МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Исследование криптографических хеш-функций

Студент: Евсеенко В. П.

ФИТ 3 курс 4 группа

Преподаватель: Савельева М. Г.

Минск 2025

Содержание

[Введение 3](#_Toc198167606)

[1 Теоретические сведения 4](#_Toc198167607)

[2 Практическое задание 6](#_Toc198167608)

[Оценка времени 8](#_Toc198167609)

[Вывод 9](#_Toc198167610)

# **Введение**

Цель лабораторной работы:

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

Задачи:

* закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций вычисления однонаправленных хеш-функций;
* освоить методику оценки криптостойкости хеш-преобразований на основе «парадокса дня рождения»;
* разработать приложение для реализации заданного алгоритма хеширования (из семейств MD и SHA);
* оценить скорость вычисления кодов хеш-функций.

# **1 Теоретические сведения**

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*.

Хеширование – 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.

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.

К основным свойствам хеш-функций можно отнести следующие.

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.

Свойство 4. Даже минимальные изменения в хешируемых данных (*М* ≠ *М*') должны изменять хеш: *Н*(*M*) ≠ *Н*(*М*').

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*').

Свойство 5. Коллизионная устойчивость (стойкость). Зная *М*, трудно найти такое *М*' (*М* ≠ *М*'), для которого *H*(*М*) = *H*(*М*'). Если последнее равенство выполняется, то говорят о коллизии 1-го рода. Если случайным образом выбраны два сообщения (*М* и *М*'), для которых *H*(*М*) = *H*(*М*'), говорят о коллизии 2-го рода.

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. При этом предполагается, что:

• в этой группе нет близнецов;

• люди рождаются независимо друг от друга, т. е. дата (день) рождения любого человека не влияет на дату рождения другого;

• люди рождаются равномерно и случайно, т. е. люди с равной вероятностью могут рождаться в любой день года; с формальной точки зрения это означает, что вероятность *р*1 рождения отдельно выбранного члена группы (как и любого человека) в любой выбранный день равна *р*1 = 1 / 365 (хотя известно, что в реальности рождение людей не совсем соответствует такому предположению).

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

# **2 Практическое задание**

В начале, было необходимо разработать приложение, выполняющее хеширование с помощью алгоритмов семейства SHA и MD.

Для реализации алгоритмом использовался модуль crypto.

Функция хеширования алгоритмом SHA-256 принимает на вход оригинальный текст и осуществляет хеширование. Код функции представлен в листинге 2.1.

|  |
| --- |
| app.post('/sha', (*req*: *Request*, *res*: *Response*): *void* => {      const message = *req*.body.message as *string*;      if (!message) {  *res*.status(400).json({ error: 'Message is required' });          return;      }      const startTime = performance.now();      const hash = *crypto*.createHash('sha256').update(message).digest('hex');      const endTime = performance.now();      const hashingTime = endTime - startTime;      const length = hash.length / 2;  *res*.json({ hash, length, hashingTime });  }); |

Листинг 2.1 – Функция хеширования алгоритмом SHA-256

Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.2.



Рисунок 2.1 – Результат работы приложения

Длина хеша – 32 байта или 256 бит.

Время хеширования сообщения – менее 1 мс, что отражает высокую временную эффективность алгоритма.

Функция хеширования алгоритмом MD5 принимает на вход оригинальный текст и осуществляет хеширование. Код функции представлен в листинге 2.2.

|  |
| --- |
| app.post('/md5', (*req*: *Request*, *res*: *Response*): *void* => {      const message = *req*.body.message as *string*;      if (!message) {  *res*.status(400).json({ error: 'Message is required' });          return;      }      const startTime = performance.now();      const hash = *crypto*.createHash('md5').update(message).digest('hex');      const endTime = performance.now();      const hashingTime = endTime - startTime;      const length = hash.length / 2;  *res*.json({ hash, length, hashingTime });  }); |

Листинг 2.2 – Функция хеширования алгоритмом MD5

Результат работы приложения с исходным текстом «Evseenko Viktoriya Pavlovna» представлен на рисунке 2.2.

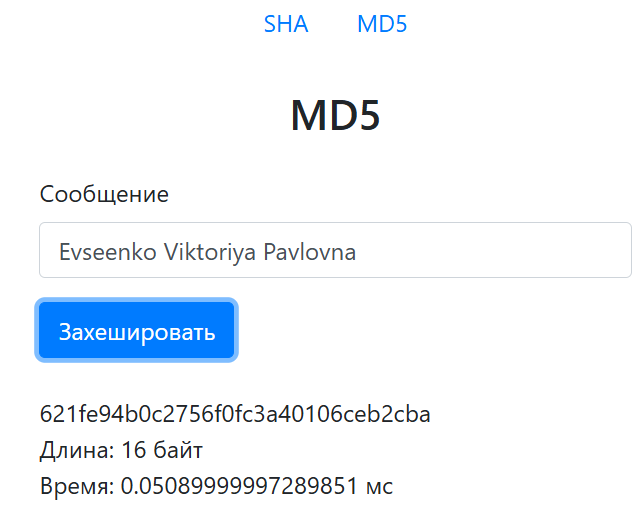


Рисунок 2.2 – Результат работы приложения

Можно заметить, что длина хеша равна 16 байтами или 128 битам.

