2^{n+a}}\right|_{2^n - 1} = 2^{n-a-1},$$

$$\left|P_2^{-1}\right|_{p_2} = \left|\frac{1}{(2^n - 1)(2^n + 1)}\right|_{2^{n+a}} = 2^{n+a} - 1,$$

$$\left|P_3^{-1}\right|_{p_3} = \left|\frac{1}{(2^n - 1)2^{n+a}}\right|_{2^n+1} = \left|\frac{1}{2^{a+1}}\right|_{2^n+1} = 2^n - 2^{n-a-1} + 1.$$

Проверим, выполнение условия Теоремы 2.6.2 для $N = \lfloor \log_2 P \rfloor = 3n + a - 1$, для этого вычислим:

$$\sum_{i=1}^{n} \left\lfloor \frac{2^N}{p_i} \left|P_i^{-1}\right|_{p_i} \right\rfloor = \left\lfloor \frac{2^{3n+a-1}}{2^n - 1} 2^{n-a-1} \right\rfloor$$

$$+ \left\lfloor \frac{2^{3n+a-1}}{2^{n+a}} \left(2^{n+a} - 1\right) \right\rfloor + \left\lfloor \frac{2^{3n+a-1}}{2^n + 1} \left(2^n - 2^{n-a-1} + 1\right) \right\rfloor$$

$$= 2^{3n-2} + 2^{2n-2} + 2^{n-2} + 2^{3n+a-1} - 2^{2n-1} + 2^{3n+a-1} - 2^{3n-2} + 2^{2n-2} - 2^{n-2} = 2^{3n+a}.$$

Вычислим $r(1)$, получим:

$$r(1) = \left\lfloor \sum_{i=1}^{n} \frac{1}{p_i} \cdot \left|P_i^{-1}\right|_{p_i} \right\rfloor = \left\lfloor \frac{2^{n-a-1}}{2^n - 1} + \frac{2^{n+a} - 1}{2^{n+a}} + \frac{2^n - 2^{n-a-1} + 1}{2^n + 1} \right\rfloor$$

$$= 2 + \left\lfloor \frac{2^{n-a-1}}{2^n - 1} - \frac{1}{2^{n+a}} - \frac{2^{n-a-1}}{2^n + 1} \right\rfloor = 2 + \left\lfloor \frac{2^{n-a}}{2^{2n} - 1} - \frac{1}{2^{n+a}} \right\rfloor = 2 + \left\lfloor \frac{1}{2^{3n+a} - 2^{n+a}} \right\rfloor = 2.$$

Используя Теорему 2.6.3 вычислим $r\left(2^N\right)$, мы имеем:

$$r(2^N) = 2^N r(1) - \sum_{i=1}^{n} \left\lfloor \frac{2^N}{p_i} \left|P_i^{-1}\right|_{p_i} \right\rfloor = 2^{3n+a-1} \cdot 2 - 2^{3n+a} = 0.$$

Следовательно, условия Теоремы 2.6.2 выполняются и существуют такие w_i для которых выполняется равенство:

$$2^N = \sum_{i=1}^{n} w_i P_i.$$

Вычислим значения w_i используя формулу $w_i = \left|\left|2^N\right|_{p_i} \cdot \left|P_i^{-1}\right|_{p_i}\right|_{p_i}$, мы получим:

$$w_1 = \left|\left|2^N\right|_{p_1} \cdot \left|P_1^{-1}\right|_{p_1}\right|_{p_1} = \left|\left|2^{3n+a-1}\right|_{2^n-1} \cdot 2^{n-a-1}\right|_{2^n-1} = \left|2^{a-1} \cdot 2^{n-a-1}\right|_{2^n-1} = 2^{n-2},$$

$$w_2 = \left|\left|2^N\right|_{p_2} \cdot \left|P_2^{-1}\right|_{p_2}\right|_{p_2} = \left|\left|2^{3n+a-1}\right|_{2^{n+a}} \cdot (2^{n+a}-1)\right|_{2^{n+a}} = 0,$$

$$w_3 = \left|\left|2^N\right|_{p_3} \cdot \left|P_3^{-1}\right|_{p_3}\right|_{p_3} = \left|\left|2^{3n+a-1}\right|_{2^n+1} \cdot (2^n - 2^{n-a-1} + 1)\right|_{2^n+1}$$
$$= \left|2^{a-1} \cdot 2^{n-a-1}\right|_{2^n-1} = 2^{n-2}.$$

Вычислим $\gcd(2^N, w_1, w_2, \ldots, w_n)$*, получим:*

$$\gcd(2^N, w_1, w_2, \ldots, w_n) = \gcd\left(2^{3n+a-1}, 2^{n-2}, 0, 2^{n-2}\right) = 2^{n-2}.$$

Следовательно, используя Теорему 2.6.4 найдем N_{min} и w_i^*:

$$N_{min} = N - \log_2 \gcd\left(2^N, w_1, w_2, \ldots, w_n\right) = 3n + a - 1 - n + 2 = 2n + a + 1$$

$$w_1^* = \frac{w_1}{\gcd(2^N, w_1, w_2, \ldots, w_n)} = \frac{2^{n-2}}{2^{n-2}} = 1,$$

$$w_2^* = \frac{w_2}{\gcd(2^N, w_1, w_2, \ldots, w_n)} = \frac{0}{2^{n-2}} = 0,$$

$$w_3^* = \frac{w_3}{\gcd(2^N, w_1, w_2, \ldots, w_n)} = \frac{2^{n-2}}{2^{n-2}} = 1.$$

В таблице 5 приведены некоторые наборы модулей, и оптимальные веса с целью построения функций ядра для операции сравнения чисел.

В случае если не существует такого $N \leqslant \lfloor \log_2 P \rfloor$ для которого выполнялось условие Теоремы 2.6.2, то, следовательно, необходимо искать $N > \lfloor \log_2 P \rfloor$. В параграфе 3.2.2 рассмотрим алгориты поиска $N > \lfloor \log_2 P \rfloor$ для которого выполняется равенство $C_P = \sum_{i=1}^{n} w_i P_i = 2^N$.

Рассмотрим пример сравнения чисел с минимальной функцией ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$.

Таблица 5 — Веса функций ядра для сравнения чисел

№	Набор модулей	w_i	N_{min}	$\lfloor \log_2 P \rfloor$
1	$\{2^n-1, 2^{n+a}, 2^n+1\}$	{1,0,1}	$2n+a+1$	$3n+a-1$
2	$\{2^{n+1}-1, 2^n, 2^n-1\}$	$\{2^n-1,0,1\}$	$3n$	$3n$
3	$\{2^{2n}, 2^n-1, 2^{n-1}-1\}$	$\{0,2^{n-1}-1,1\}$	$4n-2$	$4n-2$

Пример 2.6.2. *Для* $n = 3, a = 0$ *модули* $p_1 = 7, p_2 = 8, p_3 = 9$. *В соответствии с таблицей 5 веса* $w_1 = 1, w_2 = 0, w_3 = 1$. *Размер динамического диапазона* $P = 2040$ *и* $C_P = 128$. *Ортогональные базисы равны:*

$$B_1 = 288, B_2 = 441, B_3 = 441.$$

Далее найдем коэффициенты $k_i = C(B_i)$.

$$k_1 = 73, k_2 = 112, k_3 = 71.$$

Сравним число $X = (1, 6, 6)$ *с числом* $Y = (1, 3, 7)$. *Для этого вычислим* $C(X)$ *и* $C(Y)$.

$$C(X) = |73 \cdot 1 + 112 \cdot 6 + 71 \cdot 6|_{128} = 19,$$

$$C(Y) = |73 \cdot 1 + 112 \cdot 3 + 71 \cdot 7|_{128} = 10.$$

Так как $C(X) > C(Y)$, *следовательно* $X > Y$.

Таким образом, предложенный метод позволяет найти минимальную функцию ядра с заданными свойствами для сравнения чисел в системе остаточных классов, что, в свою очередь, приводит к уменьшению размера операндов и позволяет выполнять операции по модулю 2^N.

2.7 Выводы по второй главе

Система остаточных классов представляет собой одну из непозиционных систем счисления, которая обеспечивает высокую производительность благодаря отсутствию необходимости выполнять переносы между разрядами при проведении арифметических операций. Тем не менее, некоторые операции в

этой системе, известные как немодульные, требуют значительных вычислительных ресурсов и нуждаются в для повышения эффективности.

Представлены математические основы функции ядра Акушского, которая необходима для определения позиционной характеристики числа в системе остаточных классов. Рассмотрена проблема критических ядер функции ядра Акушского, которая ограничивает использование оптимальной формы функции ядра. Разработан метод на основе Теоремы 2.2.1 позволяющий определить критические ядра. Метод с использованием Теоремы 2.2.1 в среднем на 99% быстрее предложенного алгоритма усеченного перебора для определения критических ядер.

Исследованы методы обратного преобразования чисел из системы остаточных классов в позиционную систему счисления с использованием Китайской теоремы об остатках, ее приближенный вариант, обобщенной позиционной системы счисления, диагональной функции и функции ядра. В результате был предложен новый метод на основе Китайской теоремы об остатках и ранга числа вычисляемого на основе функции ядра. Доказано равенство рангов числа определенного из Китайской теоремы об остатках и функции ядра Акушского, в случае отсутствия критических ядер.

Исследован алгоритм итерационного деления в СОК. Разработаны модификации алгоритма итерационного деления, в которых применяется новые методы вычисления функции ядра от половины числа.

Были исследованы различные методы определения знака числа в СОК. Для специальных наборов модулей вида $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ и $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$ построены минимальные функции ядра Акушского с заданными свойствами. Разработанные на основе этих функций алгоритмы обладают преимуществом перед классическими методами, которое заключается в уменьшении размера операндов и снижении вычислительной сложности операции нахождения остатка от деления с $O(n^2)$ до $O(n)$.

Проведен анализ алгоритма сравнения чисел с использованием функции ядра Акушского. Рассмотрена проблема монотонности функции ядра, необходимая для операции сравнения чисел в СОК. Предложен метод построения функций ядра для операции сравнения чисел, где $C_P = 2^N$ и $N \leqslant \lfloor \log_2 P \rfloor$.

Таким образом, предложенные методы и алгоритмы позволяют реализовать набор немодульных операций, необходимых для создания эффективной системы обработки данных, работающей в системе остаточных классов.

Глава 3. Разработка программного комплекса выполнения немодульных операций модулярной арифметики для проектирования высокоскоростных туманных узлов

3.1 Программный комплекс для проектирования узла туманных вычислений, работающего в системе остаточных классов

Для решения задач проектирования высокоскоростного узла туманных вычислений, функционирующего на основе модулярной арифметики, был разработан специализированный программный комплекс. Он ориентирован на автоматизацию ключевых этапов проектирования — от выбора базиса СОК до оптимизации немодульных операций.

Главное преимущество СОК заключается в возможности параллельной реализации операций сложения, вычитания и умножения по независимым модулям. Модульная структура позволяет эффективно распределять вычисления между несколькими устройствами, открывая возможности для решений в криптографии и цифровой обработке сигналов. На рисунке 3.1 представлена схема работы вычислительной системы, использующей СОК.

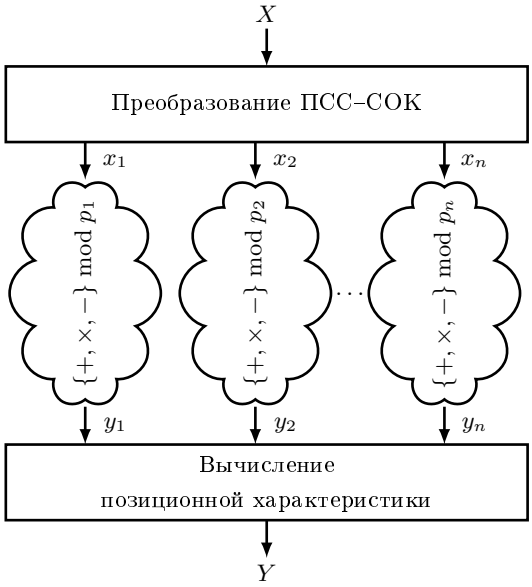

Рисунок 3.1 — Структурная схема вычислительной системы на основе модулярной арифметики

Вычислительная система, работающая с использованием модулярной арифметики, включает несколько блоков обработки данных. Входной блок выполняет преобразование чисел из ПСС в модулярное представление. Основные вычислительные блоки системы реализуют модульные операции сложения, вычитания и умножения в соответствии с свойствами СОК. Особое значение имеет блок определения позиционных характеристик, который обеспечивает выполнение немодульных операций.

Для реализации высокоскоростной системы на основе методов и алгоритмов, представленных во второй главе, был создан программный комплекс для работы в системе остаточных классов. Его архитектура представлена на рисунке 3.2. В открытом доступе есть библиотека NTL для языка программирования C++, предназначенная для работы с различными областями теории чисел. Доступные методы этой библиотеки, поддерживающие вычисления в СОК, были использованы в качестве аналогов для сравнения, отсутствующие методы были реализованы самостоятельно.

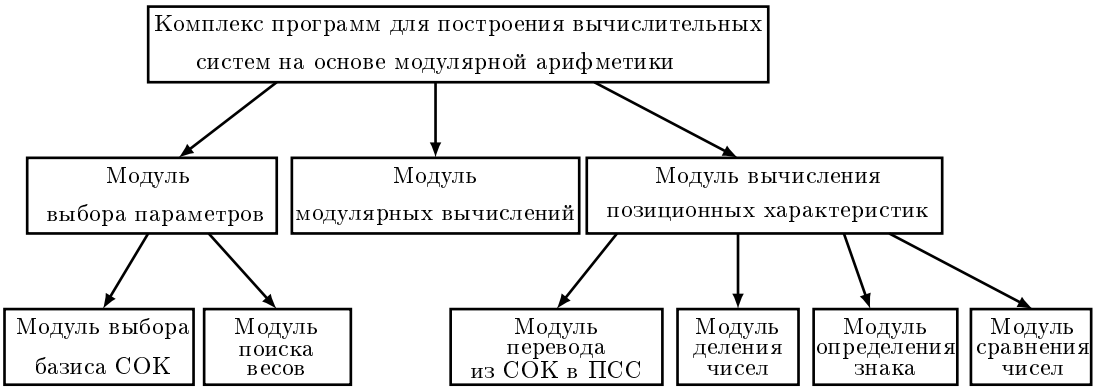

Рисунок 3.2 — Структура программного комплекса для построения вычислительных систем на основе модулярной арифметики

Проектирование архитектуры программного комплекса вычислительной системы, работающей на основе модулярной арифметики, начинается с определения подходящего базиса системы остаточных классов. В параграфе 3.2.1, будет представлен анализ влияния выбора базиса на вычислительную сложность модульных операций в СОК. Выбор базиса влияет на диапазон значений, с которыми можно работать в СОК. Кроме того, для каждого базиса необходимо подобрать свой набор оптимальных весов для функции ядра Акушского, как это было показано в параграфах 2.5.1, 2.5.2 и 2.6.

На первом этапе разработки высокоскоростной вычислительной системы на основе модулярной арифметики, необходимо выбрать набор модулей СОК. Согласно рисунку 3.2, эта задача решается через использование модуля, предназначенного для выбора базиса СОК. Для выбора модулей в параграфе 3.2.1 разработан алгоритм построения компактных базисов системы остаточных классов специального вида. Алгоритм предназначен для получения оснований СОК заданной длины. Из оснований, полученных данной программой, можно определить базис СОК, который отвечает критерию компактности, что позволяет увеличить скорость модульных операций СОК.

Следующим шагом проектирований вычислительной системы является выбор весов для вычисления функции ядра Акушского, которая служит инструментом для определения позиционной характеристики числа в СОК. Для наборов модулей специального вида, можно пользоваться методами, предложенными в параграфах 2.5.1, 2.5.2 и 2.6. При построении компактных базисов, количество оснований СОК в нем может быть достаточно большим, чтобы вычислять веса аналитически. Для данной задачи был разработан модуль поиска весов (рис. 3.2), представленный программой [16], которая позволяет выбирать веса для функции ядра с использованием генетического алгоритма. Выходом программы является набор весов для функции ядра Акушского, для которой выполняется условие $C(P) = 2^N$ и у которой отсутствуют критические ядра. В параграфе 3.2.2 представлены результаты поиска оптимальных весов с помощью метода Монте-Карло и генетического алгоритма.

На следующем этапе проектирования вычислительной системы, основанной на модулярной арифметике, производится реализация модульных вычислений, выполняемых параллельно и независимо для каждого основания, входящего в базис системы остаточных классов. Для обеспечения этих вычислений разработано программное обеспечение, использующее модуль распределенных модулярных вычислений представленного на рисунке 3.2. В качестве примера таких операций можно выбрать сложение, вычитание и умножение, находящие широкое применение в криптографических алгоритмах, системах цифровой обработки сигналов, а также нейронных сетях. Моделирование данных операций рассмотрено в параграфе 3.3.

С использованием функции ядра Акушского, можно найти позиционную характеристику числа в СОК, за это отвечает модуль вычисления позиционных характеристик (рис. 3.2). Для перевода в позиционную систему счисления мож-

но использовать методы, рассмотренные в параграфе 2.3. Реализован алгоритм обратного преобразования из СОК в ПСС предложенный в параграфе 2.3.1, использующий КТО и ранг числа, вычисляемый с использованием функции ядра Акушского. С целью реализации деления в СОК был разработан модуль деления чисел (см. рис. 3.2), которой реализован программами для ЭВМ [14; 17; 18]. Модуль определения знака числа разработан на основе алгоритмов, описанных в параграфах 2.5.1, 2.5.1 и реализован в программе [11]. Модуль сравнения чисел разработан на основе метода, предложенного в параграфе 2.6 и реализован в программе для ЭВМ [12]. Результаты моделирования немодульных операций приведены в параграфах 3.4, 3.5, 3.6, 3.7.

Рассмотрим разработанные модули, реализующих методы и алгоритмы, описанные в главе 2. Для моделирования использовалась следующая тестовая инфраструктура: Intel Core i7-7700HQ с тактовой частотой 2.80 ГГц, 8 ГБ оперативной памяти DDR4 с частотой 1196 МГц, твердотельный накопитель емкостью 512 ГБ, Windows 10 Home Edition. Для моделирования использованы языки программирования высокого уровня C++ и Python.

3.2 Модуль выбора параметров системы остаточных классов

3.2.1 Построение компактных базисов системы остаточных классов

Выбор набора модулей очень важен для достижения подходящей реализации СОК. Набор модулей влияет на всю архитектуру СОК [24; 66]. Широкое применение получили наборы модулей специального вида. Наиболее популярный из них это набор $2^n - 1, 2^n, 2^n + 1$ [86]. Худший модуль определяет общую задержку арифметического канала. Следует отметить, что худшим является модуль с наибольшей сложностью реализации. Таким модулем, как правило, является наибольший модуль в наборе общего или специального вида. За исключением случаев, когда наибольший модуль представим в виде степени двойки.

В таблице 6 представлены наиболее распространенные наборы модулей специального вида.

