Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Лабораторная работа №11

**ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ХЕШ-ФУНКЦИЙ**

Студент: Сосновец М.И.

ФИТ 3 курс 4 группа

Преподаватель: Нистюк О.А.

Минск 2025

**Цель:** изучение алгоритмов хеширования и приобретение практических навыков их реализации и использования в криптографии.

**Задачи:**

1. Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций вычисления однонаправленных хеш-функций.
2. Освоить методику оценки криптостойкости хеш-преобразований на основе «парадокса дня рождения».
3. Разработать приложение для реализации заданного алгоритма хеширования (из семейств MD и SHA).
4. Оценить скорость вычисления кодов хеш-функций.
5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Теоретические сведения**

Определение 1. Хеш-функция – математическая или иная функция h = H(М), которая принимает на входе строку символов М, называемую также прообразом, переменной длины n и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины l.

Определение 2. Хеширование (или хэширование, англ. hashing) – это преобразование входного массива данных определенного типа и произвольной длины (практически) в выходную битовую строку фиксированной длины.

Преобразования называются хеш-функциями, или функциями свертки, а их результаты называют хешем, хеш-кодом, хеш-таблицей или дайджестом сообщения (англ. message digest).

Определение 3. Криптографическая хеш-функция – это специальный класс хеш-функций, который имеет различные свойства, необходимые для решения задач в области криптографии.

Основные задачи, решаемые с помощью хеш-функций:

* аутентификация (хранение паролей);
* проверка целостности данных;
* защита файлов;
* обнаружение зловредного ПО;
* криптовалютные технологии.

Свойство 1. Детерминированность: независимо от того, сколько раз вычисляется H(M), M – const, при использовании одинакового алгоритма код хеш-преобразования h всегда должен быть одинаковым.

Свойство 2. Скорость вычисления хеша h: если процесс вычисления h недостаточно быстрый, система просто не будет эффективной.

Свойство 3. Сложность обратного вычисления: для известного H(М) невозможно (практически) определить М. Это важнейшее свойство хеш-функции для криптографических применений – свойство односторонности преобразования.

Это означает, что по хеш-коду должно быть практически невозможным восстановление входной строки М.

Определение 4. Коллизией хеш-функции Н называют ситуацию, при которой различным входам (в общем случае – х и у или М ≠ М') соответствует одинаковый хеш-код: H(x) = H(y) или H(М) = = H(М').

Мерой криптостойкости хеш-функции считается вычислительная сложность нахождения коллизии. Для хеш-функций одним из основных средств поиска коллизий является метод, основанный на известной статистической задаче – «парадоксе дня рождения».

Основной постулат парадокса «дней рождения» гласит: в группе минимум из 23 человек с вероятностью более 0,5 день рождения одинаков хотя бы у двух членов группы. Парадоксом является высокая (как кажется на первый взгляд) вероятность наступления указанного события.

Определение 5. Хеш-функция – это функция, выполняющая отображение из множества М в число, находящееся в интервале [0, m – 1]: h: M → [0, m – 1].

Алгоритмы семейства MD-x (2/4/5/6) являются творениями Р. Ривеста; MD – Message Digest. Алгоритм MD6, в отличие от предыдущих версий алгоритма этого семейства, не стандартизован.

Алгоритмы семейства SHA (SHA – Secure Hash Algorithm) являются в настоящее время широко распространенными. По существу, во многих случаях завершился переход от SHA-1 к стандартам версии SHA-2. SHA-2 – собирательное название алгоритмов SHA224, SHA-256, SHA-384 и SHA-512. SHA-224 и SHA-384 являются, по сути, аналогами SHA-256 и SHA-512 соответственно.

Алгоритмы семейства MD входные сообщения максимальной длины 264 – 1 битов (в общем случае – L битов) преобразуют в хеш длиной l = 128 битов. Исключением является последняя (6-я) из версий алгоритма, где длина результирующего хеша может изменяться от 1 до 512 бит.

Базовые алгоритмы обоих рассматриваемых семейств (MD и SHA) условно можно разделить на 5 стадий:

* расширение входного сообщения;
* разбивка расширенного сообщения на блоки;
* инициализация начальных констант;
* обработка сообщения поблочно (основная процедура алгоритма хеширования);
* вывод результата.

Расширение происходит всегда, даже если длина сообщения уже соответствует 448, по модулю 512. Эта операция выполняется следующим образом: один бит «1» добавляется к сообщению, а затем добавляются биты «0», так что длина в битах дополненного сообщения стала конгруэнтной 448 по модулю 512. Добавляется не менее одного бита, но не более 448 битов.

Основой рассматриваемых базовых алгоритмов является модуль, состоящий из циклических преобразований каждого 512-битного блока, который делится на подблоки длиной 32 либо 64 бита (в алгоритмах SHA-512, SHA-384, SHA-512/256, SHA-512/224).

SHA – это MD-4 с добавлением расширяющего преобразования, дополнительного этапа и с улучшенным лавинным эффектом. MD-5 – это MD-4 с улучшенным битовым хешированием, дополнительным этапом и улучшенным лавинным эффектом.

