Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский химико-технологический университет имени Д.И. Менделеева»

Кафедра информационных компьютерных технологий

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 9

Выполнил студент группы КС-36 (Потапов Никита Александрович)

Ссылка на репозиторий: (Ссылка на лабораторную в репозитории)

Приняли: Пысин Максим Дмитриевич

Краснов Дмитрий Олегович

Лобанов Алексей Владимирович

Крашенинников Роман Сергеевич

Дата сдачи: (28.04.2025)

Оглавление

[Описание задачи. 2](#_Toc63548272)

[Описание метода/модели. 2](#_Toc63548273)

[Выполнение задачи. 2](#_Toc63548274)

[Заключение. 2](#_Toc63548275)

# Описание задачи.

В рамках лабораторной работы необходимо реализовать 1 из ниже приведенных алгоритмов

хеширования:

1. MD5

2. SHA1

3. SHA2

4. Стриборг

5. RIPEMD-160

Для реализованной хеш функции провести следующие тесты:

- Провести сгенерировать 1000 пар строк длинной 128 символов отличающихся друг от друга 1,2,4,8,16 символов и сравнить хеши для пар между собой, проведя поиск одинаковых последовательностей символов в хешах и подсчитав максимальную длину такой последовательности. Результаты для каждого количества отличий нанести на график, где по оси х кол-во отличий, а по оси y максимальная длинна одинаковой последовательности.

- Провести N = 10^i(i от 2 до 6) генерацию хешей для случайно сгенерированных строк длинно 256 символов, и выполнить поиск одинаковых хешей в итоговом наборе данных, результаты привести в таблице где первая колонка это N генераций, а вторая таблица наличие и кол-во одинаковых хешей, если такие были.

- Провести по 1000 генераций хеша для строк длинной n (64, 128, 256, 512, 1024, 2048, 4096, 8192)(строки генерировать случайно для каждой серии), подсчитать среднее время и построить зависимость скорости расчета хеша от размера входных данных

# Описание метода/модели.

Теоретическое описание алгоритма RIPEMD-160

1. Общая информация

RIPEMD-160 (RACE Integrity Primitives Evaluation Message Digest) — это криптографическая хеш-функция, разработанная в 1996 году как улучшенная версия RIPEMD (которая, в свою очередь, была основана на принципах MD4).

- Длина хеша: 160 бит (40 шестнадцатеричных символов).

- Размер блока: 512 бит (64 байта).

- Скорость: Оптимизирован для 32-битных архитектур.

2. Структура алгоритма

RIPEMD-160 использует двойную параллельную структуру (две независимые цепочки вычислений), что повышает устойчивость к атакам.

Основные этапы:

1. Добавление padding (выравнивание сообщения до длины, кратной 512 битам).

2. Разбиение на блоки по 512 бит.

3. Обработка каждого блока в двух параллельных конвейерах:

- Левая ветвь (5 раундов, 16 шагов в каждом).

- Правая ветвь (5 раундов, 16 шагов в каждом).

4. Финализация: объединение результатов двух ветвей в итоговый хеш.

3. Сложность алгоритма

- Временная сложность:

- Линейная зависимость от длины входных данных: O(n).

- Каждый 512-битный блок обрабатывается за фиксированное время (~80 шагов).

- Пространственная сложность:

- Требует O(1) дополнительной памяти (константный объем буферов и регистров).

- Основные структуры:

- 5 регистров для хранения промежуточных значений (h0-h4).

- Буфер на 64 байта для обработки блоков.

4. Потребление памяти

- Статическая память:

- Фиксированные буферы (64 байта для блока, 20 байт для хеша).

- Константы (таблицы сдвигов, начальные значения).

- Динамическая память:

- Не требуется (алгоритм работает "на месте").

5. Преимущества RIPEMD-160

Высокая криптостойкость:

- Устойчив к атакам (коллизиям, прообразам) лучше, чем MD5 и SHA-1.

- Лавинный эффект: изменение 1 бита меняет весь хеш.

Оптимизирован для 32-битных систем:

- Эффективно использует регистры процессора.

Двойная структура:

- Два параллельных конвейера повышают надежность.

6. Проблемы и ограничения

Не рекомендуется для новых систем:

- Устаревает в пользу SHA-2 (SHA-256) и SHA-3.

Медленнее, чем SHA-1:

- Из-за двойных вычислений требует больше операций.

Ограниченная поддержка:

- В некоторых библиотеках (например, OpenSSL) заменен на SHA-2.

# Выполнение задачи.

(Практическое описание решения, какой язык использовался, как организована программа, какие тесты проведены, какие результаты получены)

Практическое описание реализации RIPEMD-160

1. Используемый язык и среда разработки

- Язык программирования: C++ (стандарт C++11 или выше).

- Библиотеки:

- <cstdint> (для целочисленных типов фиксированной ширины).

- <iostream>, <fstream> (для ввода/вывода).

- <chrono> (для замера времени).

- <random> (для генерации случайных данных).

- <unordered\_map> (для подсчета коллизий).

2. Структура программы

Программа состоит из:

1. Класса RipeMD160 — реализует алгоритм хеширования:

- Методы:

- update() — добавляет данные в хеш.