Таблица 6 — Наборы модулей специального вида

Номер набора	Ссылка	Набор модулей	Год
1	[108]	$\{2^n-1, 2^n, 2^n+1\}$	1967
2	[94]	$\{2n-1, 2n, 2n+1\}$	1995
3	[93]	$\{2^{2n}+1, 2^n+1, 2^n-1\}$	1997
4	[59]	$\{2^n-1, 2^n, 2^{n-1}-1\}$	1998
5	[76]	$\{2^n-1, 2^n, 2^{n+1}-1\}$	1999
6	[85]	$\{2^n-1, 2^n, 2^{2n+1}-1\}$	2008
7	[56]	$\{2^n-1, 2^n, 2^{2n}+1\}$	2008
8	[31]	$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$	2008
9	[62]	$\{3^n-2, 3^n-1, 3^n\}$	2007
10	[37]	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$	1999
11	[116]	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	2000
12	[40]	$\{2^n-1, 2^n, 2^n+1, 2^{2n}-1\}$	2003
13	[32]	$\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$	2004
14	[119]	$\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$	2008
15	[49]	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$	2009
16	[49]	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$	2009
17	[84]	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$	2010
18	[91]	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$	2014
19	[91]	$\{2^k, 2^n-1, 2^n+1, 2^{n-1}-1\}$	2014
20	[60]	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$	2005
21	[41]	$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$	2007
22	[83]	$\{2^{n/2}-1, 2^n, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$	2009
23	[92]	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	2012
24	[92]	$\{2^n-1, 2^{n+\beta}, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	2012
25	[50]	$\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \ldots, 2^n-k_f, 2^n+k_f\}$	2018

С целью исследования эффективности наборов модулей из таблицы 6 проведен анализ производительности данных наборов для арифметических операция сложения (add), вычитания (sub) и умножения (mult), с использованием Алгоритма 1.

Построены наборы модулей специального вида с размером динамического диапазона от 8 до 32 бит и шагом в 8 бит. Данные наборы представлены в таблице 21, приложения Б. Для каждой проверяемой операции предварительно генерируется массив из 10^4 случайных пар чисел в заданном динамическом диапазоне. Затем проводится 10^6 измерений, в каждом из которых выполняются вычисления со всеми парами из этого массива. В качестве результата берется среднее время выполнения всех измерений. Результаты моделирования представлены в таблице 22, приложения Б.

Для оценки эффективности выполнения операций в различных разрядностях наборов данных были вычислена сумма временных затрат для каждого типа набора, в рамках каждой операции. Полученные результаты представлены

в таблице 7, которая позволяет провести сравнительный анализ и сформировать итоговый рейтинг производительности.

Таблица 7 — Рейтинг наборов модулей специального вида, мкс

Количество модулей	Модули	Сумма времени		
		add +	sub −	mult ×
3	$\{2^n-1, 2^n, 2^n+1\}$	**576.939**	**576.662**	**579.223**
	$\{2n-1, 2n, 2n+1\}$	578.01	581.072	581.107
	$\{2^{2n}+1, 2^n+1, 2^n-1\}$	581.941	582.546	581.986
	$\{2^n-1, 2^n, 2^{n-1}-1\}$	586.097	589.225	589.975
	$\{2^n-1, 2^n, 2^{n+1}-1\}$	587.2	589.821	589.464
	$\{2^n-1, 2^n, 2^{2n+1}-1\}$	585.746	593.878	595.303
	$\{3^n-2, 3^n-1, 3^n\}$	587.806	594.804	597.451
	$\{2^n-1, 2^n, 2^{2n}+1\}$	586.998	594.901	598.437
	$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$	589.578	595.543	596.703
4	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$	591.188	**594.515**	**595.603**
	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$	**589.208**	594.883	599.481
	$\{2^n-1, 2^n, 2^n+1, 2^{2n}-1\}$	595	599.453	602.168
	$\{2^n-1, 2^n+1, 2^n-3, 2^n+3\}$	600.761	605.005	606.402
	$\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$	603.245	607.207	609.97
	$\{2^n-1, 2^n+1, 2^n, 2^{2n}+1\}$	612.781	622.684	623.981
	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$	616.187	620.672	627.686
	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$	615.55	625.294	630.348
	$\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$	620.039	625.347	627.064
	$\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$	620.398	627.011	631.516
5-6	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$	**630.668**	**632.565**	**639.107**
	$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$	636.571	637.999	642.16
	$\{2^{n/2}-1, 2^n, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$	649.521	648.973	655.339
	$\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	654.289	660.579	666.882
	$\{2^n-1, 2^{n+\beta}, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1, 2^{n\pm1}+1\}$	657.865	659.579	664.351
	$\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-k_1, 2^n+k_1, \ldots, 2^n-k_f, 2^n+k_f\}$	670.667	673.175	675.877

Исходя из данных, приведенных в таблице 7, можно сделать следующие выводы.

Для трехмодульных базисов в сложении лучшие результаты показали наборы $\{2^n-1, 2^n, 2^n+1\}$ и $\{2n-1, 2n, 2n+1\}$, они в среднем быстрее остальных на 0.73%. В операции вычитания лучший результат показал набор $\{2^n-1, 2^n, 2^n+1\}$, он быстрее других трехмодульных наборов на 0.81%. В умно-

жении лучший результат также показал набор $\{2^n-1, 2^n, 2^n+1\}$, превосходя остальные на 1.02%.

Для четырехмодульных базисов в сложении лучшие результаты показали наборы $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ и $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$, они в среднем быстрее других четырехмодульных наборов на 0.49%. В операции вычитания лучший результат показал набор $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, он быстрее других четырехмодульных наборов на 0.69%. В умножении лучший результат также показал набор $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, превосходя остальные на 0.64%.

Для пяти и шестимодульных базисов в сложении лучшие результаты показал набор $\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$, он в среднем быстрее остальных пяти- и шестимодульных наборов на 4.39%. В операции вычитания лучший результат показал набор $\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$, он быстрее остальных пяти и шестимодульных наборов на 4.15%. В умножении также лучший результат показал набор $\{2^n-1, 2^n, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$, он в среднем быстрее остальных пяти и шестимодульных наборов на 3.89%.

Для исследуемых динамических диапазонов набор $\{2^n-1, 2^n, 2^n+1\}$ демонстрирует лучшие результаты во всех операциях. Четырехмодульные базисы показывают меньшее преимущество в сравнении с трехмодульными, но их использование может быть оправдано при расширении динамического диапазона.

Для программной реализации СОК возможно использование наборов модулей общего вида. За счет большего количества модулей, можно достичь большей степени параллелизма. Однако для эффективной реализации наборов модулей общего вида необходимо выполнение условия компактности.

Определение 3.2.1. *Компактным набор модулей системы остаточных классов является набор, для которого выполняется условие* $p_n < 2 \cdot p_1$.

Для построения компактных наборов модулей можно использовать метод построения простых чисел который используется в стандарте СТБ 1176.2-99 (и прекратившем действии российском стандарте ГОСТ Р 34.10-94), который базируется на теореме Диемитко [5].

Теорема 3.2.1. *Пусть* $n = qR + 1$, *где* q — *простое нечетное число,* R — *четное,* $R < 4(q+1)$, *т.е.* $n < (2q+1)^2$.

Если найдется $a < n$:

1) $a^{n-1} \equiv 1 \mod n$;

2) $a^{\frac{n-1}{q}} \not\equiv 1 \mod n$, *то* n — *простое число.*

Таким образом, имея простое число q, то, перебирая четные числа R, строим числа $n = qR + 1$ и испытываем их на простоту согласно теореме Диемитко, пока не получим простое число. По полученному числу можно построить еще одно простое число.

Алгоритм 14 позволяет получить большее простое число p, имеющее битовую длину $L(p) = t$, начиная с меньшего простого числа q, длина которого составляет $L(q) = \lceil \frac{t}{2} \rceil$. В процессе применяется случайная величина ξ, равномерно распределенная на интервале $(0, 1)$. Эта величина генерируется линейным конгруэнтным способом. Для каждого шага алгоритма вычисляется новое значение ξ.

Алгоритм 14: Нахождение простых чисел с использованием теоремы Диемитко (PrimeNumbers)

Input: t — требуемая размерность простого числа,
q — простое число
Output: p

1 $R = \left\lceil \frac{2^{t-1}}{q} \right\rceil + \left\lceil \frac{2^{t-1}\xi}{q} \right\rceil$
2 **if** $R \not\equiv 0 \pmod 2$ **then**
3 $\quad R = R + 1$
4 $u = 0$
5 $n = (R + u)\, q + 1$
6 **if** $n > 2t$ **then**
7 $\quad$ Возврат на шаг 1
8 **if** $2^{(n-1)} \equiv 1 \pmod n$ **and** $2^{(R+u)} \not\equiv 1 \pmod n$ **then**
9 $\quad p = n$
10 $\quad$ **break**
11 **else**
12 $\quad u = u + 2$
13 $\quad$ Возврат на шаг 5

Result: p

Некоторые простые числа, полученные этим методом, могут не быть определены как таковые, поскольку на шаге 8 проверка условия теоремы Диемитко

проводится только для числа $a = 2$, а не для всех $a < p$. Однако вероятность того, что случайно выбранное число a выполнит условия теоремы Диемитко для простого числа n, равна $(1-1/q)$. Использование проверки исключительно для $a = 2$ оказывается вполне достаточным, чтобы исключить из рассмотрения лишь небольшое количество простых чисел. Преимущество выбора $a = 2$ связано с тем, что возведение числа 2 в степень в двоичной системе представления выполняется очень эффективно.

Приведем Пример 3.2.1 нахождения простого числа с использованием теоремы Диемитко.

Пример 3.2.1. *Для* $q = 3 = 11_2$*, вычислим простое число длиной* $t = 4$

Найдем R *при* $\xi = 0.5$*:*

$$R = \left\lceil \frac{8}{3} \right\rceil + \left\lceil \frac{8 \cdot 0.5}{3} \right\rceil = 3.$$

Для соблюдения четности $R = R + 1 = 4$.

Кандидат в простые числа $p = 4 \cdot 3 + 1 = 13$.

Так как, $2^{12} \mod 13 \equiv 1$ *и* $2^4 \mod 13 \not\equiv 1$.

Следовательно, искомое простое число $p = 13 = 1011_2$.

Таким образом, используя Алгоритм 14, разработан Алгоритм 15, позволяющий построить компактные базисы с модулями вида $p_i = Rq + 1$.

Алгоритм 15: Построение компактных базисов

Input: $\{q_1, q_2, \ldots, q_n\}$, t

Output: $b = \{p_1, p_2, \ldots, p_k\}$

```
1 p = PrimeNumbers(q_1, t)
2 p добавить в b
3 for i = 2, i ⩽ n, i++ do
4 |   p =PrimeNumbers(q_1, t)
5 |   if p < 2p_1 then
6 |   |   p добавить в b
```

Result: b

В литературе часто описываются методы построения базисов СОК на основе чисел Мерсенна [44]. В работе [55] предложен метод фильтрации для

построения очень больших базисов СОК. Однако эти подходы не учитывают условие компактности.

Для оценки эффективности предложенного алгоритма было проведено два этапа исследований. На первом из них проводилось сравнение скорости построения компактных базисов, вычисленных по теореме Диемитко, с методами построения базисов большой размерности в СОК. Для сравнения были выбраны два подхода: метод на основе чисел Мерсенна и метод фильтрации общего вида. Результаты сравнения производительности алгоритмов представлены в таблице 23 приложения В. Полученные результаты также представлены на рисунке 3.3.

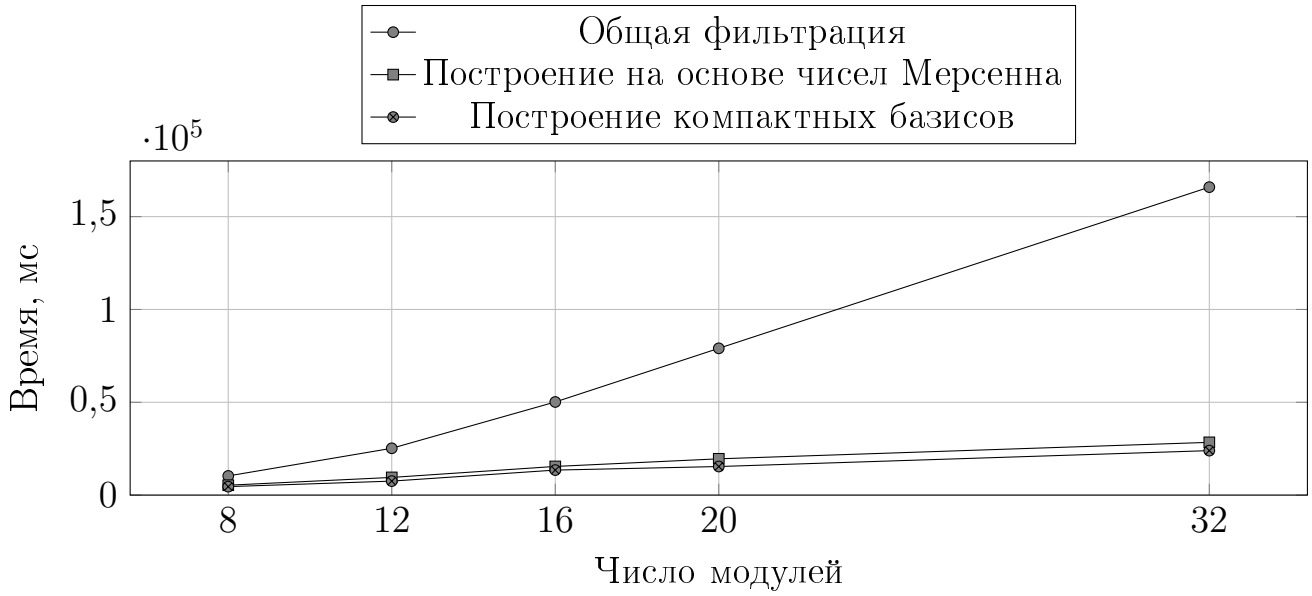

Рисунок 3.3 — Сравнение времени построения базисов СОК разными методами

Результаты исследования демонстрируют, что алгоритм построения компактных базисов в среднем на 17% быстрее метода на основе чисел Мерсенна и на 73% быстрее метода общей фильтрации. Наибольшее преимущество в производительности наблюдается при построении базиса из 32 модулей, где алгоритм построения компактных базисов превосходит метод на основе чисел Мерсенна на 15,7%, а метод общей фильтрации – на 85,6%.

На втором этапе исследования выбраны компактные базисы с размером динамического диапазона от 32 до 128 бит и сравнены с наборами модулей специального вида $2^n - 1, 2^n, 2^n + 1$ в выполнении модульных операций. Наборы для сравнения представлены в таблицах 24 и 25 приложения В. Результаты времени выполнения модульных операций представлены в таблицах 27, 27 и 28 приложения В. На основании данных таблиц построены графики 3.4–3.6.

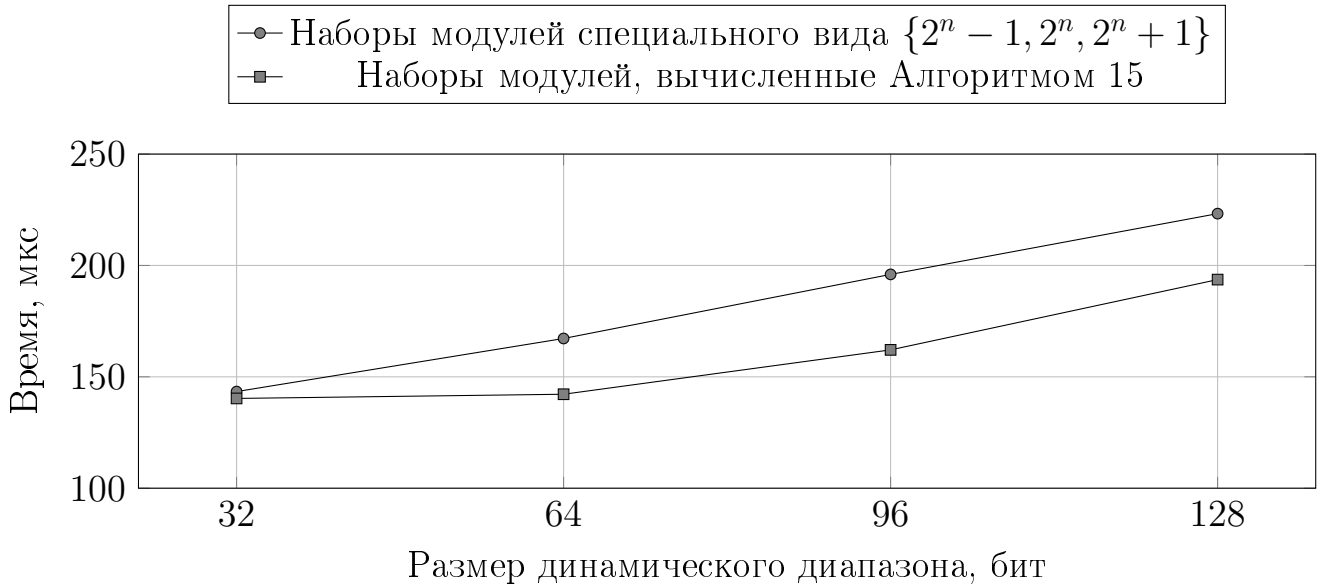

Рисунок 3.4 — Сравнения времени выполнения операции сложения для полученных компактных базисов и базисов специального вида

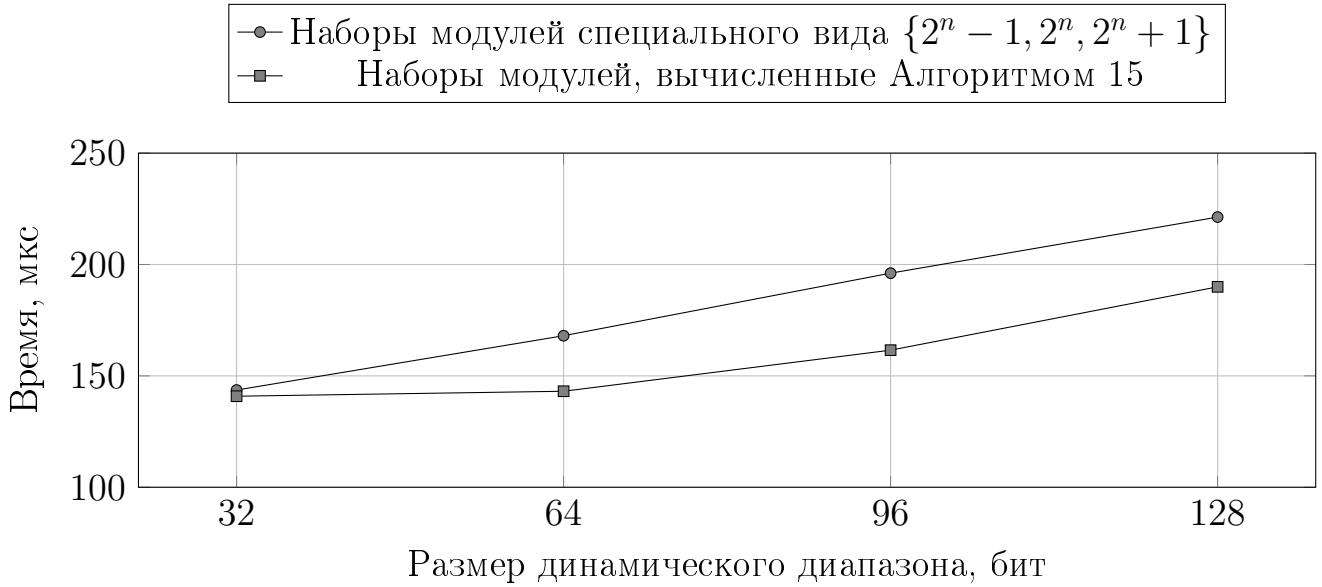

Рисунок 3.5 — Сравнение времени выполнения операции вычитания для полученных компактных базисов и базисов специального вида