**Практическое задание**

1. Разработать оконное приложение, реализующее один из алгоритмов хеширования из указанного преподавателем семейства (MD или SHA; или иного). При этом можно воспользоваться доступными готовыми библиотеками. Язык программирования – на свой выбор. Приложение должно обрабатывать входные сообщения, длина которых определяется спецификацией на реализуемый алгоритм.

2. Оценить быстродействие выбранного алгоритма хеширования.

В лабораторной работе использовался алгоритм SHA-256, который представляет собой криптографическую хеш-функцию, которая преобразует входное сообщение произвольной длины в фиксированный 256-битный хеш. Структура алгоритма SHA-256 выглядит следующим образом:

1. Подготовка данных (алгоритм начинает с добавления специального окончания к входным данным)

* В конце сообщения добавляется 1-бит.
* После этого добавляются биты 0, чтобы длина стала кратной 512 (остаток от деления на 512).
* В финале добавляется 64-битное представление исходной длины сообщения. Это позволяет алгоритму учитывать полный объем входных данных.

1. Разбиение на блоки:

* Полученное сообщение разбивается на блоки по 512 бит (64 байта).
* Каждый блок обрабатывается независимо, что упрощает параллельные вычисления.

1. Инициализация хеша:

* Для работы используются 8 фиксированных 32-битных значений (хеш-регистры), заданных алгоритмом. Эти начальные значения обеспечивают одинаковый старт для всех сообщений.

1. Обработка блоков (расширение данных)

* Каждый 512-битный блок преобразуется в 64 32-битных слова. Первые 16 слов берутся из блока напрямую, а остальные вычисляются с использованием специальных функций сдвига и сложения.
* 64 итерации:
* Для каждого слова из 64 проводятся операции с использованием временных переменных a,b,c,d,e,f,g,ha, b, c, d, e, f, g, ha,b,c,d,e,f,g,h.
* На каждом шаге используются математические операции: побитовые сдвиги, логические функции и сложения с заранее заданными константами.

1. Обновление состояния:

* После обработки всех 64 шагов временные значения a,b,c,d,e,f,g,ha, b, c, d, e, f, g, ha,b,c,d,e,f,g,h добавляются к хеш-регистрам.
* Эти обновления отражают обработанную часть сообщения и передаются для обработки следующего блока.

1. Генерация хеша:

* После обработки всех блоков 8 хеш-регистров объединяются в одну 256-битную строку.
* Итоговый результат представляет собой уникальный хеш, который зависит только от исходных данных.

В лабораторной работе реализация алгоритма SHA-256 была сделана с помощью библиотеки crypto. Реализация показана в листинге 1.1.

|  |
| --- |
| const hash = crypto.createHash('sha256').update(body).digest('hex'); |

Листинг 1.1 - Реализация алгоритма SHA-256с помощью библиотеки crypto

Работа приложения показана на рисунке 1.1.

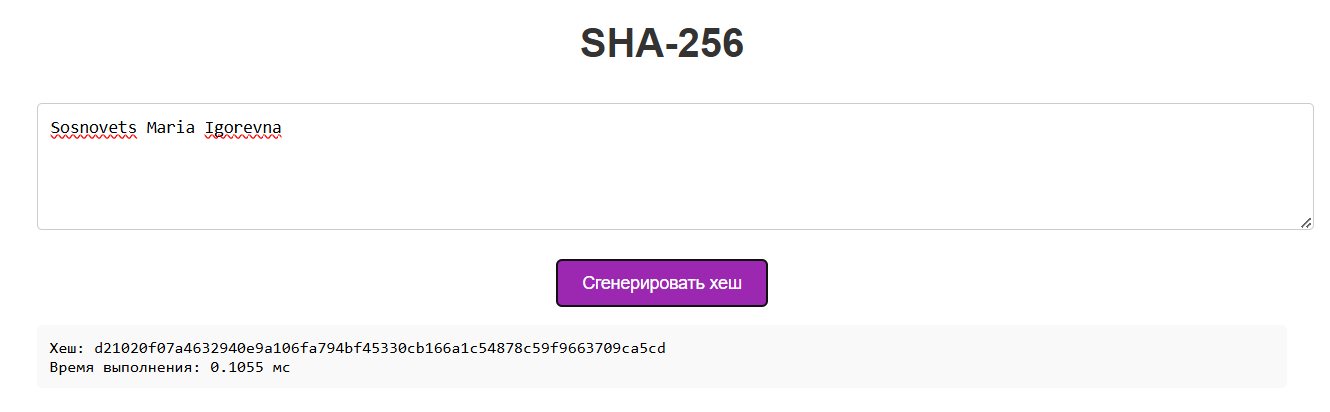


Рисунок 1.1 – Результат выполнения алгоритма SHA-256

Если в алгоритме SHA-256 изменить хотя бы один символ во входных данных, результат хеширования полностью изменится. Это ключевая особенность криптографических хеш-функций, которая называется лавинным эффектом. Результат работы при изменении одного символа показан на рисунке 1.2