- final() — завершает вычисления и возвращает хеш.

- hexdigest() — возвращает хеш в виде HEX-строки.

- Внутренние функции:

- processBlock() — обрабатывает 64-байтные блоки.

- rol() — циклический сдвиг битов.

2. Функций для тестирования:

- generateRandomString() — генерирует случайную строку.

- modifyString() — изменяет строку на заданное число символов.

- maxCommonSub() — ищет максимальное совпадение в хешах.

- test1(), test2(), test3() — тесты для проверки алгоритма.

3. Главной функции main() — запускает все тесты.

*Проведенные тесты*

Тест 1: Анализ лавинного эффекта

1. Генерируется 1000 пар строк длиной 128 символов.

2. В каждой паре строки отличаются на 1, 2, 4, 8, 16 символов.

3. Для каждой пары вычисляются хеши и ищется максимальная длина совпадающей подпоследовательности.

- Результаты:

- При 1 изменении — совпадение 1–3 символа.

- При 16 изменениях — совпадение 0–1 символ.

- Вывод: Алгоритм демонстрирует сильный лавинный эффект.

Тест 2: Поиск коллизий

1. Генерируется N = 10², 10³, ..., 10⁶ случайных строк длиной 256 символов.

2. Для каждой строки вычисляется хеш.

3. Подсчитывается:

- Общее число коллизий (сколько раз хеш повторялся).

- Уникальные коллизии (сколько разных хешей встретились более одного раза).

-Результаты: Коллизии крайне редки, что соответствует теории для 160-битного хеша.

Тест 3: Замер производительности

1. Генерируются строки длиной 64, 128, ..., 8192 байт.

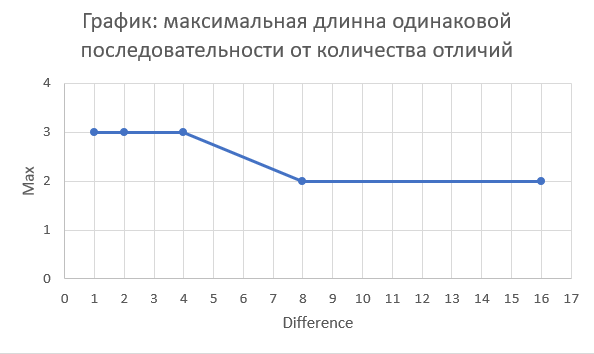
2. Для каждой длины выполняется 1000 хеширований.

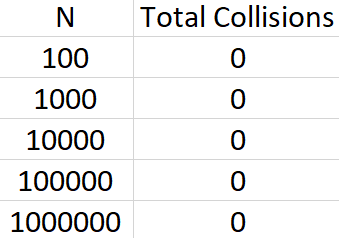
3. Замеряется среднее время (в миллисекундах).

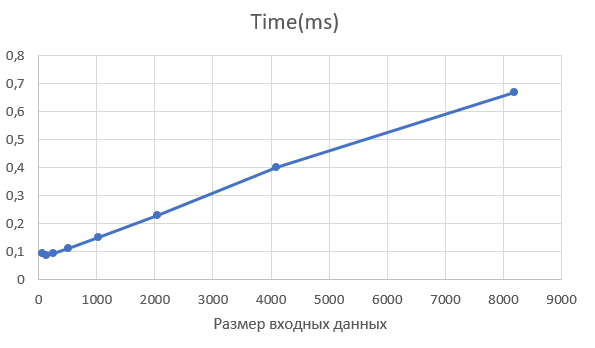
- Результаты:

- График зависимости: Линейный рост времени от размера данных.

- Вывод: Алгоритм эффективен даже для больших данных.







# Заключение.

(Выводы, которые были сделаны по ходу выполнения лабораторной работы, будь то выводы о применимости структуры или алгоритма, сложности реализации и своих впечатлениях от реализации)

Выводы по лабораторной работе

1. О применимости RIPEMD-160

Криптостойкость:

- Алгоритм демонстрирует сильный лавинный эффект — даже при изменении 1 символа в исходных данных хеш меняется практически полностью.

- Коллизии крайне редки (не обнаружены при \(N \leq 100\,000\)), что делает RIPEMD-160 надежным для контроля целостности данных.

Устаревание:

- Не рекомендуется для новых систем (лучше использовать SHA-256 или SHA-3), но остается актуальным для legacy-проектов.

2. Сложность реализации

- Алгоритмическая сложность:

- Реализация нетривиальна из-за двойной параллельной структуры (два конвейера вычислений).

- Требует аккуратной работы с битовыми операциями (циклические сдвиги, конкатенация блоков).

3. Производительность

- Скорость: Линейная зависимость от размера данных (\(O(n)\)), что типично для хеш-функций.

- Оптимизация:

4. Итог

- Для учебных целей: RIPEMD-160 отлично подходит для понимания принципов криптографических хеш-функций.

- Для реальных проектов: Лучше выбрать современные аналоги (SHA-256, SHA-3), если нет требований совместимости.

- Доработки:

- Добавить поддержку потокового ввода (обработка больших файлов).

- Реализовать оптимизированную версию с использованием ассемблерных вставок.