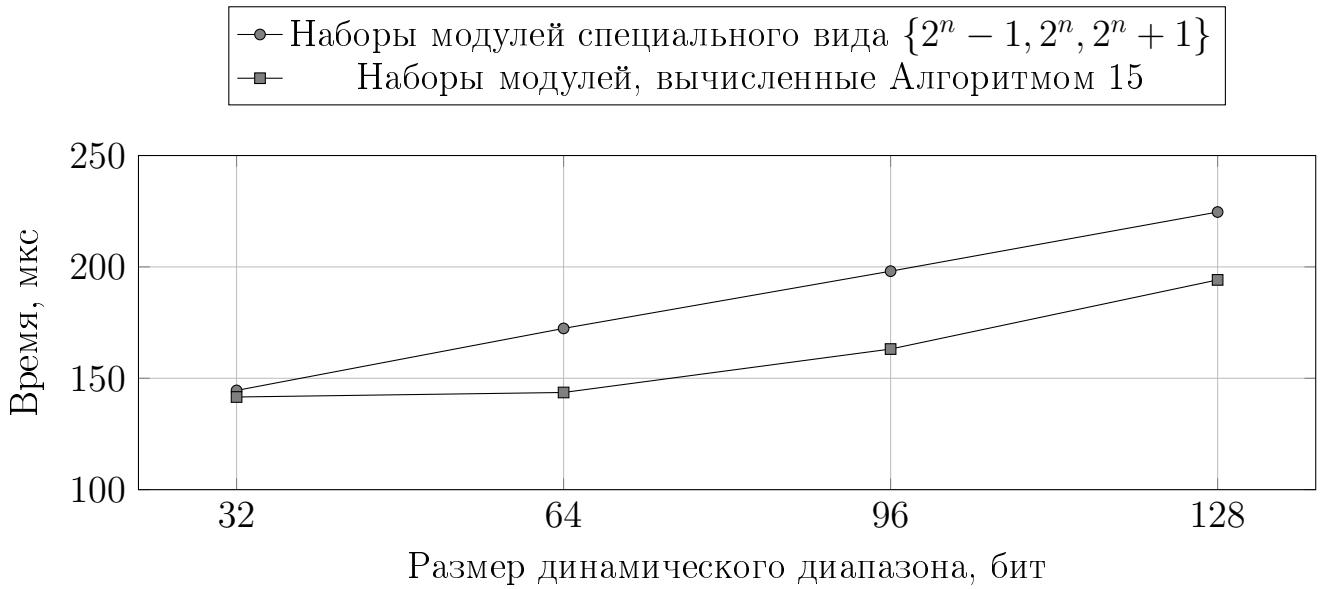

Рисунок 3.6 — Сравнение времени выполнения операции умножения для полученных компактных базисов и базисов специального вида

На основании представленных данных можно сделать вывод о том, что в среднем компактные базисы обеспечивают следующие выигрыши в скорости при выполнения модульных операций: на 11.87% при сложении, на 12.08% при вычитании и 12.43% для умножения. Таким образом, использование компактных базисов позволяет ускорить вычисления на 12% в среднем по всем операциям по сравнению с использованием модулей специального вида. Особенно заметный прирост скорости наблюдается для разрядности 96 и 128 бит, что говорит о перспективе использования алгоритма построения компактных базисов при увеличении разрядности чисел.

3.2.2 Поиск оптимальных весов для функции ядра Акушского

Оптимальными весами для функции ядра Акушского являются веса, при которых $C_P = 2^N$. Для поиска оптимальных весов можно использовать метод, предложенный в параграфе 2.6, однако при большом количестве модулей в базисе СОК, поиск оптимальных весов становиться вычислительно сложной задачей. Мы предлагаем использование метода Монте-Карло или генетического алгоритма для поиска оптимальных весов.

Метод Монте-Карло — это группа численных методов, основанных на случайном моделировании (стохастическом подходе), которые применяются для решения различных математических, статистических и других задач [78]. Метод был разработан в середине двадцатого века, главным образом в контексте исследований, связанных с ядерной физикой, но быстро нашел применение во множестве других дисциплин. Метод основан на использовании случайных или псевдослучайных чисел. Они применяются для имитации поведения сложных систем или случайных процессов. Благодаря своей случайности, метод позволяет быстро находить решения в больших диапазонах. В работе [117] были описаны основные преимущества данного метода и дано разъяснение его использования.

Данный метод может быть интегрирован для поиска оптимальных весов функции ядра Акушского, в системе остаточных классов. Ниже представлен Алгоритм 16, который реализует данный метод.

Алгоритм 16: Метод Монте-Карло для поиска оптимальных весов функции ядра

Input: $\{p_1, p_2, \ldots, p_n\}$,
$max\_iterations$,
$weigh\_limit$
Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}, i = 1, 2, \ldots, n$
Output: $\{w_1, w_2, \ldots, w_i\}, N$ — если найдены оптимальные веса

```
for i = 1, i ⩽ max_iterations, i++ do
    w_i = random(0, weight_limit)
    C_P = Σ_i^n w_i · P_i
    if C_P > 0 and (C_P ∧ (C_P − 1)) = 0 then
        N = log_2 C_P
        return w_i, N
```

Метод Монте-Карло эффективен для небольших базисов, однако с увеличением количества модулей вероятность нахождения оптимальных весов падает, из-за этого время выполнения алгоритма может быть значительным. С целью ускорения поиска оптимальных весов предложено использовать генетический алгоритм.

Генетический алгоритм — это метод оптимизации и поиска, основанный на принципах естественного отбора и эволюции в биологии [6]. Генетические алгоритмы часто рассматриваются как оптимизаторы функций, хотя спектр задач, к которым применяются генетические алгоритмы, довольно широк. Метод является разновидностью эволюционных вычислений, таких как наследование, мутации, отбор и кроссинговер. Из работы [80] можно выделить следующую структуру:

- Ген — это структура данных, представляющая одно возможное решение задачи. Множество генов — популяция.
- Функция приспособленности оценивает качество генов. Она определяет, насколько решение соответствует заданным критериям.
- Селекция (selection) — выбор генов (родителей) для создания следующего поколения. Обычно выбираются гены с высоким значением функции приспособленности.
- Скрещивание (crossover) — объединение генетической информации двух родителей для создания одного или нескольких потомков.
- Мутация (mutation) — внесение случайных изменений в потомков для поддержания разнообразия и предотвращения застревания в локальных максимумах.

Такой алгоритм можно применить для поиска оптимальных весов. Для начала нужно будет задать начальную популяцию, путем генерации случайных весов. Оценивать приспособленность будем через степень двойки функции C_P. Чем меньше значение функции приспособленности, тем лучше данное решение.

Далее выбираются лучшие веса из текущей популяции, происходит этап селекции. Отбирается верхняя половина популяции — веса с наименьшими значениями функции приспособленности. Эти лучшие решения используются для создания потомков.

Создаются новые «потомки» путем рекомбинации генов (весов) двух родителей, выбирается точка скрещивания. Для поддержания генетического разнообразия потомки подвергаются мутации с определенной вероятностью и затем, новая популяция формируется из лучших индивидов текущего поколения и потомков, тем самым получая оптимальные веса. Ниже представлен псевдокод данного алгоритма.

Для оценки эффективности предложенных алгоритмов проведен замер времени поиска оптимальных весов для наборов модулей общего вида. На-

Алгоритм 17: Генетический алгоритм для поиска оптимальных весов функции ядра

Input: $\{p_1, p_2, \ldots, p_n\}$,
$population\_size$,
$weight\_limit$
Data: $P = \prod_{i=1}^{n} p_i$,
$P_i = \frac{P}{p_i}, i = 1, 2, \ldots, n$
Output: $\{w_1, w_2, \ldots, w_i\}, N$ — если найдены оптимальные веса

for $i = 1$, $i \leqslant population\_size$, $i++$ **do**
    $w_i = random(0, weight\_limit)$

// Оценка приспособленности:

$C_P = \sum_i^n w_i \cdot P_i$
if $C_P > 0$ **and** $(C_P \wedge (C_P - 1)) = 0$ **then**
    $N = \log_2 C_P$
    return w_i, N

// Селекция:

$n = population\_size$
$f_{w_i} = \frac{C(P)}{\sum\limits_{i=1}^{n} C(P)}$, где f_{w_i} — вероятность выбора w_i веса
$crossover\_point = random(0, len(p_1, p_2, \ldots, p_n))$
$new\_population = new\_population + w_i$

// Мутация:

for $i = 1$, $i \leqslant new\_population$, $i++$ **do**
    Замена случайного веса w_i в диапазоне $weight\_limit$
    $population = population + new\_population$

Возврат на шаг 3

боры модулей используемые для моделирования представлены в таблице 8. Оптимальные веса для используемых в моделировании наборов модулей представлены в таблице 9.

Таблица 8 — Наборы модулей для поиска оптимальных весов функции ядра

Набор модулей
$\{23, 29, 31\}$
$\{23, 29, 31,37\}$
$\{23, 29, 31, 37,41\}$
$\{23, 29, 31, 37, 41, 43\}$
$\{23, 29, 31, 37, 41, 43, 47\}$
$\{23, 29, 31, 37, 41, 43, 47, 53\}$

Таблица 9 — Веса, найденные Алгоритмом 16 и 17, для наборов модулей из таблицы 8

Оптимальные веса	
Метод Монте-Карло	Генетический алгоритм
$\{4, 17, 1\}$	$\{4, 17, 1\}$
$\{7, 40, 11,26\}$	$\{8, 3, 40, 37\}$
$\{7, 20, 27, 46, 48\}$	$\{14, 49, 22, 38, 10\}$
$\{2, 25, 10, 11, 30, 38\}$	$\{1, 6, 27, 33, 27, 22\}$
$\{13, 18, 40, 26, 33, 6, 10\}$	$\{18, 6, 1, 12, 6, 19,11\}$
$\{15, 10, 36, 38, 23, 2, 43, 28\}$	$\{18, 29, 3, 48, 23, 18, 41, 11\}$

В таблице 10 представлено время работы алгоритмов.

Для наглядности на рисунке 3.7 отображены результаты времени выполнения алгоритмов из таблицы 10.

С увеличением размера набора модулей наблюдается значительное увеличение времени выполнения для обоих методов. Начиная с набора $\{23, 29, 31, 37\}$ и далее, генетический алгоритм демонстрирует значительно более высокую производительность по сравнению с методом Монте-Карло. Например, для набора из восьми модулей $\{23, 29, 31, 37, 41, 43, 47, 53\}$ генетический алгоритм работает в 5.8 раз быстрее.

Таблица 10 — Время нахождения оптимальных весов, с

Набор модулей	Метод Монте-Карло	Генетический алгоритм
$\{23, 29, 31\}$	0.014	0.006
$\{23, 29, 31, 37\}$	2	12
$\{23, 29, 31, 37, 41\}$	47	24
$\{23, 29, 31, 37, 41, 43\}$	125	31
$\{23, 29, 31, 37, 41, 43, 47\}$	189	42
$\{23, 29, 31, 37, 41, 43, 47, 53\}$	278	48

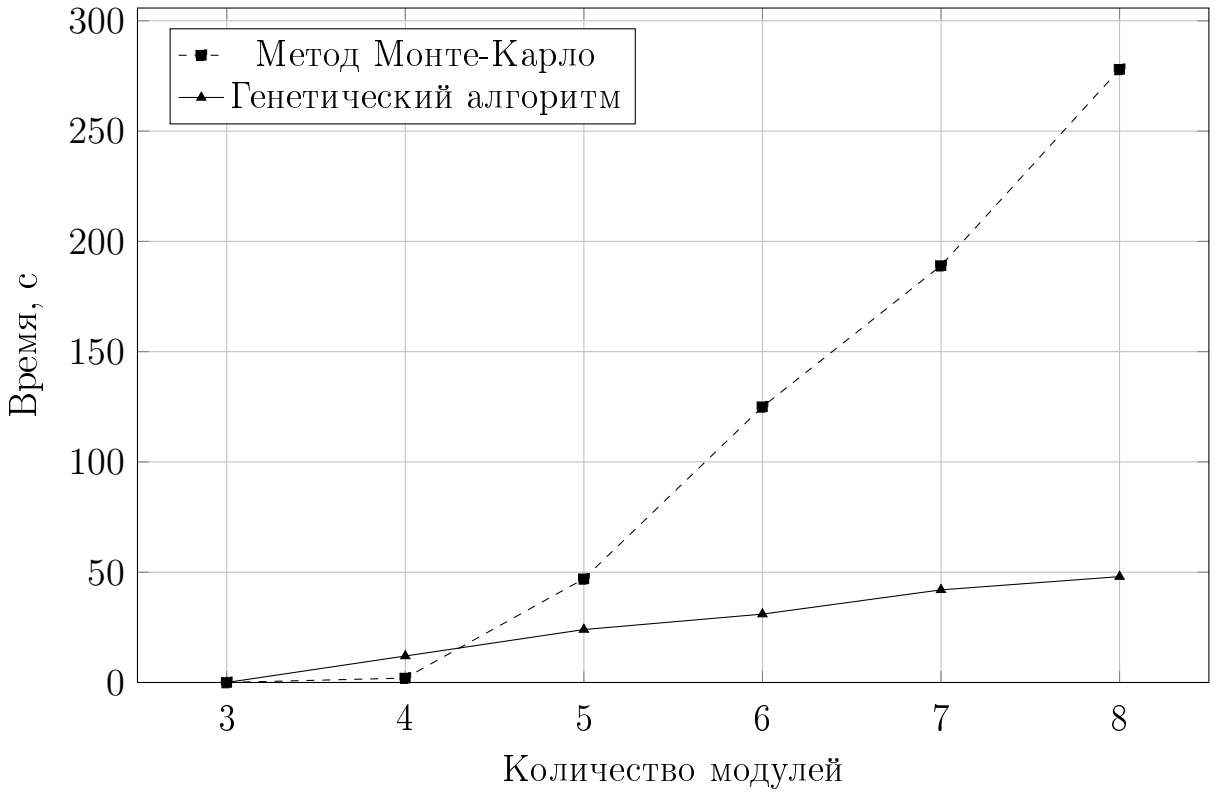

Рисунок 3.7 — График зависимости времени выполнения от количества модулей для методов поиска оптимальных весов

В среднем, генетический алгоритм быстрее метода Монте-Карло на 71.2%. Для задач с большим числом модулей целесообразнее использовать генетический алгоритм, который показывает лучшее время выполнения, несмотря на усложнение задачи.

3.3 Модуль арифметических операций в СОК

Вычислительный модуль арифметических операций в системе остаточных классов представляет собой специализированную структуру, ориентированную на выполнение операций сложения, вычитания и умножения чисел, представленных в системе остатков. Основное преимущество такой архитектуры заключается в возможности параллельной обработки остатков по взаимно простым модулям, что обеспечивает ускорение вычислений за счет независимости вычислительных потоков. При создании вычислительного модуля были использованы технологии параллельного программирования OpenMP — для реализации параллелизма на уровне разрядов чисел, обеспечиваемого выполнением вычислений в системе остаточных классов.

Для сравнения скорости операций модулярного сложения и умножения в качестве реализации в двоичной системе счисления были выбраны библиотеки NTL и MIRACLE. NTL — высокопроизводительная портативная C++ библиотека для решения задач теории чисел. Включает структуры данных и алгоритмы обработки целых чисел любой длины, векторов, матриц и полиномов над целыми числами и над конечными полями, а также арифметику с плавающей точкой произвольной точности [87]. MIRACL — это библиотека для работы с большими числами и выполнения криптографических операций. Она поддерживает арифметику произвольной точности и часто используется в криптографических приложениях, таких как RSA, ECC [79].

Для моделирования и сравнения был взят диапазон от 128 до 1024 бит. В системе остаточных классов наборы модулей выбирались так, чтобы для динамического диапазона P выполнялось условие $\lfloor \log_2 P \rfloor = k$ бит. Наборы получены с использованием Алгоритма 15. Данные о наборах модулей представлены в таблице 11.

Таблица 11 — Параметры наборов модулей системы остаточных классов, использованных для моделирования модуля арифметических операций

Размер набора модулей, n	Длина динамического диапазона, бит
8	128
12	256
16	512
20	768
32	1024

Измерения проводились на числах фиксированной длины, что исключает влияние случайной вариативности выполнения отдельных операций на итоговые результаты. Битовая длина числа, по модулю которого вычисляется сумма или произведение, равна $\left\lfloor \log_2 \frac{P}{2} \right\rfloor$. Во всех экспериментах тестовые данные создавались с использованием алгоритма «Вихрь Мерсенна», который применялся для генерации исходных данных. Эти данные затем передавались в СОК. Диапазон случайных чисел для входных данных ограничивался в зависимости от типа операции, чтобы обеспечить соответствие динамическому диапазону СОК и избежать переполнения. Для операции сложения диапазон значений был сокращен до половины полного диапазона и ограничивался интервалом $\left[0, \frac{P}{2}\right)$. Для операции умножения диапазон был уменьшен до квадратного корня от исходного диапазона и ограничивался интервалом $\left[0, \sqrt{P}\right)$. Обработка данных выполнялась последовательно с использованием библиотек NTL и MIRACL, при этом фиксировалось время выполнения операций каждым из методов. Все доступные потоки центрального процессора были задействованы в процессе вычислений.

Для каждого динамического диапазона и для каждой проверяемой операции было выполнено по 10^5 измерений. В таблице 12 представлены данные о времени выполнения операции модулярного сложения, а в таблице 13 – результаты модулярного умножения. На рисунках 3.8 и 3.9 показаны графики, иллюстрирующие зависимость времени выполнения операций от размерности чисел.