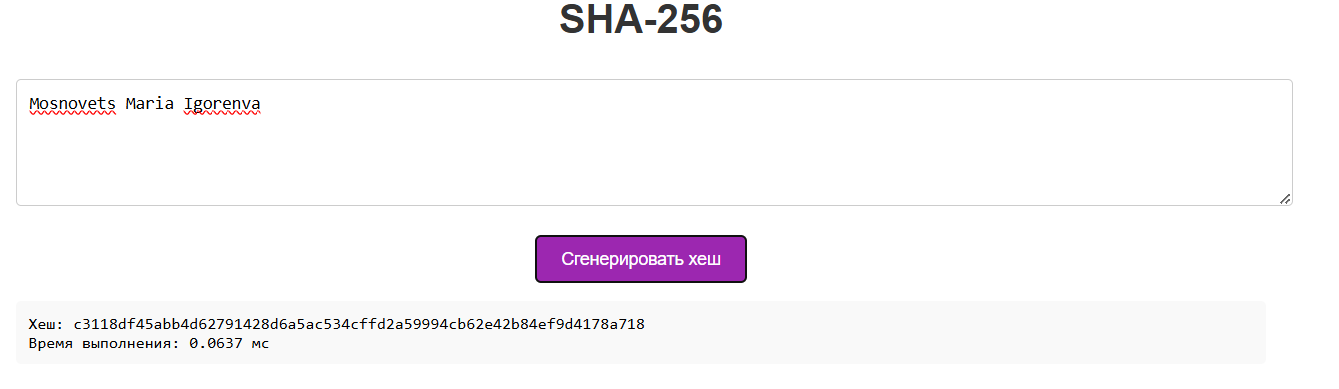


Рисунок 1.2 – Результат выполнения алгоритма SHA-256 при изменении одного символа

На рисунке 1.2 видно, что хеш изменился. Такая особенность гарантирует уникальность хеша для каждого набора данных и обеспечивает высокую стойкость алгоритма, делая невозможным предсказать, как изменится результат.

Оценка быстродействия алгоритма SHA-256 показана на рисунке 1.2 в виде графика.

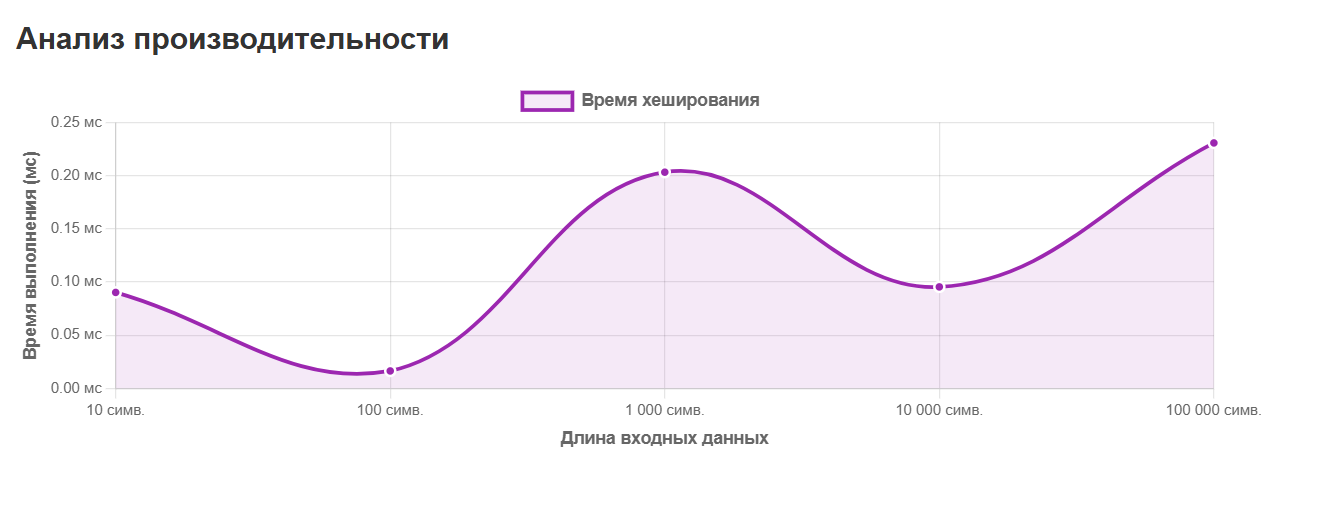


Рисунок 1.2 – Анализ производительности с разным количеством данных

Как можно заметить время выполнения хеширования слегка увеличивается с ростом длины входных данных, так больший объем данных требует больше операций обработки. Алгоритм SHA-256 демонстрирует стабильную и высокую производительность даже при увеличении длины входных данных.

**Вывод:** в ходе выполнения работы была изучена теория и алгоритмы работы однонаправленных хеш-функций, в частности алгоритма SHA-256. Были приобретены навыки практической реализации хеширования и оценки его производительности. Разработано приложение, демонстрирующее работу алгоритма SHA-256, включая расчет хеша и анализ скорости его выполнения при различных объемах входных данных.