**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-38..... Прилепский Артем Сергеевич
Ссылка на репозиторий: .....github.com/news1d/Algorithms_and_structures

Приняли: .....Пысин Максим Дмитриевич
.....Краснов Дмитрий Олегович
.....Лобанов Алексей Владимирович
.....Крашенинников Роман Сергеевич

Дата сдачи: .....28.04.2023

# Оглавление

## Описание задачи.

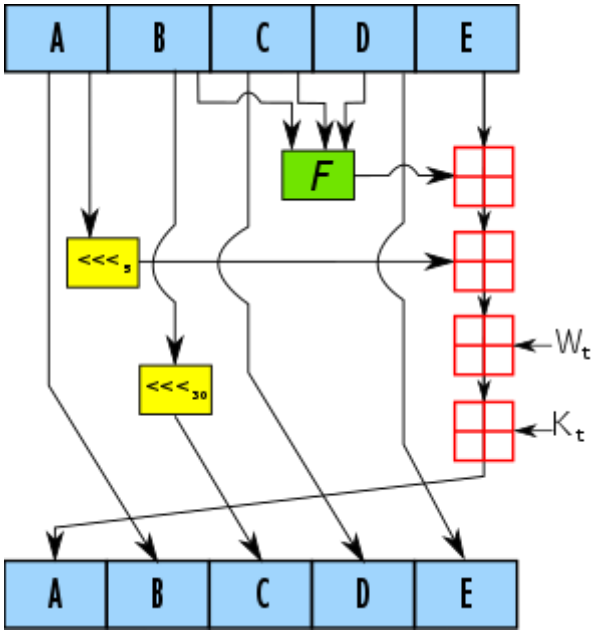
В рамках лабораторной работы необходимо реализовать SHA-1 алгоритм хеширования.

Для реализованной хеш функции провести следующие тесты:

- 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 x 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.

**SHA-1** реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами, хеш-блок $M_i$ равен $h_i = f(M_i, h_{i-1})$. Хеш-значением всего сообщения является выход последнего блока.




*Одна итерация алгоритма SHA-1*

## Выполнение задачи.

Для реализации программы был выбран язык программирования Python.

1) Класс хеш-функции SHA-1:

```
class SHA1:
    def __init__(self, data):
        self.data = data.encode('utf-8')
        self.h = [0x67452301, 0xEFCDAB89, 0x98BADCFE, 0x10325476, 0xC3D2E1F0]

    # Функция для сдвига бит
    def rotate(self, n, b):
        return ((n << b) | (n >> (32 - b))) & 0xFFFFFFFF

    # Функция для добавления дополнительных байтов, чтобы общее количество байтов
стало кратно 64
    def padding(self):
        # Добавляем 1 бит и затем нули до тех пор, пока длина данных в битах не
станет равной 448
        padded_data = self.data + b"\x80" + b"\x00" * (63 - (len(self.data) + 8)
% 64)
        # Добавляем длину исходного сообщения (до предварительной обработки) как
целое 64-битное значение, в битах
        padded_data += struct.pack(">Q", 8 * len(self.data))
        return padded_data

    # Функция разделения данных на блоки
    def split_blocks(self, padded_data):
        # Создание пустого списка блоков данных
        blocks = []
        # Цикл для разбивки данных на блоки по 64 байта
        for i in range(0, len(padded_data), 64):
            block = padded_data[i: i + 64]
            blocks.append(block)
        return blocks

    # Функция расширения блока
    def expand_block(self, block):
        # Распаковываем блок сообщения в список из 16 значений по 32 бита и
добавляем в конец 64 нуля
        w = list(struct.unpack(">16L", block)) + [0] * 64
        # Добавляем ещё 64 значения (изменяем добавленные 0 значения)
        for i in range(16, 80):
            # Применение функции циклического сдвига к числам
            w[i] = self.rotate((w[i - 3] ^ w[i - 8] ^ w[i - 14] ^ w[i - 16]), 1)
        return w

    # Функция окончательного хеширования
    def final_hash(self):
        # Добавление дополнительных байтов
        padded_data = self.padding()
        # Разделение данных на блоки
        blocks = self.split_blocks(padded_data)
        # Цикл по блокам данных
        for block in blocks:
            # Расширение блока
            expanded_block = self.expand_block(block)
            # Инициализация промежуточных значений хеша
            a, b, c, d, e = self.h
            # Цикл по значениям блока
            for i in range(0, 80):
                if i < 20:
                    f = (b & c) | ((~b) & d)
                    k = 0x5A827999
```

```
                elif i < 40:
                    f = b ^ c ^ d
                    k = 0x6ED9EBA1
                elif i < 60:
                    f = (b & c) | (b & d) | (c & d)
                    k = 0x8F1BBCDC
                elif i < 80:
                    f = b ^ c ^ d
                    k = 0xCA62C1D6
                # Обновляем регистры хеша в соответствии с алгоритмом хэширования
                a, b, c, d, e = (self.rotate(a, 5) + f + e + k +
expanded_block[i] & 0xFFFFFFFF, a, self.rotate(b, 30), c, d)
            # Обновляем регистры хеша для следующего блока
            self.h = (
                self.h[0] + a & 0xFFFFFFFF,
                self.h[1] + b & 0xFFFFFFFF,
                self.h[2] + c & 0xFFFFFFFF,
                self.h[3] + d & 0xFFFFFFFF,
                self.h[4] + e & 0xFFFFFFFF,
            )
        # Возвращаем окончательный хеш-код в виде строки.
        return ("{:08x}" * 5).format(*self.h)
```