Анализ операции модулярного сложения показывает, что уже на малой разрядности 128 бит наблюдается преимущество СОК в 17.4% относительно

Таблица 12 — Результаты моделирования операции модулярного сложения, мкс

Длина динамического диапазона, бит	СОК	NTL	MIRACL
128	1904	2306	2453
256	2374	3431	4895
512	2449	4227	5963
768	2810	5109	7007
1024	3113	5910	8438

Таблица 13 — Результаты моделирования операции модулярного умножения, мкс

Длина динамического диапазона, бит	СОК	NTL	MIRACL
128	1967	2411	2629
256	2384	3646	5130
512	2571	4520	6678
768	2892	5311	7535
1024	3254	6337	9762

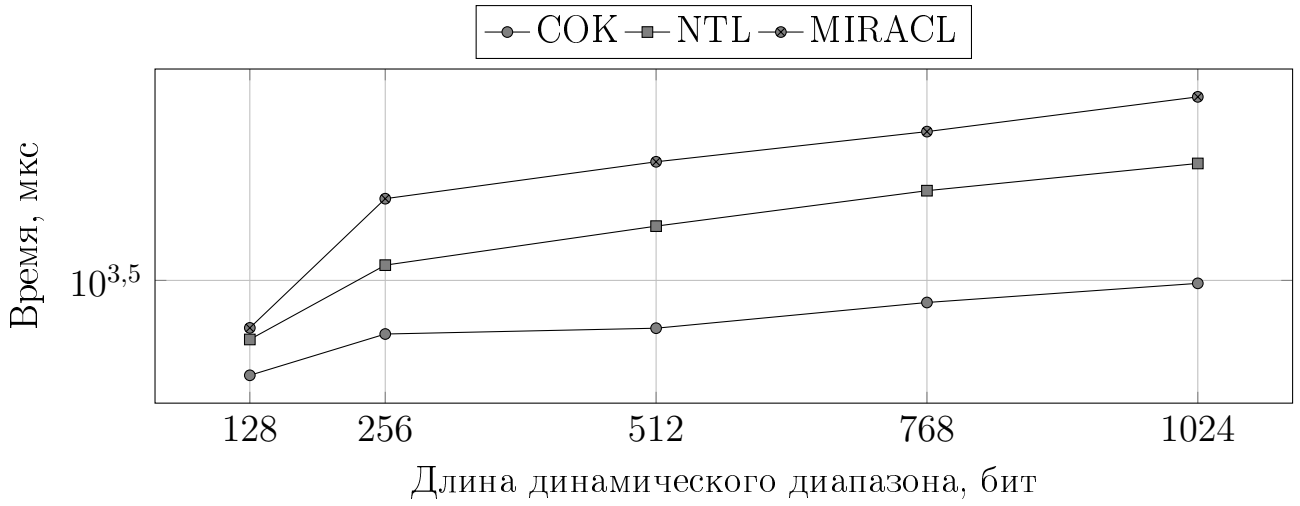

Рисунок 3.8 — График времени выполнения операции модулярного сложения

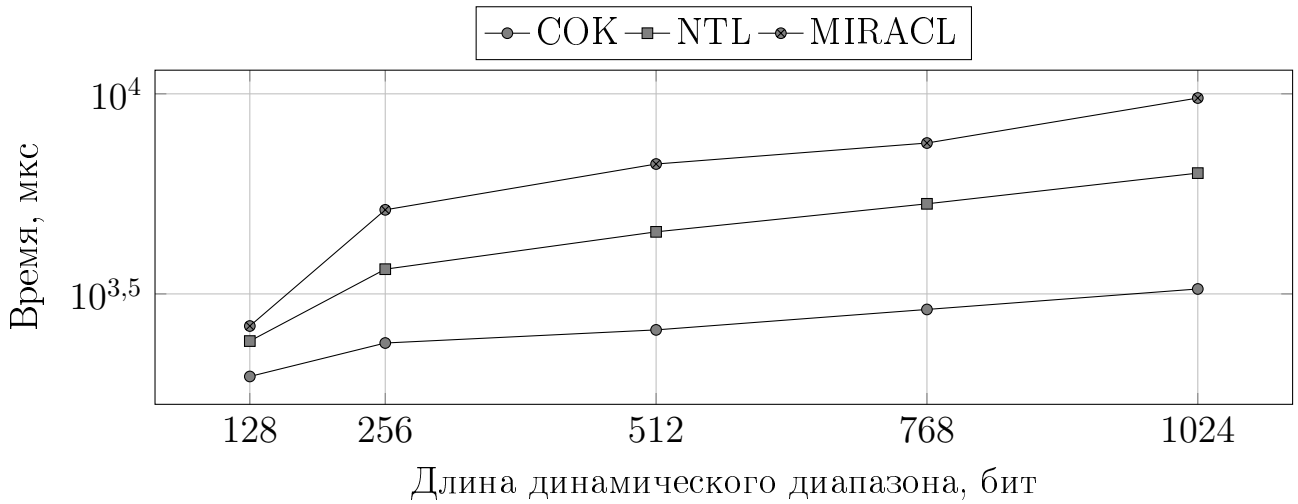

Рисунок 3.9 — График времени выполнения операции модулярного умножения

NTL и 22.4% относительно MIRACL. С увеличением длины динамического диапазона эффективность СОК существенно возрастает, достигая на 1024 битах преимущества в 47.3% над NTL и 63,1% над MIRACL. В среднем использование СОК позволяет сократить время выполнения операции модулярного сложения на 36.5% по сравнению с NTL и на 51.2% по сравнению с MIRACL.

Наиболее эффективная операция — это модулярное умножение, при размерности в 1024 бит модулярное умножение в СОК на 48.65% быстрее NTL и на 66.67% быстрее MIRACL. В среднем СОК на 38.1% быстрее NTL и 53.7% быстрее MIRACL для операции модулярного умножения чисел.

Модуль обеспечивает низкую задержку вычислений, высокую масштабируемость и возможность интеграции в более сложные вычислительные системы, включая системы цифровой обработки сигналов и криптографические алгоритмы.

3.4 Модуль обратного преобразования чисел из системы остаточных классов

В параграфе 2.3.1 было предложено использование КТО и ранга функции ядра Акушского, для обратного преобразования из СОК в ПСС. Согласно предложенному методу, формулу (2.17) можно записать в виде

$$X = \sum_{i=1}^{n} P_i \cdot x_i \cdot \left|P_i^{-1}\right|_{p_i} - \check{r}\left(X\right) \cdot P,$$

Для подтверждения статистической значимости исследования всех немодульных операций СОК для каждого набора модулей проводится 10^6 циклов, в каждом из которых выполняется по 10^4 реализаций. Результатом является среднее время выполнения.

Для методов обратного преобразования было проведено два этапа моделирования.

На первом изучается производительность семи наборов модулей специального вида $\{2^n - 1, 2^n, 2^n + 1\}$ с размерностью динамического диапазона от 16 до 64 бит. Данные базисы представлены в таблице 14.

Таблица 14 — Наборы модулей, используемые для первого этапа моделирования обратного преобразования

Размер динамического диапазона, бит	Набор модулей
16	$\{63, 64, 65\}$
24	$\{255, 256, 257\}$
32	$\{2047, 2048, 2049\}$
40	$\{16383, 16384, 16385\}$
48	$\{65535, 65536, 65537\}$
56	$\{524287, 524288, 524289\}$
64	$\{4194403, 4194304, 4194305\}$

На втором этапе проводится анализ семи наборов моделей общего вида, варьируя от 3 до 9 модулей, где каждый модуль имеет 8-битную длину. Данные наборы представлены в таблице 15.

Результаты первого этапа моделирования представлены в таблице 29 приложения Г, на основании данных результатов построен рисунок 3.10. Результаты второго этапа моделирования представлены в таблице 30 приложения Г, данные результаты отображены на рисунке 3.11.

Алгоритм на основе КТО и ранга функции ядра имеет лучшие результаты в диапазоне от 16 до 64 бит. Он на 10.5% в среднем быстрее КТО и на 20.4% быстрее приближенной КТО. В случае динамического изменения числа восьми битных модулей алгоритм на основе КТО и ранга функции ядра Акушского также показывает лучшие результаты. Он быстрее КТО на 7.2% и быстрее приближенной КТО на 17.1%.

Таблица 15 — Наборы модулей, используемые для второго этапа моделирования обратного преобразования

Количество модулей	Набор модулей
3	$\{257, 263, 271\}$
4	$\{257, 263, 271, 277\}$
5	$\{257, 263, 271, 277, 281\}$
6	$\{257, 263, 269, 271, 277, 281, 283\}$
7	$\{257, 263, 269, 271, 277, 281, 283, 293\}$
8	$\{257, 263, 269, 271, 277, 281, 283, 293, 307\}$
9	$\{257, 263, 269, 271, 277, 281, 283, 293, 307, 311\}$

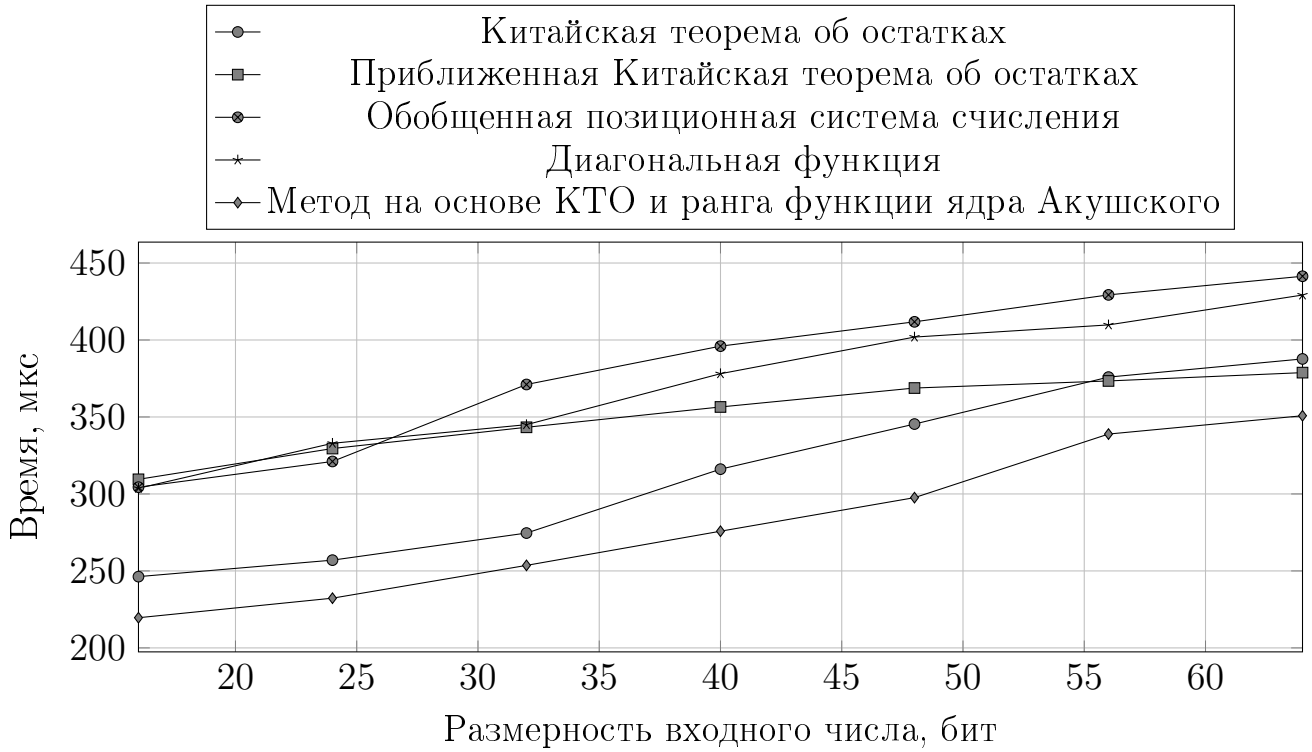

Рисунок 3.10 — Сравнение времени для методов обратного преобразования из СОК в ПСС, первый этап

3.5 Модуль деления чисел в системе остаточных классов

Деление считается одной из самых сложных арифметических операций. Даже в вычислительных системах, использующих позиционную систему счисления, эта операция выделяется, так как ее выполнение занимает значительно

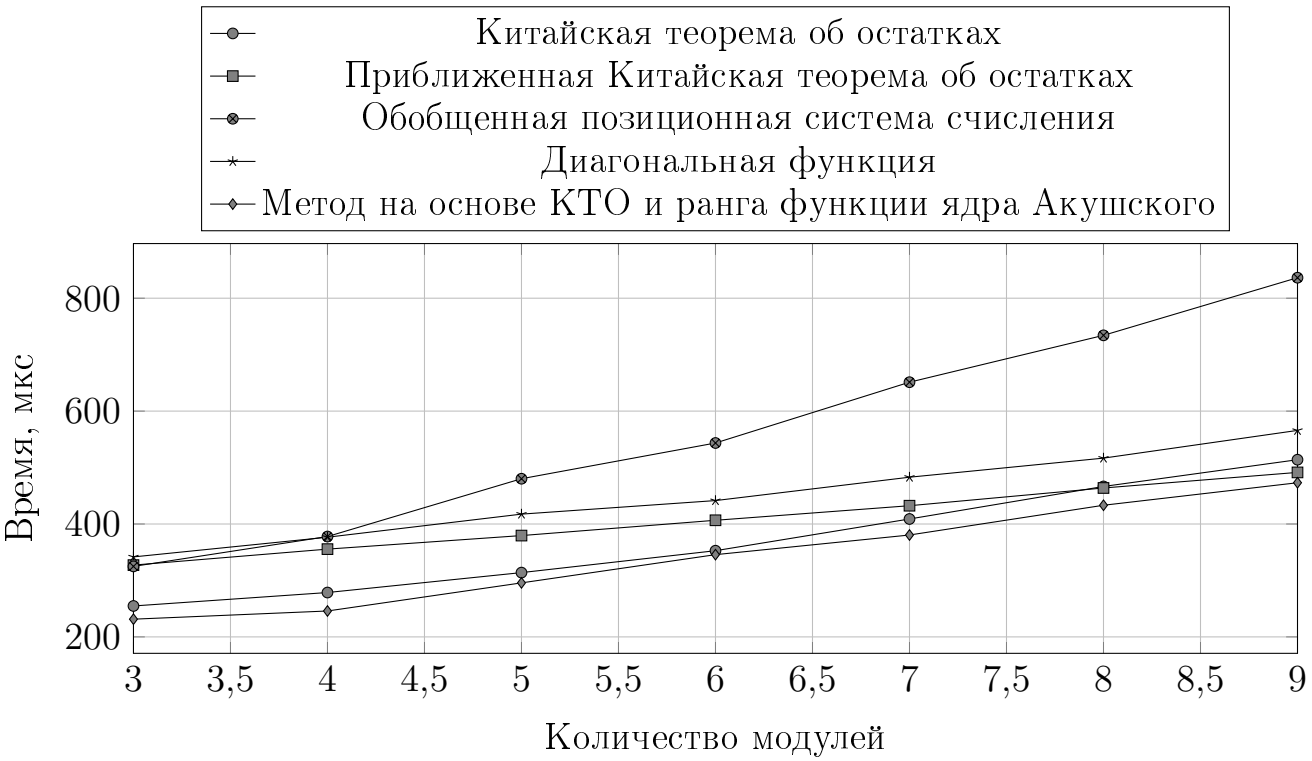

Рисунок 3.11 — Сравнение времени для методов обратного преобразования из СОК в ПСС, второй этап

больше времени по сравнению с большинством простых операций — примерно на порядок дольше. В СОК сложность деления возрастает еще больше, поскольку оно, в общем случае, не является модульной операцией. Это означает, что цифра частного для отдельного модуля не может быть определена исключительно на основе цифр делимого и делителя в этом модуле, а требует дополнительной информации о значениях этих чисел в целом [2].

В Примере 2.4.1 из параграфа 2.4 было показано, что на каждом шаге итерационного деления требуется вычислять функцию ядра для промежуточных частных. Так в процессе делимое и делитель делятся на два, функцию ядра для текущего частного можно вычислять на основе предыдущего частного с использованием Теорем 2.4.1 и 2.4.2.

Для подтверждения свойств предложенных алгоритмов оптимизации деления мы реализовали их на языке C++ и сравнили производительность с классическим итерационным делением. Для моделирования использовались модули из таблицы 15. При проведении моделирования были получены временные характеристики каждого метода. Результаты отражены в таблице 16 и на рисунке 3.12.

Таблица 16 — Результаты моделирования модуля деления чисел в СОК, мкс

Размер набора модулей, n	Итерационное деление	Итерационное деление с использованием Теоремы 2.4.1	Итерационное деление с использованием Теоремы 2.4.2
3	345	334.3	320.1
4	446.2	426.8	430.9
5	532.2	499.3	472.2
6	654.7	603.1	566.5
7	774.5	726.1	651.8
8	884	821.3	733.4
9	1079.2	968.9	905.7

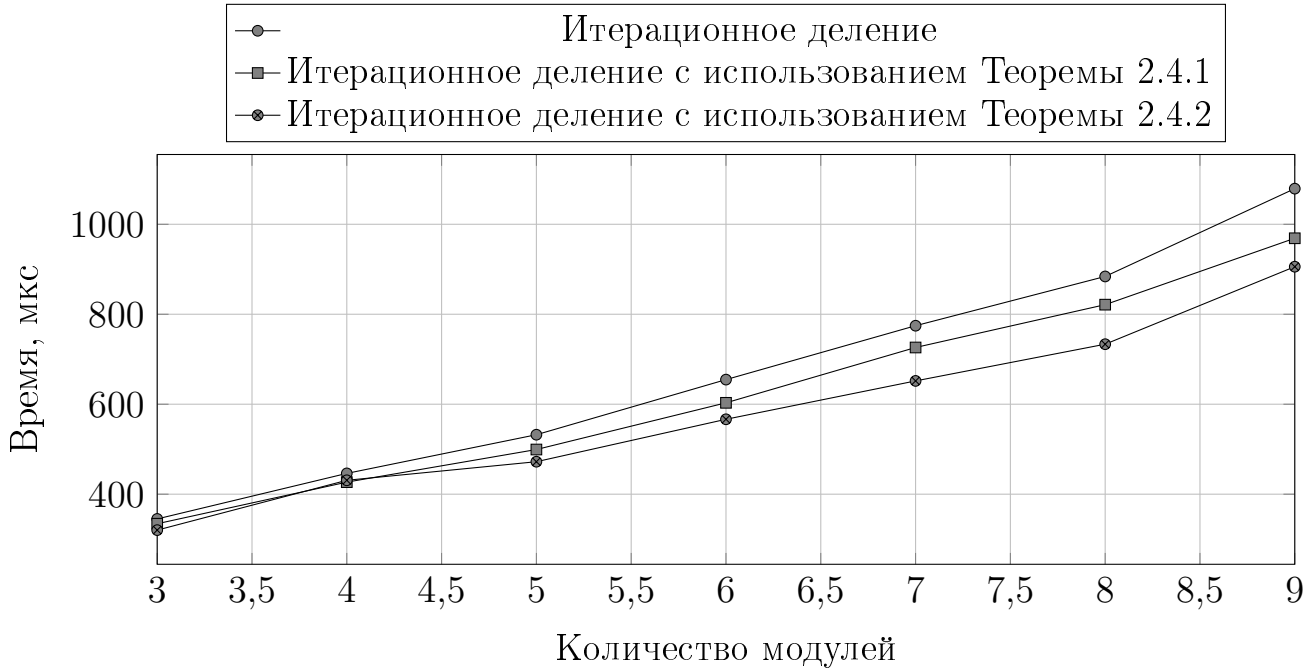

Рисунок 3.12 — Сравнение времени для алгоритмов деления в СОК

Предложенные подходы позволяют увеличить скорость вычисления функции ядра Акушского, а значит, увеличить эффективность выполнения итерационного деления в СОК.