# **Оценка времени**

Приложение выполняет оценку скорости алгоритма хеширования семейств MD и SHA Результат нескольких попыток хеширования представлен на рисунке 2.3.

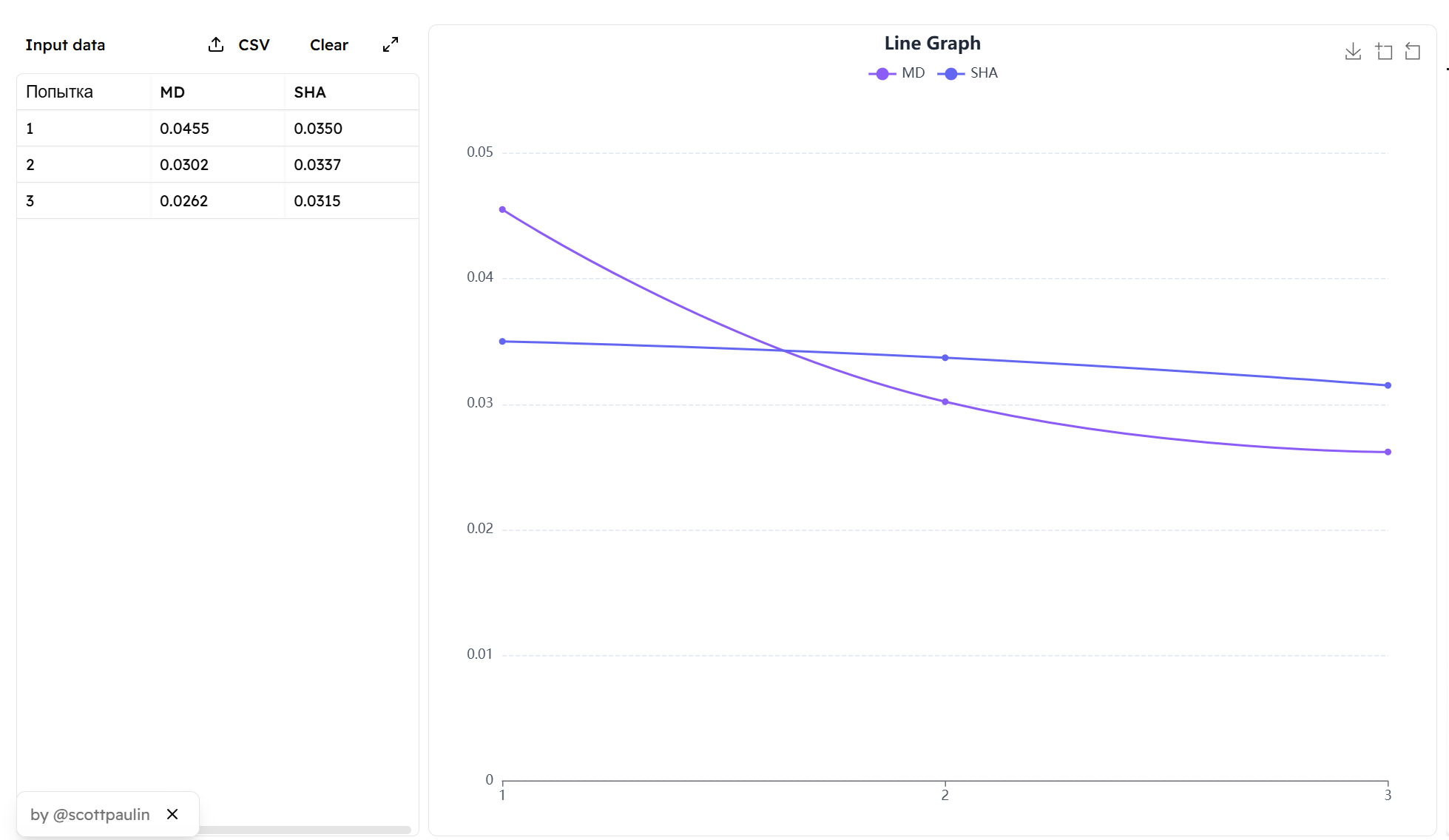


Рисунок 2.3 – Оценка времени операций хеширования

# **Вывод**

В ходе выполнения лабораторной работы была достигнута цель по изучению алгоритмов хеширования и приобретению практических навыков их реализации и использования в криптографии. Были успешно реализованы и протестированы хеш-функции из семейств MD (MD5) и SHA (SHA-256) с использованием языка TypeScript и библиотеки Express.js.

В процессе работы были закреплены теоретические знания об однонаправленных хеш-функциях, их свойствах и алгоритмах реализации. Особое внимание уделено оценке криптостойкости хеш-преобразований на основе «парадокса дня рождения», что позволило глубже понять уязвимости, возникающие при недостаточной длине хеш-кодов.

Было разработано веб-приложение, позволяющее пользователю ввести произвольное сообщение и получить его хеш-сумму с использованием выбранного алгоритма. Также была реализована функция измерения времени, затраченного на вычисление хеша, что позволило провести сравнительный анализ скорости работы различных алгоритмов.

В результате экспериментов были получены значения времени хеширования и длины выходных хеш-кодов, что подтвердило различия в производительности и степени криптостойкости алгоритмов MD5 и SHA-256. Итоги лабораторной работы оформлены в виде описания архитектуры приложения, пошаговой методики проведения экспериментов и анализа полученных результатов.