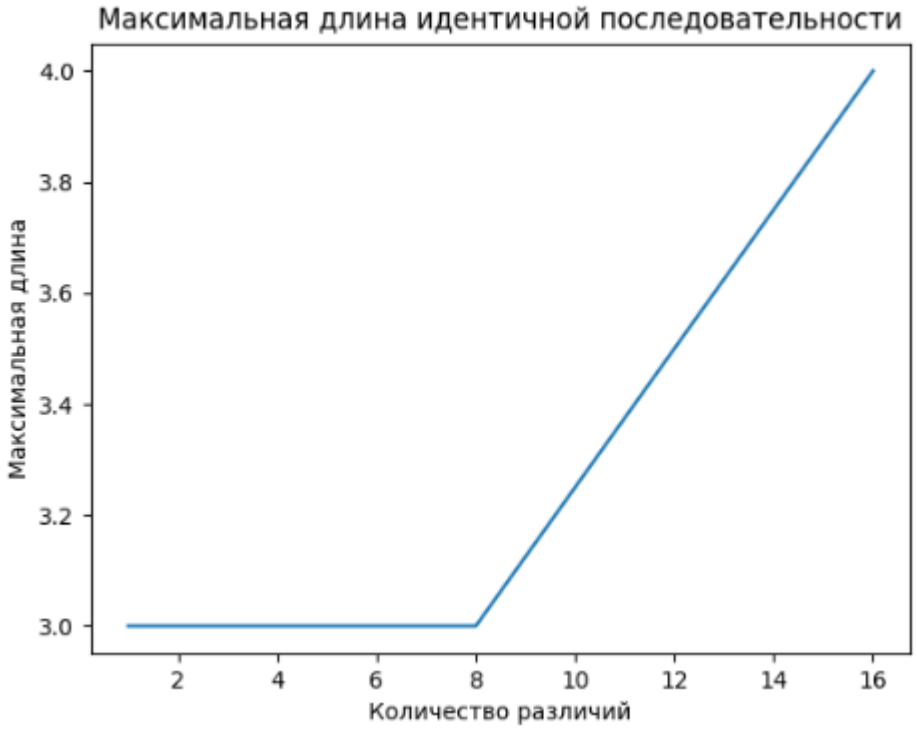



*График зависимости максимальной длины идентичной последовательности от кол-ва различных символов строки*

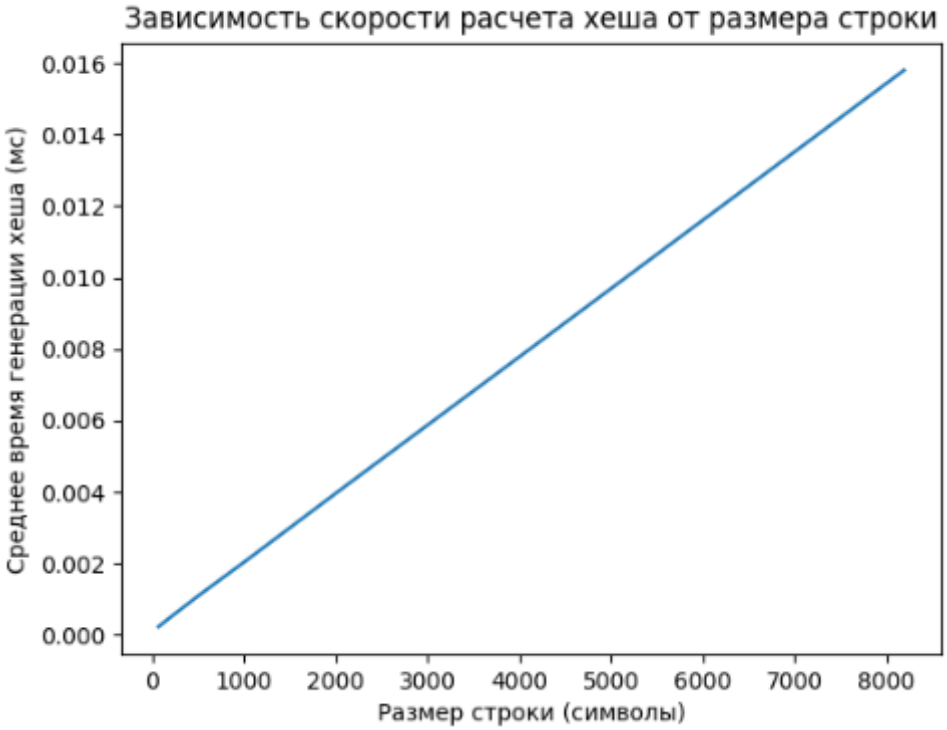



*График зависимости скорости расчета хеша от размера строки*

| Кол-во генераций | Кол-во дубликатов |
|---|---|
| 100 | 0 |
| 1000 | 0 |
| 10000 | 0 |
| 100000 | 0 |
| 1000000 | 0 |

*Таблица зависимости кол-ва дубликатов хеша от кол-ва генераций хеша*

## Заключение.

В настоящее время метод SHA-1 не считается безопасным, так как известно, что метод подвержен коллизиям, то есть возможности получения двух различных сообщений с одинаковым хеш-кодом, хотя и в результате нашего тестирования подобного выявлено не было.

В целом, использование метода SHA-1 на данный момент не рекомендуется, и его использование должно быть заменено на более современные и безопасные алгоритмы хеширования.