Использование Теорем 2.4.1 и 2.4.2 позволяет значительно сократить время выполнения итерационного деления в СОК. Особенно это становится важным при увеличении количества модулей. Метод, основанный на Теореме 2.4.2, демонстрирует наилучшие результаты. Алгоритм с использованием Теоремы 2.4.2 в среднем быстрее обычного итерационного деления на 12.2% и быстрее итерационного деления с Теоремой 2.4.1 на 6.2%.

Однако стоит еще раз отметить, что использование теоремы 2.4.2 возможно только при использовании нечетных модулей в базисе СОК, что в большинстве случаев делает непригодным использование специальных наборов модулей (см. таблицу 6). В случае необходимости применения специальных наборов модулей необходимо использовать теорему 2.4.1.

3.6 Модуль определения знака числа в системе остаточных классов

Одним из ключевых аспектов множества вычислительных методов является способность работать с отрицательными числами. Однако система остаточных классов в своем классическом виде не поддерживает представление отрицательных значений, так как оперирует числами в пределах класса вычетов по модулю P. Это ограничивает ее применение в ряде задач. И.Я. Акушский детально изучил различные подходы к включению отрицательных чисел в систему остаточных классов [2]. В своих исследованиях он предложил концепцию искусственных форм чисел, которая оказалась одним из наиболее эффективных методов. На основе этого подхода были разработаны правила выполнения арифметических операций, обеспечивающие корректные результаты как по значению, так и по знаку. Однако сам знак числа остается неявным, поскольку из-за непозиционной структуры кода СОК невозможно однозначно определить его расположение на числовой оси.

В параграфах 2.5.1 и 2.5.2 приложены алгоритмы определения знака числа с использованием минимальной функции ядра удовлетворяющей условиям $C(X) \leqslant C_P < P$ и $C_P = 2^N$. Использование минимальной функции ядра

для определения знака позволяет снизить размер операндов, а также вычислительно сложную операцию взятия по модулю можно заменить взятием N младших бит числа.

Проведено моделирование методов определения знака числа в СОК на основе Китайской теоремы об остатках, ее приближенной реализации, а также функции Пирло рассмотренных в параграфе 2.5 с алгоритмами использующими минимальную функцию ядра. Результаты моделирования представлены в таблицах 31–35 приложения Д. Лучшие результаты времени для каждой размерности представлены в таблице 36. На основе полученных данных построим графики сравнения рассмотренных методов.

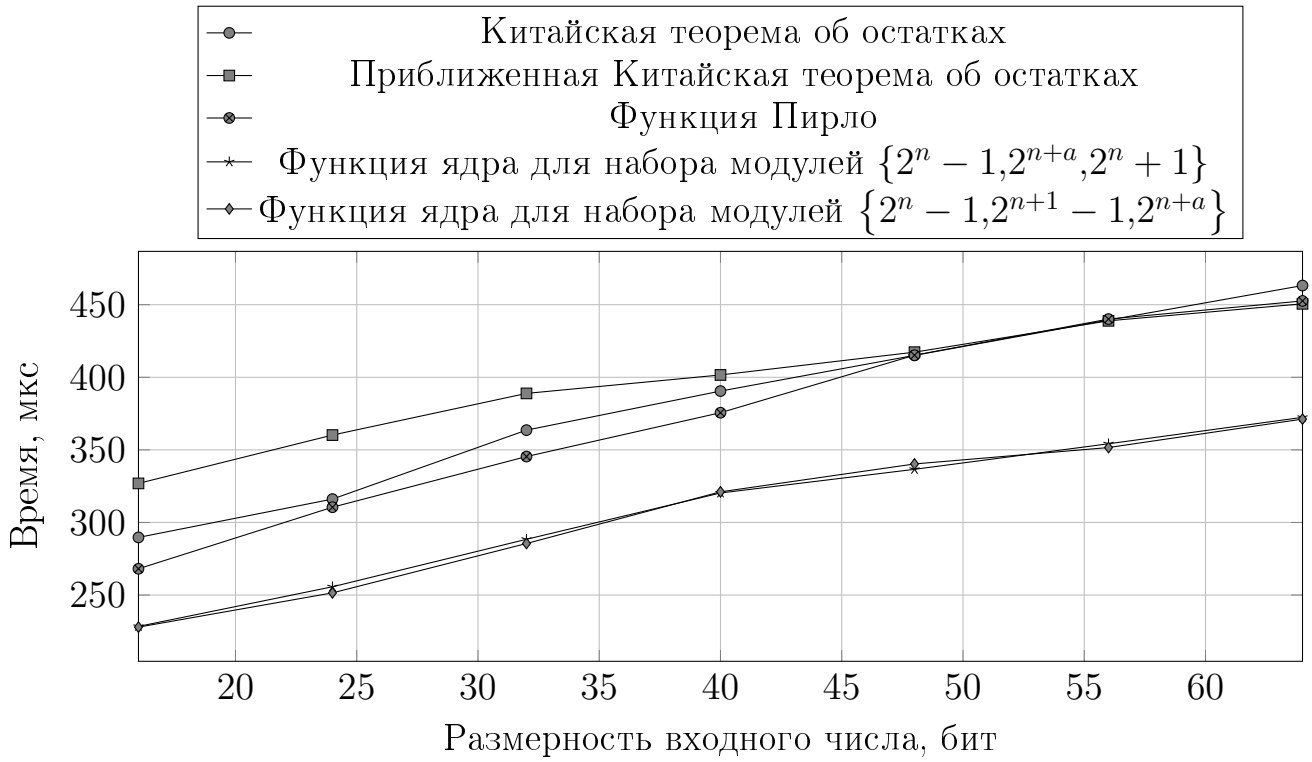

Рисунок 3.13 — Сравнение времени для методов определения знака числа в СОК

Из графика 3.13 можно сделать вывод, что алгоритм на основе использования минимальной функции ядра для набора модулей $\{2^n-1, 2^{n+1}-1, 2^{n+a}\}$ в среднем на 20% быстрее КТО, на 23.4% быстрее приближенной КТО, на 17.6% быстрее функции Пирло и на 0.5% быстрее алгоритма для набора модулей $\{2^n-1, 2^{n+a}, 2^n+1\}$.

3.7 Модуль сравнения чисел в системе остаточных классов

Реализация алгоритма сравнения чисел в системе остаточных классов включает два основных этапа. На первом этапе происходит вычисление позиционной характеристики для модулярных чисел $X = (x_1, x_2, \ldots, x_n)$ и $Y = (y_1, y_2, \ldots, y_n)$. Эта характеристика, обозначаемая как ПХ(X) и ПХ(Y), отражает уникальные свойства чисел в рамках выбранной системы модулей. На втором этапе осуществляется непосредственное сравнение характеристик ПХ(X) и ПХ(Y) в позиционной системе счисления, что позволяет сравнить числа СОК.

В большинстве методов задача сравнения чисел решается посредством перевода числа из СОК в ПСС и их последующего сравнения. К примеру, с использованием КТО [89] или ОПСС [64]. Высокая вычислительная сложность преобразования чисел из системы остаточных классов в позиционную систему счисления стала причиной активного изучения методов аппроксимации их позиционного представления. В работе [97] применяется приближенная КТО, а в работе диагональная функция [46].

В параграфе 2.6 предложен метод построения функций ядра для операции сравнения чисел, которые, как и в случае определения знака позволяют снизить размер операндов, а также заменить операцию деления с остатком на операцию взятия старших бит числа.

Для моделирования методов сравнения в качестве функции ядра была выбрана функция для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ (см. таблицу 5 параграфа 2.6). Предложенный метод был сравнен с методами сравнения чисел в СОК на основе Китайской теоремы об остатках, ее приближенной реализации, обобщенной позиционной системы счисления, а также диагональной функции. Для сравнения данных методов из таблиц 37–41 приложения Е выбраны лучшие значения по времени для каждой размерности. На основе полученных данных построим график сравнения рассмотренных методов.

Из графика можно сделать выводы о том, что метод на основе приближенной КТО, метод на основе диагональной функции и метод использования функции ядра у которой $C_P = 2^N$ имеют лучшие характеристики по времени работы по сравнению с КТО и ОПСС. В среднем разработанная функция ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$ быстрее других методов на 15.1%.

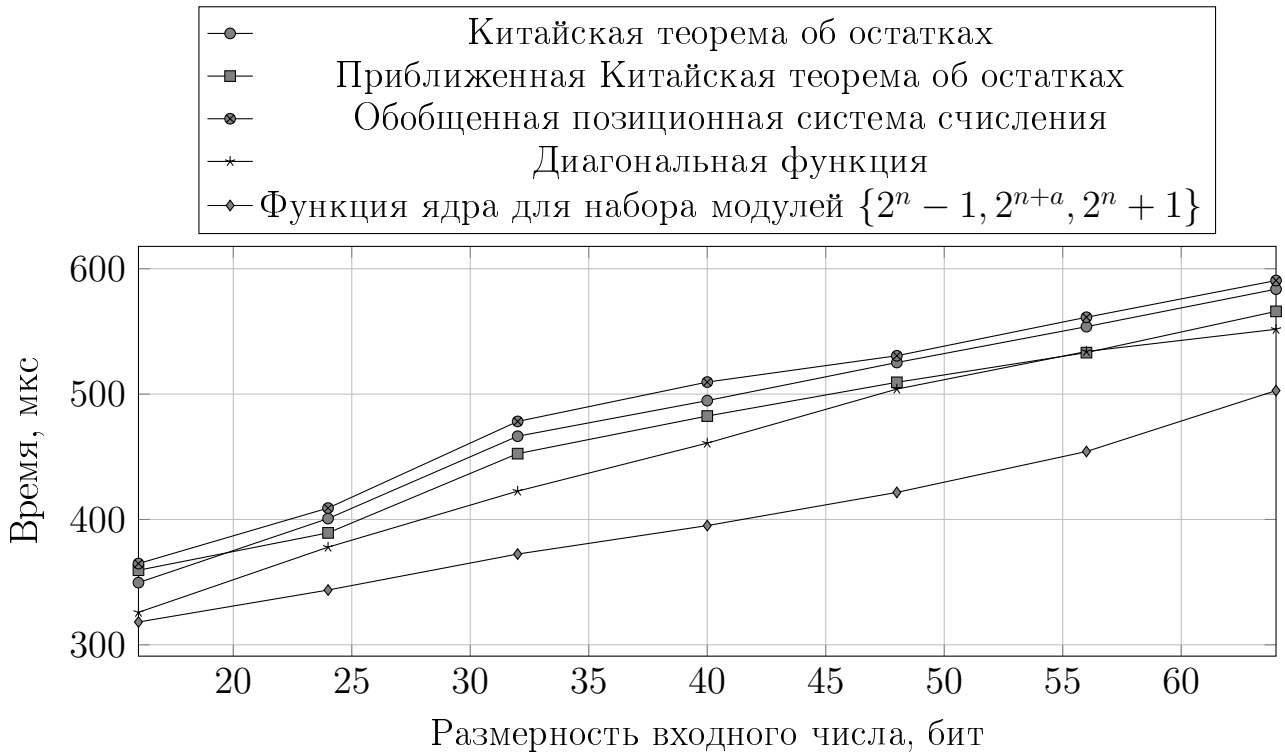

Рисунок 3.14 — Сравнение времени для методов сравнения чисел в СОК

Таким образом, разработанный метод построения функций ядра для операции сравнения чисел в СОК имеет лучшие результаты времени по сравнению с известными методами.

3.8 Выводы по третьей главе

В параграфе 3.1 предложена архитектура программного комплекса для проектирования высокоскоростных вычислительных систем на основе модулярной арифметики. Архитектура программного комплекса включает модули выбора параметров СОК, модулярных вычислений и определения позиционных характеристик для выполнения немодульных операций СОК. В рамках данного комплекса разработано и зарегистрировано 8 программ для ЭВМ [11—18].

Основное преимущество системы остаточных классов заключается в возможности параллельной обработки остатков по взаимно простым модулям, что обеспечивает ускорение вычислений за счет независимости вычислительных потоков. Проведенное в параграфе 3.2.1 исследование показало, что наиболее эффективными базисами среди наборов модулей специального вида являются

базы $\{2^n-1, 2^n, 2^n+1\}$ и $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$. Однако для работы с многоразрядными числами целесообразно использовать наборы модулей с произвольным количеством модулей. За счет большего количества модулей, можно достичь большей степени параллелизма. Однако для эффективной реализации наборов модулей общего вида необходимо выполнении условия компактности. В параграфе 3.2.1 рассмотрен алгоритм построения компактных базисов с использованием теоремы Диемитко. Предложенный алгоритм позволяет снизить время выбора набора модулей СОК в среднем на 17% по сравнению с методом на основе чисел Мерсенна и на 73% по сравнению с методом общей фильтрации. Также алгоритм обеспечивает ускорение при выполнении модульных операций в среднем на 12% по сравнению с использованием модулей специального вида $\{2^n-1, 2^n, 2^n+1\}$.

Для вычисления позиционной характеристики числа в системе остаточных классов была использована функция ядра Акушского. Оптимальными весами для функции ядра Акушского являются веса, при которых $C_P = 2^N$. С целью поиска оптимальных весов в параграфе 3.2.2 предложено использование метода Монте-Карло и генетического алгоритма. В среднем, генетический алгоритм быстрее метода Монте-Карло на 71.2%. Для приложений СОК, где необходимо использовать большое число модулей, целесообразнее использовать генетический алгоритм.

С использованием алгоритма построения компактных базисов в параграфе 3.3 проведено моделирование модулярного сложения и умножения чисел двоичной системы счисления и системы остаточных классов. Модулярное сложение и умножение в двоичной системе реализовано с использованием библиотек NTL и MIRACL для возможности проведения операций над многоразрядными числами. Моделирование проводилось на языке программирования высокого уровня C++. Реализация в СОК позволяет сократить время модулярного сложения на 36.5% по сравнению с NTL, и на 51.2% по сравнению с MIRACL. Модулярное умножение в СОК на 38.1% быстрее NTL и 53.7% быстрее MIRACL.

В параграфе 3.4 проведено моделирование методов обратного преобразования из системы остаточных классов в позиционную систему счисления. Предложенный алгоритм на основе КТО и ранга числа функции ядра в среднем на 10.5% быстрее КТО при изменении длины динамического диапазона, но

фиксированном количестве модулей и на 7.2% быстрее КТО при динамическом изменении количества модулей.

В параграфе 3.5 промоделировано итерационное деление в СОК. Алгоритм итерационного деления с использованием Теоремы 2.4.2 в среднем быстрее обычного итерационного деления на 12.2% и быстрее итерационного деления с Теоремой 2.4.1 на 6.2%. Однако использование Теоремы 2.4.2 возможно только при использовании нечетных модулей в базисе СОК.

В параграфе 3.6 рассмотрены алгоритмы определения знака числа в СОК. Предложены алгоритмы определения знака использующие минимальные функции ядра для наборов модулей $\{2^n-1, 2^{n+a}, 2^n+1\}$ и $\left\{2^n-1, 2^{n+1}-1, 2^{n+a}\right\}$. Алгоритм на основе использования минимальной функции ядра для набора модулей $\left\{2^n-1, 2^{n+1}-1, 2^{n+a}\right\}$ в среднем на 20% быстрее КТО, на 23.4% быстрее приближенной КТО и на 17.6% быстрее функции Пирло.

В параграфе 3.7 рассмотрены методы сравнения чисел в СОК. На основе метода построения функций ядра для сравнения чисел из параграфа 2.6 проведено моделирование сравнения чисел в СОК с использованием функции ядра для набора модулей $\{2^n-1, 2^{n+a}, 2^n+1\}$. Предложенный метод на основе использования минимальных функций ядра, у которых $C_P = 2^N$, в среднем быстрее классических методов сравнения чисел в СОК на 15.1%.

Таким образом, была разработана архитектура программного комплекса для выполнения модульных и немодульных операций в СОК. Представленные модули могут быть использованы при реализации криптографических алгоритмов в системах туманных вычислений, а также в модулях вычислительных систем обработки данных, работающих в системе остаточных классов.

Заключение

Основные результаты исследования могут быть сформулированы следующим образом:

1. Разработан метод для перевода из СОК в ПСС на основе КТО и ранга числа функции ядра Акушского, который за счет ухода от операции нахождения остатка от деления позволяет сократить время вычислений по сравнению с Китайской теоремой об остатках и с приближенным методом на основе Китайской теоремой об остатках.
2. Разработаны модифицированные методы итерационного деления в СОК. Данные методы позволяют повысить скорость вычислений по сравнению с классическим итерационным делением за счет вычисления функции ядра на основе ее значения для предыдущего частного.
3. Разработаны алгоритмы определения знака на основе минимальной функции ядра Акушского с заданными свойствами для специальных наборов модулей. Эти алгоритмы позволяют за счет уменьшения размера операндов и снижения вычислительной сложности операции нахождения остатка от деления с $O(n^2)$ до $O(n)$ повысить скорость определения знака по сравнению с классическими методами.
4. Разработан метод построения функций ядра Акушского для операции сравнения чисел. Сравнение чисел в СОК на основе построенных минимальных функций ядра Акушского, у которых $C(P) = 2^N$, позволяет повысить скорость вычислений за счет снижения размера операндов, а также замены операции нахождения остатка от деления взятием N младших бит числа.
5. Разработан алгоритм построения компактных базисов системы остаточных классов. Наборы базисов, полученные данным алгоритмом, позволяют снизить время выполнения модульных операций в СОК по сравнению с использованием модулей специального вида за счет большей степени параллелизма и соответствия критерию компактности.
6. Разработаны алгоритмы поиска оптимальных весов функции ядра Акушского. Использование генетического алгоритма для решения задачи поиска оптимальных весов функции ядра позволило сократить время поиска на 71.2% по сравнению с методом Монте-Карло. Для

задач с большим числом модулей в системе остаточных классов целесообразнее использовать генетический алгоритм, который демонстрирует лучшее время выполнения, несмотря на возрастающую сложность задачи.

7. Разработан программный комплекс для выполнения модульных и немодульных операций в СОК на вычислительных узлах туманных сред. Комплекс показал высокую эффективность при выполнении указанных операций. Представленные вычислительные модули могут быть применены в облачных и туманных системах, работающих в условиях ограниченных вычислительных ресурсов. К направлениям внедрения результатов диссертационного исследования можно отнести криптографию и искусственные нейронные сети. Кроме того, разработанные методы, алгоритмы и программные средства могут быть применены в других областях, где требуется параллельная обработка больших объемов данных.

