Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ  
ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

[Кафедра информационных](https://www.belstu.by/fakultety/fit/vm) систем и технологий

Специальность 1-40 01 01 Программное обеспечение информационных технологий

**Отчёт по лабораторной работе №11**

по дисциплине Информационная безопасность

Тема: Исследование криптографических хеш-функций

        Исполнитель:

Студент 3 курса группы 4

Станчик Максим Андреевич

Руководитель:

Ассистент Савельева М. Г.

Минск, 2025

**Лабораторная работа №11**

**Цель:** исследование 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

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 *р1*рождения отдельно выбранного члена группы (как и любого человека) в любой выбранный день равна *р1* = 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].

# Ход работы

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

Для реализации алгоритмов использовалась библиотека java.security.

Статический финальный массив строк hashAlgorithms содержит два элемента: "SHA3-256" и "MD5". Используется для хранения названий алгоритмов хэширования, которые используются в методе generateHash для генерации хэша. Код массива представлен в листинге 2.1.

public static final String[] hashAlgorithms = {

"SHA3-256",

"MD5"

};

Листинг 2.1 – Код массива, который хранит названия используемых в программе алгоритмов генерации хэша

Метод createSalt предназначен для генерации соли (salt) — случайной последовательности байтов, которая используется в криптографических операциях, таких как хэширование. Соль помогает защитить пароли и предотвратить атаки с использованием предвычисленных хэш-таблиц (rainbow tables). В качестве параметров принимает размер соли в байтах, который передается в метод. Это определяет, сколько случайных байтов будет сгенерировано. Метод возвращает строку, представляющую собой сгенерированную соль в формате Base64. Код метода представлен в листинге 2.2.

public String createSalt(int size) {

SecureRandom rng = new SecureRandom();

byte[] buff = new byte[size];

rng.nextBytes(buff);

return Base64.getEncoder().encodeToString(buff);

};

Листинг 2.2 – Код метода для генерации соли

Метод generateHash генерирует хэш для входного значения (input), добавляя к нему соль, сгенерированную с помощью метода generateSalt или заданную пользователем через терминал. В качестве параметров принимает исходное значение для хэширования, алгоритм хэширования ("SHA3-256" или "MD5"), и соль, которая добавляется к исходному значению перед хэшированием. Если указанный алгоритм недоступен, выбрасывается исключение NoSuchAlgorithmException, которое обрабатывается в блоке catch. Возвращает строку, представляющую хэш, полученный после объединения input и salt. Код метода представлен в листинге 2.3.

public String generateHash(String input, String algorithm, String salt) {

return generateHash((input + salt), algorithm);

}

public String generateHash(String input, String algorithm) {

try {

MessageDigest digest = MessageDigest.getInstance(algorithm);

byte[] hash = digest.digest(input.getBytes());

StringBuilder hexString = new StringBuilder();

for (byte b : hash) {

String hex = Integer.toHexString(0xff & b);

if (hex.length() == 1) hexString.append('0');

hexString.append(hex);

}

return hexString.toString();

} catch (NoSuchAlgorithmException e) {

e.printStackTrace();

}

return null;

};

Листинг 2.3 – Код метода для генерации хэша

Результат работы приложения с исходным текстом «Maxim Stanchik Andreevich» представлен на рисунке 2.1.

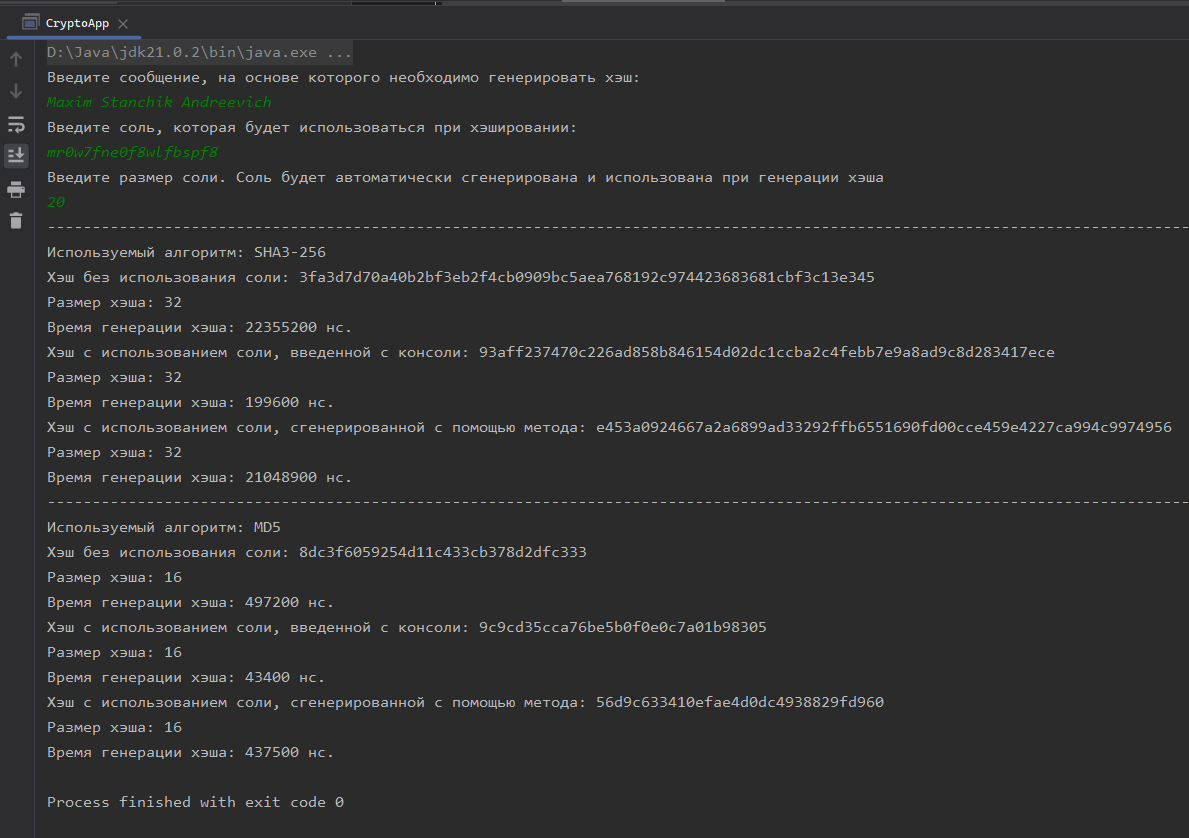


Рисунок 2.1 – Результат работы приложения

# Время вычисления хеша

Построим график от количества символов в исходном сообщении при фиксированной длине соли. График зависимости представлен на рисунке 3.1.



Рисунок 3.1– График зависимости

Из него видно, что текст длинной до 1000000 символов хешируется довольно быстро, однако с увеличением размера исходного текста возрастает время хеширования.

**Вывод:** в ходе выполнения лабораторной работы был изучен принцип реализации алгоритмов хеширования SHA3-256 и MD5. Также было разработано приложение, выполняющее хеширования на основе данных алгоритмов. Была оценена скорость хеширования различными алгоритмами.