Список литературы

1. *Акушский, И. Я.* О новой позиционной характеристике непозиционного кода и ее приложении [Текст] / И. Я. Акушский, В. М. Бурцев, И. Т. Пак // Теория кодирования и оптимизация сложных систем. — Алма-Ата, Наука, КазССР, 1977. — С. 8—16.
2. *Акушский, И. Я.* Машинная арифметика в остаточных классах [Текст] / И. Я. Акушский, Д. И. Юдицкий. — М. : Сов. радио, 1968.
3. *Бабенко, М. Г.* Математические модели, методы и алгоритмы обработки зашифрованных данных в распределенных средах [Текст] : Дис. д-ра физ.-мат. наук / Бабенко М. Г. — М. : Институт системного программирования им. В.П. Иванникова Российской академии наук, 2022.
4. Исследование специальных наборов модулей системы остаточных классов [Текст] / В. В. Луценко [и др.] // Труды Института системного программирования РАН. — 2025. — Т. 37, № 3. — С. 107—120.
5. *Королева, М. Н.* Метод Диемитко формирования больших простых чисел [Текст] / М. Н. Королева, В. А. Липницкий. — 2015.
6. *Курейчик, В. М.* Генетические алгоритмы [Текст] / В. М. Курейчик // Известия Южного федерального университета. Технические науки. — 1998. — Т. 8, № 2. — С. 4—7.
7. *Луценко, В. В.* Исследование нейросетевых методов обнаружения и исправления ошибок в системе остаточных классов [Текст] / В. В. Луценко, М. Г. Бабенко // Всероссийская научно-практическая конференция им. Жореса Алфёрова: сборник тезисов статей. — 2023. — С. 248.
8. Оптимизация алгоритма деления чисел в системе остаточных классов на основе функции ядра Акушского [Текст] / В. В. Луценко [и др.] // Труды Института системного программирования РАН. — 2023. — Т. 35, № 5. — С. 157—168.
9. *Панкратова, И. А.* Теоретико-числовые методы в криптографии: учебное пособие [Текст] / И. А. Панкратова. — Томск: ТГУ, 2009.

10. Поиск оптимальных весов для функции ядра Акушского [Текст] / В. В. Луценко [и др.] // Вестник Южно-Уральского государственного университета. Серия: Вычислительная математика и информатика. — 2025. — Т. 14, № 2. — С. 26—41.

11. *Свидетельство о государственной регистрации программы для ЭВМ 2023619967 Российская Федерация.* Реализация вычисления знака числа в системе остаточных классов / Кучеров Н.Н., Ширяев Е.М., Безуглова Е.С., Луценко В.В., Гробова С.К.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2023617877; заявл. 26.04.2023; опубл. 17.05.2023. — 1 с. [Текст].

12. *Свидетельство о государственной регистрации программы для ЭВМ 2023662227 Российская Федерация.* Реализация сравнения чисел в системе остаточных классов / Кучеров Н.Н., Ширяев Е.М., Безуглова Е.С., Луценко В.В., Колбин М.Д.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2023617795; заявл. 26.04.2023; опубл. 07.06.2023. — 1 с. [Текст].

13. *Свидетельство о государственной регистрации программы для ЭВМ 2024619754 Российская Федерация.* Программный комплекс для определения критических ядер при построении функции ядра Акушского / Луценко В.В., Бабенко М.Г.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619005; заявл. 25.04.2024; опубл. 25.04.2024. — 1 с. [Текст].

14. *Свидетельство о государственной регистрации программы для ЭВМ 2024619970 Российская Федерация.* Программа для определения четности числа в системе остаточных классов с использованием функции ядра Акушского / Луценко В.В., Бабенко М.Г., Герюгова А.Э.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619046; заявл. 25.04.2024; опубл. 02.05.2024. — 1 с. [Текст].

15. *Свидетельство о государственной регистрации программы для ЭВМ 2024660104 Российская Федерация.* Программа для определения переполнения при сложении чисел в системе остаточных классов с использованием функции ядра Акушского / Луценко В.В., Бабенко М.Г.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619026; заявл. 25.04.2024; опубл. 02.05.2024. — 1 с. [Текст].

16. *Свидетельство о государственной регистрации программы для ЭВМ 2024660122 Российская Федерация.* Программа для подбора оптимальных весов для функции ядра Акушского с использованием генетического алгоритма / Луценко В.В., Бабенко М.Г., Безуглова Е.С., Ширяев Е.М; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619092; заявл. 25.04.2024; опубл. 02.05.2024. — 1 с. [Текст].

17. *Свидетельство о государственной регистрации программы для ЭВМ 2024660861 Российская Федерация.* Программа для реализации итерационного деления чисел в системе остаточных классов с использованием функции ядра Акушского / Луценко В.В., Бабенко М.Г., Безуглова Е.С., Ширяев Е.М; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619025; заявл. 25.04.2024; опубл. 14.05.2024. — 1 с. [Текст].

18. *Свидетельство о государственной регистрации программы для ЭВМ 2024661039 Российская Федерация.* Программа для реализации масштабирования чисел в системе остаточных классов с использованием функции ядра Акушского / Луценко В.В., Бабенко М.Г.; заявитель и правообладатель Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет». — № 2024619051; заявл. 25.04.2024; опубл. 15.05.2024. — 1 с. [Текст].

19. Сравнительный анализ методов восстановления чисел, заданных на основе модульного представления [Текст] / Е. В. Непретимова [и др.] //

Естественные науки-основа настоящего и фундамент для будущего. — 2019. — С. 70—73.

20. Элементарное введение в эллиптическую криптографию [Текст] / А. А. Болотов [и др.] // Протоколы криптографии на эллиптических кривых. Изд. — 2006. — Т. 2.

21. A Compact FPGA-Based Accelerator for Curve-Based Cryptography in Wireless Sensor Networks [Текст] / M. Morales-Sandoval [et al.] // Journal of Sensors. — 2021. — Vol. 2021, no. 1. — P. 8860413.

22. A comparison of the key size and security level of the ecc and rsa algorithms with a focus on cloud/fog computing [Текст] / D. Patel [et al.] // International Conference on Information and Communication Technology for Intelligent Systems. — Springer. 2023. — P. 43—53.

23. A novel ASIC implementation of RSA algorithm [Текст] / Z. Keija [et al.] // 2003 5th International Conference on ASIC Proceedings. Vol. 2. — IEEE. 2003. — P. 1300—1303.

24. *Abdallah, M.* A systematic approach for selecting practical moduli sets for residue number systems [Текст] / M. Abdallah, A. Skavantzos // Proceedings of the Twenty-Seventh Southeastern Symposium on System Theory. — IEEE. 1995. — P. 445—449.

25. *Abood, O.* A survey on cryptography algorithms [Текст] / O. Abood, S. Guirguis // International Journal of Scientific and Research Publications. — 2018. — Vol. 8, no. 7. — P. 495—516.

26. Acceleration of RSA processes based on hybrid ARM-FPGA cluster [Текст] / X. Bai [et al.] // 2017 IEEE Symposium on Computers and Communications (ISCC). — IEEE. 2017. — P. 682—688.

27. *Ahmed, S.* A survey on security and privacy challenges in smarthome based IoT [Текст] / S. Ahmed, S. Zeebaree // International Journal of Contemporary Architecture. — 2021. — Vol. 8, no. 2. — P. 489—510.

28. Algorithm for Determining the Optimal Weights for the Akushsky Core Function with an Approximate Rank [Текст] / E. Shiriaev [et al.] // Applied Sciences. — 2023. — Vol. 13, no. 18. — P. 10495.

29. An analysis and evaluation of lightweight hash functions for blockchain-based IoT devices [Текст] / S. Abed [et al.] // Cluster computing. — 2021. — Vol. 24. — P. 3065—3084.

30. An ECC-based lightweight remote user authentication and key management scheme for IoT communication in context of fog computing [Текст] / U. Chatterjee [et al.] // Computing. — 2022. — Vol. 104, no. 6. — P. 1359—1395.

31. An efficient architecture for designing reverse converters based on a general three-moduli set [Текст] / A. S. Molahosseini [et al.] // Journal of Systems Architecture. — 2008. — Vol. 54, no. 10. — P. 929—934.

32. An efficient VLSI design for a residue to binary converter for general balance moduli $\{2^n - 1, 2^n + 1, 2^n - 3, 2^n + 3\}$ [Текст] / M.-H. Sheu [et al.] // IEEE Transactions on Circuits and Systems II: Express Briefs. — 2004. — Vol. 51, no. 3. — P. 152—155.

33. AR-RRNS: Configurable reliable distributed data storage systems for Internet of Things to ensure security [Текст] / N. Chervyakov [et al.] // Future Generation Computer Systems. — 2019. — Vol. 92. — P. 1080—1092.

34. *Atlam, H. F.* Fog computing and the internet of things: A review [Текст] / H. F. Atlam, R. J. Walters, G. B. Wills // Big data and cognitive computing. — 2018. — Vol. 2, no. 2. — P. 10.

35. *Bajard, J.-C.* An RNS Montgomery modular multiplication algorithm [Текст] / J.-C. Bajard, L.-S. Didier, P. Kornerup // IEEE Transactions on Computers. — 1998. — Vol. 47, no. 7. — P. 766—776.

36. *Bertino, E.* Data Security and Privacy in the IoT. [Текст] / E. Bertino // EDBT. Vol. 2016. — 2016. — P. 1—3.

37. *Bhardwaj, M.* Reverse converter for the 4-moduli superset $2^n - 1, 2^n, 2^n + 1, 2^{n+1}+1$ [Текст] / M. Bhardwaj, T. Srikanthan, C. T. Clarke // Proceedings of the IEEE Conference on Computer Arithmetic. Vol. 14. — Citeseer. 1999. — P. 168—175.

38. *Biswas, R.* A fast implementation of the RSA algorithm using the GNU MP library [Текст] / R. Biswas, S. Bandyopadhyay, A. Banerjee // IIIT-Calcutta, National workshop on cryptography. — 2003.

39. *Brickell, E. F.* A survey of hardware implementations of RSA [Текст] / E. F. Brickell // Advances in Cryptology—CRYPTO'89 Proceedings 9. — Springer. 1990. — P. 368—370.

40. *Cao, B.* An efficient reverse converter for the 4-moduli set $2^n - 1, 2^n, 2^n + 1, 2^{2n} - 1$ based on the new Chinese remainder theorem [Текст] / B. Cao, C. Chang, T. Srikanthan // IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications. — 2003. — Vol. 50, no. 10. — P. 1296—1303.

41. *Cao, B.* A residue-to-binary converter for a new five-moduli set [Текст] / B. Cao, C.-H. Chang, T. Srikanthan // IEEE Transactions on Circuits and Systems I: Regular Papers. — 2007. — Vol. 54, no. 5. — P. 1041—1049.

42. *Chen, L.* Wireless network security [Текст] / L. Chen, J. Ji, Z. Zhang. — Springer, 2013.

43. Comparison of ECC and RSA algorithm in resource constrained devices [Текст] / M. Bafandehkar [et al.] // 2013 international conference on IT convergence and security (ICITCS). — IEEE. 2013. — P. 1—3.

44. Cox-rower architecture for fast parallel montgomery multiplication [Текст] / S. Kawamura [et al.] // International Conference on the Theory and Applications of Cryptographic Techniques. — Springer. 2000. — P. 523—538.

45. Creating distributed artificial neural networks based on orthogonal transformations [Текст] / N. A. Vershkov [et al.] // Proceedings of the Institute for System Programming of the RAS. — 2024. — Vol. 36, no. 4. — P. 57—68.

46. *Dimauro, G.* A new technique for fast number comparison in the residue number system [Текст] / G. Dimauro, S. Impedovo, G. Pirlo // IEEE transactions on computers. — 1993. — Vol. 42, no. 5. — P. 608—612.

47. *Edamatsu, T.* Acceleration of large integer multiplication with intel avx-512 instructions [Текст] / T. Edamatsu, D. Takahashi // 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). — IEEE. 2018. — P. 211—218.

48. Efficient Acceleration of RSA Algorithm on GPU [Текст] / Z. Yao [et al.] // IEEE International Conference on Oxide Materials for Electronic Engineering (OMEE), Lviv, Ukraine. — 2012. — P. 531—534.

49. Efficient Reverse Converter Designs for the New 4-Moduli Sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ Based on New CRTs [Текст] / A. S. Molahosseini [et al.] // IEEE Transactions on Circuits and Systems I: Regular Papers. — 2009. — Vol. 57, no. 4. — P. 823—835.

50. Efficient RNS Reverse Converters for Moduli Sets with Dynamic Ranges Up to (10n+ 1)(10 n+ 1)-bit [Текст] / H. Pettenghi [et al.] // Circuits, Systems, and Signal Processing. — 2018. — Vol. 37. — P. 5178—5196.

51. *Fadhil, H. M.* Parallelizing RSA algorithm on multicore CPU and GPU [Текст] / H. M. Fadhil, M. I. Younis // International Journal of Computer Applications. — 2014. — Vol. 87, no. 6.

52. *Fedina, A.* Analytical Review of Classification and Clustering Methods of Cyber Attacks Based on Data Mining and Neural Network Approach [Текст] / A. Fedina, V. Lutsenko, N. Gladkova // Conference on Current Problems of Applied Mathematics and Computer Systems. — Springer. 2023. — P. 285—294.

53. Fog-based Self-Sovereign Identity with RSA in Securing IoMT Data. [Текст] / A. J. Basha [et al.] // Intelligent Automation & Soft Computing. — 2022. — Vol. 34, no. 3.

54. FPGA implementation of high-efficiency ECC point multiplication circuit [Текст] / X. Zhao [et al.] // Electronics. — 2021. — Vol. 10, no. 11. — P. 1252.

55. Generating very large RNS bases [Текст] / J. C. Bajard [et al.] // IEEE Transactions on Emerging Topics in Computing. — 2022. — Vol. 10, no. 3. — P. 1289—1301.

56. *Hariri, A.* A new high dynamic range moduli set with efficient reverse converter [Текст] / A. Hariri, K. Navi, R. Rastegar // Computers & mathematics with applications. — 2008. — Vol. 55, no. 4. — P. 660—668.

57. *Hassan, W.* Current research on Internet of Things (IoT) security: A survey [Текст] / W. Hassan, M. Noor // Computer networks. — 2019. — Vol. 148. — P. 283—294.

58. *Hawkins, S.* Awareness and challenges of Internet security [Текст] / S. Hawkins, D. Yen, D. Chou // Information Management & Computer Security. — 2000. — Vol. 8, no. 3. — P. 131—143.

59. *Hiasat, A. A.* Residue-to-binary arithmetic converter for the moduli set $(2^k, 2^k - 1, 2^{k-1} - 1)$ [Текст] / A. A. Hiasat, S. H. Abdel-Aty-Zohdy // IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. — 1998. — Vol. 45, no. 2. — P. 204—209.

60. *Hiasat, A.* VLSI implementation of new arithmetic residue to binary decoders [Текст] / A. Hiasat // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. — 2005. — Vol. 13, no. 1. — P. 153—158.

61. High-Speed Parity Number Detection Algorithm in RNS Based on Akushsky Core Function [Текст] / V. Lutsenko [et al.] // International Conference on Communication and Computational Technologies. — Springer. 2024. — P. 491—504.

62. *Hosseinzad, M.* A new moduli set for residue number system in ternary valued logic [Текст] / M. Hosseinzad, K. Navi // Journal of Applied Sciences. — 2007. — Vol. 7, no. 23. — P. 3729—3735.

63. Implementation of RSA algorithm based on RNS Montgomery multiplication [Текст] / H. Nozaki [et al.] // International Workshop on Cryptographic Hardware and Embedded Systems. — Springer. 2001. — P. 364—376.

64. *Isupov, K.* An Algorithm for Magnitude Comparison in RNS based on Mixed-Radix Conversion II [Текст] / K. Isupov // International Journal of Computer Applications. — 2016. — Vol. 141, no. 5.

65. *Lin, Y.-S.* Efficient parallel RSA decryption algorithm for manycore GPUs with CUDA [Текст] / Y.-S. Lin, C.-Y. Lin, D.-C. Lou // Proc. International Conference on Telecommunication Systems, Modeling and Analysis. — 2012. — P. 85—94.

66. *Liu, Y.* Moduli set selection and cost estimation for RNS-based FIR filter and filter bank design [Текст] / Y. Liu, E. M.-K. Lai // Design Automation for Embedded Systems. — 2004. — Vol. 9. — P. 123—139.

67. LoRaWAN protocol: specifications, security, and capabilities [Текст] / A. Yegin [et al.] // LPWAN Technologies for iot and m2m applications. — Elsevier, 2020. — P. 37—63.

68. *Lutsenko, V.* Comparative Analysis of Methods and Algorithms for Building a Digital Twin of a Smart City [Текст] / V. Lutsenko, M. Babenko // International Conference on Actual Problems of Applied Mathematics and Computer Science. — Springer. 2022. — P. 277—287.

69. *Lutsenko, V.* Construction of Akushsky Core Functions Without Critical Cores [Текст] / V. Lutsenko, M. Babenko, M. Deryabin // Mathematics. — 2024. — Vol. 12, no. 21. — P. 3399.

70. *Lutsenko, V.* An efficient implementation of the Montgomery algorithm using the Akushsky core function [Текст] / V. Lutsenko, E. Bezuglova // International Workshop on Advanced Information Security Management and Applications. — Springer. 2023. — P. 166—177.

71. *Lutsenko, V.* Fault Tolerant System for Data Storage, Transmission and Processing in Fog Computing Using Artificial Neural Networks [Текст] / V. Lutsenko, M. Zgonnikov // Conference on Current Problems of Applied Mathematics and Computer Systems. — Springer. 2023. — P. 199—212.

72. *Lutsenko, V.* Investigation of Neural Network Methods for Error Detection and Correction in the Residue Number System [Текст] / V. Lutsenko, M. Zgonnikov // International Workshop on Advanced Information Security Management and Applications. — Springer. 2024. — P. 194—206.

73. *Lutsenko, V. V.* High speed method of conversion numbers from residue number system to positional notation [Текст] / V. V. Lutsenko, M. G. Babenko, M. M. Khamidov // Proceedings of the Institute for System Programming of the RAS. — 2024. — Vol. 36, no. 4. — P. 117—132.

74. *Madakam, S.* Internet of Things (IoT): A literature review [Текст] / S. Madakam, R. Ramaswamy, S. Tripathi // Journal of computer and communications. — 2015. — Vol. 3, no. 5. — P. 164—173.

75. *Maleh, Y.* Towards an efficient datagram transport layer security for constrained applications in internet of things [Текст] / Y. Maleh, A. Ezzati // International Review on Computers and Software. — 2016. — Vol. 11, no. 7. — P. 611—621.

76. *Mathew, J.* Fast residue-to-binary converter architectures [Текст] / J. Mathew, D. Radhakrishnan, T. Srikanthan // 42nd Midwest Symposium on Circuits and Systems (Cat. No. 99CH36356). Vol. 2. — IEEE. 1999. — P. 1090—1093.

77. *McIvor, C.* Modified Montgomery modular multiplication and RSA exponentiation techniques [Текст] / C. McIvor, M. McLoone, J. McCanny // IEE Proceedings-Computers and Digital Techniques. — 2004. — Vol. 151, no. 6. — P. 402—408.

78. *Metropolis, N.* The monte carlo method [Текст] / N. Metropolis, S. Ulam // Journal of the American statistical association. — 1949. — Vol. 44, no. 247. — P. 335—341.

79. *MIRACL Library.* MIRACL Cryptographic SDK [Текст] / MIRACL Library. — URL: https : / / github . com / miracl / MIRACL (visited on 05/15/2025).

80. *Mirjalili, S.* Genetic algorithm [Текст] / S. Mirjalili, S. Mirjalili // Evolutionary algorithms and neural networks: theory and applications. — 2019. — P. 43—55.

81. Modified RSA-based algorithm: A double secure approach [Текст] / I. Al Barazanchi [et al.] // Telkomnika (Telecommunication Computing Electronics and Control). — 2019. — Vol. 17, no. 6. — P. 2818—2825.

82. *Mohan, P.* RNS to Binary Conversion [Текст] / P. Mohan, P. Ananda Mohan // Residue Number Systems: Theory and Applications. — 2016. — P. 81—132.

83. *Molahosseini, A. S.* A new five-moduli set for efficient hardware implementation of the reverse converter [Текст] / A. S. Molahosseini, C. Dadkhah, K. Navi // IEICE Electronics Express. — 2009. — Vol. 6, no. 14. — P. 1006—1012.

84. *Molahosseini, A. S.* A reverse converter for the enhanced moduli set $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n+1} - 1\}$ using CRT and MRC [Текст] / A. S. Molahosseini, K. Navi // 2010 IEEE Computer Society Annual Symposium on VLSI. — IEEE. 2010. — P. 456—457.

85. *Molahosseini, A. S.* A new residue to binary converter based on mixed-radix conversion [Текст] / A. S. Molahosseini, K. Navi, M. K. Rafsanjani // 2008 3rd International Conference on Information and Communication Technologies: From Theory to Applications. — IEEE. 2008. — P. 1—6.

86. *Navi, K.* How to teach residue number system to computer scientists and engineers [Текст] / K. Navi, A. S. Molahosseini, M. Esmaeildoust // IEEE Transactions on Education. — 2010. — Vol. 54, no. 1. — P. 156—163.

87. *NTL Library.* NTL: A Library for doing Number Theory [Текст] / NTL Library. — URL: https://libntl.org (visited on 05/15/2025).

88. *Obaid, O.* Security and privacy in IoT-based healthcare systems: a review [Текст] / O. Obaid, S.-B. Salman // Mesopotamian Journal of Computer Science. — 2022. — Vol. 2022. — P. 29—39.

89. *Omondi, A. R.* Residue number systems: theory and implementation [Текст]. Vol. 2 / A. R. Omondi, A. B. Premkumar. — World Scientific, 2007.

90. *Pagar, Y.* Load balancing of fog computing centre and its security using elliptic curve cryptography [Текст] / Y. Pagar, D. Rathod, G. Chowdhary // International Journal of Autonomic Computing. — 2020. — Vol. 3, no. 3/4. — P. 245—260.

91. *Patronik, P.* Design of Reverse Converters for General RNS Moduli Sets $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and $\{2^k, 2^n-1, 2^n+1, 2^{n-1}-1\}$ (n even) [Текст] / P. Patronik, S. J. Piestrak // IEEE Transactions on Circuits and Systems I: Regular Papers. — 2014. — Vol. 61, no. 6. — P. 1687—1700.

92. *Pettenghi, H.* RNS reverse converters for moduli sets with dynamic ranges up to $(8n+1)$-bit [Текст] / H. Pettenghi, R. Chaves, L. Sousa // IEEE Transactions on Circuits and Systems I: Regular Papers. — 2012. — Vol. 60, no. 6. — P. 1487—1500.

93. *Pourbigharaz, F.* A signed-digit architecture for residue to binary transformation [Текст] / F. Pourbigharaz // IEEE Transactions on Computers. — 1997. — Vol. 46, no. 10. — P. 1146—1150.

94. *Premkumar, A. B.* An RNS to binary converter in a three moduli set with common factors [Текст] / A. B. Premkumar // IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. — 1995. — Vol. 42, no. 4. — P. 298—301.

95. Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications [Текст] / C.-H. Chang [et al.] // IEEE circuits and systems magazine. — 2015. — Vol. 15, no. 4. — P. 26—44.

96. Residue-to-binary conversion by the quotient function [Текст] / G. Dimauro [et al.] // IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing. — 2003. — Vol. 50, no. 8. — P. 488—493.

97. Residue-to-binary conversion for general moduli sets based on approximate Chinese remainder theorem [Текст] / N. I. Chervyakov [et al.] // International journal of computer mathematics. — 2017. — Vol. 94, no. 9. — P. 1833—1849.

98. *Rivest, R. L.* A method for obtaining digital signatures and public-key cryptosystems [Текст] / R. L. Rivest, A. Shamir, L. Adleman // Communications of the ACM. — 1978. — Vol. 21, no. 2. — P. 120—126.

99. *Schinianakis, D.* Multifunction residue architectures for cryptography [Текст] / D. Schinianakis, T. Stouraitis // IEEE Transactions on Circuits and Systems I: Regular Papers. — 2014. — Vol. 61, no. 4. — P. 1156—1169.

100. *Schoinianakis, D.* Residue arithmetic systems in cryptography: a survey on modern security applications [Текст] / D. Schoinianakis // Journal of Cryptographic Engineering. — 2020. — Vol. 10, no. 3. — P. 249—267.

101. *Selent, D.* Advanced encryption standard [Текст] / D. Selent // Rivier Academic Journal. — 2010. — Vol. 6, no. 2. — P. 1—14.

102. *Shiriaev, E.* An Approximate Algorithm for Determining the Sign Function of a Number Using Neural Network Methods [Текст] / E. Shiriaev, V. Lutsenko, M. Babenko // International Workshop on Advanced Information Security Management and Applications. — Springer. 2023. — P. 247—255.

103. *Singh, A.* Cloud security issues and challenges: A survey [Текст] / A. Singh, K. Chatterjee // Journal of Network and Computer Applications. — 2017. — Vol. 79. — P. 88—115.

104. *Soderstrand, M.* An improved residue number system digital-to-analog converter [Текст] / M. Soderstrand, C. Vernia, J.-H. Chang // IEEE transactions on circuits and systems. — 1983. — Vol. 30, no. 12. — P. 903—907.

105. *Solapurkar, P.* Building secure healthcare services using OAuth 2.0 and JSON web token in IOT cloud scenario [Текст] / P. Solapurkar // 2016 2nd International Conference on Contemporary Computing and Informatics (IC3I). — IEEE. 2016. — P. 99—104.

106. *Suárez-Albela, M.* A practical evaluation on RSA and ECC-based cipher suites for IoT high-security energy-efficient fog and mist computing devices [Текст] / M. Suárez-Albela, P. Fraga-Lamas, T. Fernández-Caramés // Sensors. — 2018. — Vol. 18, no. 11. — P. 3868.

107. *Surendiran, R.* A fog computing approach for securing IoT devices data using DNA-ECC cryptography [Текст] / R. Surendiran, K. Raja // DS Journal of Digital Science and Technology. — 2022. — Vol. 1, no. 1. — P. 10—16.

108. *Szabo, N. S.* Residue arithmetic and its applications to computer technology [Текст] / N. S. Szabo, R. I. Tanaka. — New York: McGraw-Hill, 1967. — P. 236.

109. *Tahir, A.* Design and Implementation of RSA Algorithm using FPGA [Текст] / A. Tahir // International Journal of Computers & Technology. — 2015. — Vol. 14, no. 12. — P. 6361—6367.

110. *Taylor, F. J.* Residue arithmetic a tutorial with examples [Текст] / F. J. Taylor // Computer. — 1984. — Vol. 17, no. 05. — P. 50—62.

111. TensorCrypto: High throughput acceleration of lattice-based cryptography using tensor core on GPU [Текст] / W.-K. Lee [et al.] // IEEE Access. — 2022. — Vol. 10. — P. 20616—20632.

112. The SIMON and SPECK lightweight block ciphers [Текст] / R. Beaulieu [et al.] // Proceedings of the 52nd annual design automation conference. — 2015. — P. 1—6.

113. Towards understanding uncertainty in cloud computing with risks of confidentiality, integrity, and availability [Текст] / A. Tchernykh [et al.] // Journal of Computational Science. — 2019. — Vol. 36. — P. 100581.

114. *Turner, S.* Transport layer security [Текст] / S. Turner // IEEE Internet Computing. — 2014. — Vol. 18, no. 6. — P. 60—63.

115. *Van Vu, T.* Efficient implementations of the Chinese remainder theorem for sign detection and residue decoding [Текст] / T. Van Vu // IEEE Transactions on Computers. — 1985. — Vol. 100, no. 7. — P. 646—651.

116. *Vinod, A. P.* A Memoryless Reverse Converter for the 4-Moduli Superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ [Текст] / A. P. Vinod, A. B. Premkumar // Journal of Circuits, Systems, and Computers. — 2000. — Vol. 10, no. 1/2. — P. 85—99.

117. Why the Monte Carlo method is so important today [Текст] / D. Kroese [et al.] // Wiley Interdisciplinary Reviews: Computational Statistics. — 2014. — Vol. 6, no. 6. — P. 386—392.

118. *Yassine, H. M.* Improved mixed-radix conversion for residue number system architectures [Текст] / H. M. Yassine, W. R. Moore // IEE Proceedings G (Circuits, Devices and Systems). — 1991. — Vol. 138, no. 1. — P. 120—124.

119. *Zhang, W.* An efficient design of residue to binary converter for four moduli set $(2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3)$ based on new CRT II [Текст] / W. Zhang, P. Siy // Information Sciences. — 2008. — Vol. 178, no. 1. — P. 264—279.

Список рисунков

Список таблиц

Приложение А

Результаты моделирования методов определения критических ядер функции ядра Акушского

Таблица 17 — Наборы модулей с динамическим количеством модулей, используемые для моделирования методов определения критических ядер

Размер набора модулей, n	Набор модулей
7	$\{257, 263, 269, 271, 277, 281, 283\}$
8	$\{257, 263, 269, 271, 277, 281, 283, 293\}$
9	$\{257, 263, 269, 271, 277, 281, 283, 293, 307\}$
10	$\{257, 263, 269, 271, 277, 281, 283, 293, 307,337\}$
11	$\{257, 263, 269, 271, 277, 281, 283, 293, 307, 337, 449\}$
12	$\{257, 263, 269, 271, 277, 281, 283, 293, 307, 337, 449, 887\}$

Таблица 18 — Наборы модулей с диапазоном разной длины, используемые для моделирования методов определения критических ядер

Длинна динамического диапазона, бит	Набор модулей
24	$\{255, 256, 257\}$
32	$\{2047, 2048, 2049\}$
40	$\{16383, 16384, 16385\}$
48	$\{65535, 65536, 65537\}$
56	$\{524287, 524288, 524289\}$
64	$\{4194403, 4194304, 4194304\}$

Таблица 19 — Время определения критических ядер с динамическим числом модулей, мс

Количество модулей	7	8	9	10	11	12
Усеченный перебор	1.9157	4.2223	5.4515	5.7955	7.473	7.6149
Метод на основе Теоремы 2.2.1	0.0308	0.0323	0.0385	0.0438	0.0537	0.0602

Таблица 20 — Время определения критических ядер работы с динамическим диапазоном разной длинны, мс

Длинна динамического диапазона, бит	24	32	40	48	56	64
Усеченный перебор	0.2885	1.5381	1.692	1.7088	2.3019	2.2711
Метод на основе Теоремы 2.2.1	0.0078	0.0084	0.0088	0.0096	0.0103	0.0121

Приложение Б

Результаты исследования специальных наборов модулей системы остаточных классов

Таблица 21 — Наборы базисов, используемые для исследования специальных наборов модулей

Номер	Размер динамического диапазона, бит			
	8	16	24	32
1	{7, 8, 9}	{63, 64, 65}	{255, 256, 257}	{2047, 2048, 2049}
2	{7, 8, 9}	{43, 44, 45}	{257, 258, 259}	{1629, 1630, 1631}
3	{65, 9, 7}	{257, 17, 15}	{4097, 65, 63}	{65537, 257, 255}
4	{15, 16, 7}	{63, 64, 31}	{511, 512, 255}	{2047, 2048, 1023}
5	{7, 8, 15}	{63, 64, 127}	{255, 256, 511}	{2047, 2048, 4095}
6	{3, 4, 31}	{15, 16, 511}	{63, 64, 8191}	{255, 256, 131071}
7	{7, 8, 65}	{31, 32, 1025}	{127, 128, 16385}	{511, 512, 262145}
8	{16, 7, 9}	{128, 31, 33}	{1024, 127, 129}	{16385, 511, 513}
9	{7, 8, 9}	{79, 80, 81}	{727, 728, 729}	{2185, 2186, 2187}
10	{3, 4, 5, 9}	{15, 16, 17, 33}	{63, 64, 65, 129}	{255, 256, 257, 513}
11	{3, 4, 5, 7}	{15, 16, 17, 31}	{63, 64, 65, 127}	{255, 256, 257, 511}
12	{3, 4, 5, 17}	{15, 16, 17, 257}	{31, 32, 33, 1025}	{127, 128, 129, 16385}
13	{7, 9, 5, 11}	{31, 33, 29, 35}	{63, 65, 61, 67}	{255, 257, 253, 259}
14	{3, 5, 6, 29}	{7, 9, 10, 125}	{31, 33, 18, 2045}	{127, 129, 263, 2765}
15	{3, 5, 16, 17}	{7, 9, 64, 65}	{15, 17, 256, 257}	{63, 65, 4096, 4097}
16	{3, 4, 5, 31}	{15, 16, 17, 511}	{31, 32, 33, 2047}	{127, 128, 129, 32767}
17	{3, 5, 16, 31}	{7, 9, 64, 127}	{15, 17, 256, 511}	{63, 65, 4096, 8191}
18	{4, 3, 5, 9}	{8, 15, 17, 33}	{4, 127, 129, 257}	{16, 511, 513, 1025}
19	{2, 7, 9, 5}	{4, 31, 33, 17}	{2, 255, 257, 129}	{8, 1023, 1025, 513}
20	{7, 8, 9, 5, 13}	{31, 32, 33, 25, 41}	{127, 128, 129, 113, 145}	{511, 512, 513, 481, 545}
21	{3, 4, 5, 1, 7}	{15, 16, 17, 7, 31}	{31, 32, 33, 15, 63}	{127, 128, 129, 63, 255}
22	{3, 16, 5, 17,127}	{7, 64, 9, 65, 2047}	{7, 64, 9, 65, 2047}	{15, 256, 17, 257, 32767}
23	{7, 8, 9, 5, 13, 17}	{31, 32, 33, 25, 41,17}	{31, 32, 33, 25, 41, 17}	{127, 128, 129, 113, 145, 257}
24	{7, 8, 9, 5, 13, 17}	{7, 16, 9, 5, 13, 17}	{31, 32, 33, 25, 41, 17}	{31, 256, 33, 25, 41, 17}
25	{8, 7, 9, 5, 11}	{16, 15, 17, 13, 19}	{32, 31, 33, 29, 35}	{128, 127, 129, 125, 131}

Таблица 22 — Результаты исследования модулей специального вида, мкс

Номер набора	Размер динамического диапазона, бит											
	8			16			24			32		
	add	sub	mult	add	sub	mult	add	sub	mult	add	sub	mult
1	**143.326**	143.718	**143.81**	144.585	**143.951**	**145.052**	144.392	**144.141**	145.463	**144.636**	**144.852**	**144.898**
2	143.926	144.718	143.91	144.242	144.687	145.198	**144.299**	145.34	145.533	145.543	146.227	146.466
3	144.87	**143.462**	144.132	145.022	145.432	145.73	145.456	146.416	145.473	146.593	147.236	146.651
4	145.673	145.742	146.13	145.996	147.431	147.736	146.994	147.787	147.817	147.434	148.265	148.292
5	146.913	146.072	147.612	145.631	147.673	147.993	147.214	147.879	**145.209**	147.442	148.197	148.650
6	146.48	147.445	147.96	**144.499**	148.114	148.949	147.293	148.926	148.965	147.474	149.393	149.429
7	145.523	146.533	147.598	146.85	148.909	149.527	147.256	149.479	150.402	147.369	149.960	150.910
8	146.521	146.962	147.398	147.579	148.483	148.925	147.471	149.8	149.941	148.007	150.298	150.439
9	145.926	146.718	146.91	146.25	147.003	148.451	147.567	150.289	150.791	148.063	150.794	151.299
10	146.089	145.487	147.003	147.269	147.683	148.655	147.587	150.565	151.653	148.263	151.148	152.17
11	146.135	146.375	147.391	147.898	147.967	149.574	147.77	150.512	148.250	149.385	149.661	150.388
12	148.305	148.754	149.811	148.53	149.303	149.959	148.821	151.279	151.625	149.344	150.117	150.773
13	149.048	149.709	150.265	150.055	150.928	151.222	150.789	152.126	152.379	150.869	152.242	152.536
14	149.746	149.993	151.517	150.623	151.425	152.054	151.439	152.85	153.031	151.437	152.939	153.368
15	152.499	154.159	154.841	152.992	155.122	155.292	153.484	156.38	156.655	153.806	157.023	157.193
16	153.024	155.884	156.789	153.608	155.973	156.851	154.496	156.334	158.603	154.422	157.103	158.105
17	154.305	154.839	155.371	154.762	156.237	156.496	155.396	156.627	157.185	155.576	157.644	158.012
18	154.013	155.909	156.785	155.012	155.988	157.088	155.547	156.857	158.599	155.826	158.257	159.044
19	152.753	153.823	154.086	153.411	153.882	155.415	154.798	156.643	158.900	155.225	156.324	159.285
20	157.198	157.693	158.916	157.674	157.906	159.346	157.308	157.583	160.730	158.488	159.383	160.115
21	158.29	158.53	159.321	158.939	158.645	160.299	159.589	160.311	161.235	159.753	160.513	161.305
22	161.802	161.096	162.981	162.277	161.331	163.274	162.351	163.132	164.795	163.091	163.414	164.289
23	163.071	164.848	165.643	163.403	165.114	166.474	163.598	163.871	167.257	164.217	166.746	167.508
24	163.783	163.936	164.392	164.323	164.562	166.302	164.622	165.194	166.306	165.137	165.887	167.351
25	166.876	166.654	167.138	167.313	167.995	168.756	167.918	168.538	169.385	168.560	169.988	170.598

Приложение В

Результаты моделирования алгоритма построения компактных базисов для системы остаточных классов

Таблица 23 — Результаты моделирования методов построения базисов СОК, мс

Число модулей	Общая фильтрация	Построение на основе чисел Мерсенна	Построение компактных базисов
8	10324	5328	4561
12	25211	9523	7531
16	50164	15433	13467
20	79057	19544	15389
32	165897	28413	23953

Таблица 24 — Наборы модулей, полученные Алгоритмом 15 для моделирования модульных операций

Размер динамического диапазона, бит	Модули
32	$\{1823, 1997, 1997\}$
64	$\{521, 599, 613, 617, 647, 761\}$
96	$\{4217, 4447, 4951, 5279, 5281, 5461, 5521, 6521\}$
128	$\{16633, 17317, 17579, 17747, 20287, 20981, 21067, 22079, 24179\}$

Таблица 25 — Наборы модулей специального вида $\{2^n - 1, 2^n, 2^n + 1\}$ для моделирования модульных операций

Размер динамического диапазона, бит	Модули
32	$\{2047, 2048, 2049\}$
64	$\{4194303, 4194304, 4194305\}$
96	$\{4294967295, 4294967296, 4294967297\}$
128	$\{8796093022207, 8796093022208, 8796093022209\}$

Таблица 26 — Результаты моделирования операции сложения чисел в СОК с использованием вычисленных компактных базисов, мкс

Размер динамического диапазона, бит	32	64	96	128
Наборы модулей специального вида $\{2^n - 1, 2^n, 2^n + 1\}$	143.384	167.211	195.984	223.263
Наборы модулей, вычисленные Алгоритмом 15	140.301	142.184	162.101	193.652

Таблица 27 — Результаты моделирования операции вычитания чисел в СОК с использованием вычисленных компактных базисов, мкс

Размер динамического диапазона, бит	32	64	96	128
Наборы модулей специального вида $\{2^n - 1, 2^n, 2^n + 1\}$	143.641	168.023	196.112	221.287
Наборы модулей, вычисленные Алгоритмом 15	140.871	143.1	161.539	189.975

Таблица 28 — Результаты моделирования операции умножения чисел в СОК с использованием вычисленных компактных базисов, мкс

Размер динамического диапазона, бит	32	64	96	128
Наборы модулей специального вида $\{2^n - 1, 2^n, 2^n + 1\}$	144.544	172.382	198.073	224.632
Наборы модулей, вычисленные Алгоритмом 15	141.593	143.624	163.122	194.162

Приложение Г

Результаты моделирования обратного преобразования из системы остаточных классов в позиционную систему счисления

Таблица 29 — Время выполнения обратного преобразования первый этап сравнения, мкс

Метод	Размерность, бит						
	16	24	32	40	48	56	64
Китайская теорема об остатках	246.3	257	274.6	316.1	345.4	375.9	387.7
Приближенная Китайская теорема об остатках	309.5	329.5	343.3	356.5	368.8	373.4	378.8
Обобщенная позиционная система счисления	304.4	321.1	371.1	396	411.8	429.3	441.4
Диагональная функция	303.7	332.9	345	378.1	401.9	409.8	429.2
Метод на основе КТО и ранга функции ядра Акушского	219.6	232.3	253.6	275.8	297.6	338.9	350.8

Таблица 30 — Время выполнения обратного преобразования второй этап сравнения, мкс

Метод	Количество модулей						
	3	4	5	6	7	8	9
Китайская теорема об остатках	254.8	278.6	313.9	352.7	408.9	466.5	513.9
Приближенная Китайская теорема об остатках	327.2	355.5	379.5	406.6	432.4	463.8	491.4
Обобщенная позиционная система счисления	324.9	377.8	480.2	543.5	651.2	734.2	836.3
Диагональная функция	341.7	376.6	417.5	441.5	482.9	516.7	565.8
Метод на основе КТО и ранга функции ядра Акушского	231.4	245.9	295.7	345.7	380.5	433.2	473.1

Приложение Д

Результаты моделирования определения знака числа в системе остаточных классов

Таблица 31 — Результаты моделирования определения знака числа в СОК с использованием КТО

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	289.7
{29, 67, 64}	298.4
{7, 11, 19, 64}	309.5
24 бита	
{131,257, 1024}	331.30
{29, 63, 67, 256}	316.1
{5, 7, 11, 19, 29, 128}	333.5
32 бита	
{1031, 2039, 4096}	363.6
{131, 257, 511, 512}	375.9
{29, 63, 65, 131, 512}	381.7
40 бит	
{8171, 16383, 16411}	411.6
{511, 1023, 2047, 2048}	393.4
{127, 255, 263, 509, 512}	390.5
{19, 31, 65, 129, 509, 512}	417.33
48 бит	
{65535, 65537, 131072}	434.4
{2047, 4095, 8191, 8192}	446.7
{263, 511, 1023, 1025, 2048}	423.4
{61, 127, 129, 263, 509, 2048}	415.1
{17, 31, 61, 127, 129, 509, 1024}	440.5
56 бит	

{8191, 16383, 32767, 32768}	459.9
{127, 257, 511, 513, 2047, 8192}	460.9
{17, 31, 65, 127, 129, 257, 511, 1024}	439.42
64 бита	
{32767, 65535, 131071, 131072}	464.2
{257, 511, 1025, 2049, 8191, 8192}	463.2
{65, 127, 257, 511, 1023, 2047, 8192}	465.17

Таблица 32 — Результаты моделирования определения знака числа в СОК с использованием приближенной КТО

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	326.9
{29, 67, 64}	336.7
{7, 11, 19, 64}	330.5
24 бита	
{131,257, 1024}	366.6
{29, 63, 67, 256}	360.2
{5, 7, 11, 19, 29, 128}	378.5
32 бита	
{1031, 2039, 4096}	401.4
{131, 257, 511, 512}	396.1
{29, 63, 65, 131, 512}	388.9
40 бит	
{8171, 16383, 16411}	412.6
{511, 1023, 2047, 2048}	407.9
{127, 255, 263, 509, 512}	401.6
{19, 31, 65, 129, 509, 512}	412.3
48 бит	
{65535, 65537, 131072}	420.6
2047, 4095, 8191, 8192	422.5
{263, 511, 1023, 1025, 2048}	417.3
{61, 127, 129, 263, 509, 2048}	425.9

{17, 31, 61, 127, 129, 509, 1024}	428.2
56 бит	
{8191, 16383, 32767, 32768}	443.4
{127, 257, 511, 513, 2047, 8192}	438.9
{17, 31, 65, 127, 129, 257, 511, 1024}	441.9
64 бита	
{32767, 65535, 131071, 131072}	453.5
{257, 511, 1025, 2049, 8191, 8192}	450.6
{65, 127, 257, 511, 1023, 2047, 8192}	455.8

Таблица 33 — Результаты моделирования определения знака числа в СОК с использованием функции Пирло

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	268.1
{29, 67, 64}	275
{7, 11, 19, 64}	290.7
24 бита	
{131,257, 1024}	320.2
{29, 63, 67, 256}	310.5
{5, 7, 11, 19, 29, 128}	330.3
32 бита	
{1031, 2039, 4096}	360.5
{131, 257, 511, 512}	350.1
{29, 63, 65, 131, 512}	345.4
40 бит	
{8171, 16383, 16411}	390.2
{511, 1023, 2047, 2048}	380.8
{127, 255, 263, 509, 512}	375.6
19, 31, 65, 129, 509, 512	395.7
48 бит	
{65535, 65537, 131072}	420
{2047, 4095, 8191, 8192}	430.8

{263, 511, 1023, 1025, 2048}	425.4
{61, 127, 129, 263, 509, 2048}	415.2
{17, 31, 61, 127, 129, 509, 1024}	435.2
56 бит	
{8191, 16383, 32767, 32768}	440.1
{127, 257, 511, 513, 2047, 8192}	450.9
{17, 31, 65, 127, 129, 257, 511, 1024}	445.4
64 бита	
{32767, 65535, 131071, 131072}	455.8
{257, 511, 1025, 2049, 8191, 8192}	452.6
{65, 127, 257, 511, 1023, 2047, 8192}	457.3

Таблица 34 — Результаты моделирования определения знака числа в СОК с использованием функции ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$

Набор модулей СОК	Время, мкс
16 бит	
{63, 64, 65}	228.4
24 бита	
{255,256, 257}	255.7
32 бита	
{2047, 2048, 2049}	288.4
40 бит	
{16383, 16384, 16385}	320.3
48 бит	
{65535, 65536, 65537}	336.6
56 бит	
{262143,1048576,262145}	354.2
64 бита	
{2097151, 4194304, 2097153}	372.4

Таблица 35 — Результаты моделирования определения знака числа в СОК с использованием функции ядра для набора модулей $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$

Набор модулей СОК	Время, мкс
16 бит	
{31, 63, 32}	227.9
24 бита	
{127, 255, 512}	251.5
32 бита	
{1023, 2047, 2048}	285.5
40 бит	
{8191, 16383, 8192}	321.1
48 бит	
{32767, 65535, 131072}	340.3
56 бит	
{262143, 524287, 524288}	351.6
64 бита	
{2097151, 4194303, 2097152}	371.2

Таблица 36 — Время выполнения определения знака числа в СОК, мкс

Размерность, бит	16	24	32	40	48	56	64
Китайская теорема об остатках	289.7	316.1	363.6	390.5	415.1	439.42	463.2
Приближенная Китайская теорема об остатках	326.9	360.2	388.9	401.6	417.3	438.9	450.6
Функция Пирло	268.1	310.5	345.4	375.6	415.2	440.1	452.6
Функция ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$	228.4	255.7	288.4	320.3	336.6	354.2	372.4
Функция ядра для набора модулей $\left\{2^n - 1, 2^{n+1} - 1, 2^{n+a}\right\}$	227.9	251.5	285.5	321.1	340.3	351.6	371.2

Приложение Е

Результаты моделирования сравнения чисел в системе остаточных классов

Таблица 37 — Результаты моделирования сравнения чисел в СОК с использованием КТО

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	349.6
{29, 67, 64}	376.2
{7, 11, 19, 64}	391.4
24 бита	
{131,257, 1024}	418.5
{29, 63, 67, 256}	400.7
{5, 7, 11, 19, 29, 128}	426.1
32 бита	
{1031, 2039, 4096}	466.4
{131, 257, 511, 512}	472.8
{29, 63, 65, 131, 512}	488.6
40 бит	
{8171, 16383, 16411}	526.5
{511, 1023, 2047, 2048}	500.1
{127, 255, 263, 509, 512}	494.8
{19, 31, 65, 129, 509, 512}	534.6
48 бит	
{65535, 65537, 131072}	553.2
{2047, 4095, 8191, 8192}	567.1
{263, 511, 1023, 1025, 2048}	537.9
{61, 127, 129, 263, 509, 2048}	525.2
{17, 31, 61, 127, 129, 509, 1024}	558.4
56 бит	

{8191, 16383, 32767, 32768}	582.6
{127, 257, 511, 513, 2047, 8192}	584.9
{17, 31, 65, 127, 129, 257, 511, 1024}	553.8
64 бита	
{32767, 65535, 131071, 131072}	586.5
{257, 511, 1025, 2049, 8191, 8192}	583.7
{65, 127, 257, 511, 1023, 2047, 8192}	588.9

Таблица 38 — Результаты моделирования сравнения чисел в СОК с использованием приближенной КТО

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	359.5
{29, 67, 64}	368.4
{7, 11, 19, 64}	379.8
24 бита	
{131,257, 1024}	409.7
{29, 63, 67, 256}	389.3
{5, 7, 11, 19, 29, 128}	411.1
32 бита	
{1031, 2039, 4096}	452.5
{131, 257, 511, 512}	458.9
{29, 63, 65, 131, 512}	471.2
40 бит	
{8171, 16383, 16411}	509.0
{511, 1023, 2047, 2048}	487.3
{127, 255, 263, 509, 512}	482.5
{19, 31, 65, 129, 509, 512}	516.2
48 бит	
{65535, 65537, 131072}	536.1
{2047, 4095, 8191, 8192}	548.0
{263, 511, 1023, 1025, 2048}	517.4
{61, 127, 129, 263, 509, 2048}	509.4

{17, 31, 61, 127, 129, 509, 1024}	540.1
56 бит	
{8191, 16383, 32767, 32768}	559.6
{127, 257, 511, 513, 2047, 8192}	570
{17, 31, 65, 127, 129, 257, 511, 1024}	533.1
64 бита	
{32767, 65535, 131071, 131072}	572.9
{257, 511, 1025, 2049, 8191, 8192}	566
{65, 127, 257, 511, 1023, 2047, 8192}	576.4

Таблица 39 — Результаты моделирования сравнения чисел в СОК с использованием ОПСС

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	364.8
{29, 67, 64}	367.4
{7, 11, 19, 64}	396.8
24 бита	
{131,257, 1024}	409.1
{29, 63, 67, 256}	449.1
{5, 7, 11, 19, 29, 128}	466.6
32 бита	
{1031, 2039, 4096}	478.2
{131, 257, 511, 512}	526.3
{29, 63, 65, 131, 512}	564.9
40 бит	
{8171, 16383, 16411}	509.6
{511, 1023, 2047, 2048}	551.6
{127, 255, 263, 509, 512}	579.6
{19, 31, 65, 129, 509, 512}	570.5
48 бит	
{65535, 65537, 131072}	530.6
{2047, 4095, 8191, 8192}	562.2

{263, 511, 1023, 1025, 2048}	589.9
{61, 127, 129, 263, 509, 2048}	584.6
{17, 31, 61, 127, 129, 509, 1024}	605.9
56 бит	
{8191, 16383, 32767, 32768}	561.3
{127, 257, 511, 513, 2047, 8192}	601.5
{17, 31, 65, 127, 129, 257, 511, 1024}	651.3
64 бита	
{32767, 65535, 131071, 131072}	590.6
{257, 511, 1025, 2049, 8191, 8192}	640.6
{65, 127, 257, 511, 1023, 2047, 8192}	672.2

Таблица 40 — Результаты моделирования сравнения чисел в СОК с использованием диагональной функции

Набор модулей СОК	Время, мкс
16 бит	
{19, 29, 128}	325.9
{29, 67, 64}	336.8
{7, 11, 19, 64}	354.6
24 бита	
{131,257, 1024}	392.2
{29, 63, 67, 256}	377.9
{5, 7, 11, 19, 29, 128}	404.6
32 бита	
{1031, 2039, 4096}	437.8
{131, 257, 511, 512}	424.6
{29, 63, 65, 131, 512}	422.6
40 бит	
{8171, 16383, 16411}	475.3
{511, 1023, 2047, 2048}	464.5
{127, 255, 263, 509, 512}	460.9
19, 31, 65, 129, 509, 512	484.3
48 бит	

{65535, 65537, 131072}	514.5
{2047, 4095, 8191, 8192}	528.6
{263, 511, 1023, 1025, 2048}	519
{61, 127, 129, 263, 509, 2048}	504
{17, 31, 61, 127, 129, 509, 1024}	532.8
56 бит	
{8191, 16383, 32767, 32768}	548.7
{127, 257, 511, 513, 2047, 8192}	534
{17, 31, 65, 127, 129, 257, 511, 1024}	540.9
64 бита	
{32767, 65535, 131071, 131072}	551.7
{257, 511, 1025, 2049, 8191, 8192}	555.8
{65, 127, 257, 511, 1023, 2047, 8192}	558.5

Таблица 41 — Результаты моделирования сравнения чисел в СОК с использованием дружественной функции ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$

Набор модулей СОК	Время, мкс
16 бит	
{63, 64, 65}	318.2
24 бита	
{255, 256, 257}	343.7
32 бита	
{2047, 2048, 2049}	372.4
40 бит	
{16383, 16384, 16385}	395.1
48 бит	
{65535, 65536, 65537}	421.6
56 бит	
{524287, 524288, 524289}	454.2
64 бита	
{4194303, 4194304, 4194305}	502.8

Таблица 42 — Время выполнения сравнения чисел в СОК, мкс

Размерность, бит	16	24	32	40	48	56	64
Китайская теорема об остатках	349.6	400.7	466.4	494.8	525.2	553.8	583.7
Приближенная Китайская теорема об остатках	359.5	389.3	452.5	482.5	509.4	533.1	566
Обобщенная позиционная система счисления	364.8	409.1	478.2	509.6	530.6	561.3	590.6
Диагональная функция	325.9	377.9	422.6	460.9	504.0	534.0	551.7
Дружественная функция ядра для набора модулей $\{2^n - 1, 2^{n+a}, 2^n + 1\}$	318.2	343.7	372.4	395.1	421.6	454.2	502.8

Приложение Ж

Свидетельства о государственной регистрации программ для ЭВМ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023619967

Реализация вычисления знака числа в системе остаточных классов

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Кучеров Николай Николаевич (RU), Ширяев Егор Михайлович (RU), Безуглова Екатерина Сергеевна (RU), Луценко Владислав Вячеславович (RU), Гробова Софья Кирилловна (RU)***

Заявка № **2023617877**
Дата поступления **26 апреля 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***17 мая 2023 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 68b80077e14e40f0a94edbd24145d5c7
Владелец Зубов Юрий Сергеевич
Действителен с 2.03.2022 по 26.05.2023

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2023662227

Реализация сравнения чисел в системе остаточных классов

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Кучеров Николай Николаевич (RU), Ширяев Егор Михайлович (RU), Безуглова Екатерина Сергеевна (RU), Луценко Владислав Вячеславович (RU), Колбин Михаил Дмитриевич (RU)***

Заявка № **2023617795**
Дата поступления **26 апреля 2023 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***07 июня 2023 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024619754

Программный комплекс для определения критических ядер при построении функции ядра Акушского

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU)***

Заявка № **2024619005**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***25 апреля 2024 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024619970

Программа для определения четности числа в системе остаточных классов с использованием функции ядра Акушского

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU), Герюгова Айсанат Эдуардовна (RU)***

Заявка № **2024619046**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***02 мая 2024 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024660104

Программа для определения переполнения при сложении чисел в системе остаточных классов с использованием функции ядра Акушского

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU)***

Заявка № **2024619026**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***02 мая 2024 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024660122

Программа для подбора оптимальных весов для функции ядра Акушского с использованием генетического алгоритма

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU), Безуглова Екатерина Сергеевна (RU), Ширяев Егор Михайлович (RU)***

Заявка № **2024619092**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***02 мая 2024 г.***

Руководитель Федеральной службы по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024660861

Программа для реализации итерационного деления чисел в системе остаточных классов с использованием функции ядра Акушского

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU), Безуглова Екатерина Сергеевна (RU), Ширяев Егор Михайлович (RU)***

Заявка № **2024619025**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***14 мая 2024 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов

РОССИЙСКАЯ ФЕДЕРАЦИЯ

СВИДЕТЕЛЬСТВО

о государственной регистрации программы для ЭВМ

№ 2024661039

Программа для реализации масштабирования чисел в системе остаточных классов с использованием функции ядра Акушского

Правообладатель: ***Федеральное государственное автономное образовательное учреждение высшего образования «Северо-Кавказский федеральный университет» (RU)***

Авторы: ***Луценко Владислав Вячеславович (RU), Бабенко Михаил Григорьевич (RU)***

Заявка № **2024619051**
Дата поступления **25 апреля 2024 г.**
Дата государственной регистрации
в Реестре программ для ЭВМ ***15 мая 2024 г.***

Руководитель Федеральной службы
по интеллектуальной собственности

ДОКУМЕНТ ПОДПИСАН ЭЛЕКТРОННОЙ ПОДПИСЬЮ
Сертификат 429b6a0fe3853164baf96f83b73b4aa7
Владелец Зубов Юрий Сергеевич
Действителен с 10.05.2023 по 02.08.2024

Ю.С. Зубов