**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**Факультет прикладной математики – процессов управления**

**отчет**

**по лабораторной работе**

**по дисциплине «Алгоритмы и структуры данных»**

**на тему «Исследование хеш-функций с различными вводными условиями»**

**1 вариант**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22.Б15 |  | Агишев А.Б. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г.**

**Содержание**

[**1.** **Цель работы** 2](#_Toc152024077)

[**3.** **Теоретическая часть** 2](#_Toc152024078)

[**4.** **Описание алгоритма** 4](#_Toc152024079)

[**5.** **Описание программы** 7](#_Toc152024080)

[**5.1** **Описание функций** 7](#_Toc152024081)

[**6.** **Исследование по использованию соли** 8](#_Toc152024082)

[**7.** **Заключение** 8](#_Toc152024083)

# **Цель работы**

Разработать программу, которая способна расшифровать набор данных, зашифрованный с помощью хеш-функции с использованием модификатора входа – соли, а также проанализировать решение аналогичной задачи при различных условиях.

1. **Задачи**

Приобрести и закрепить знания в области шифрования телефонных номеров. Написать программу для деобезличивания датасета. Протестировать программу на заданном варианте, а также с еще минимум 3-мя различными хеш-функциями, которыми зашифрован исходный деобезличенный набор. Определить от чего меняется скорость расшифровки.

# **Теоретическая часть**

*MD5 (Message Digest Algorithm 5)*:

* Описание: MD5 — это криптографический хэш-алгоритм, созданный для вычисления 128-битного хэша из произвольного объема данных. Он производит фиксированный 32-значный шестнадцатеричный хэш-код.
* Применение: использовался широко в прошлом для хэширования паролей и проверки целостности данных. Однако, MD5 уязвим к коллизиям (различным данным, создающим одинаковый хэш), и поэтому не рекомендуется для безопасных приложений.

*SHA-1 (Secure Hash Algorithm 1)*:

* Описание: SHA-1 — криптографический хэш-алгоритм, представляющий собой 160-битный хэш. Он является улучшением MD5 и производит более длинные хэши.
* Применение: ранее широко использовался, но с течением времени выявлены уязвимости, и он не рекомендуется для криптографических целей. SHA-1 подвержен коллизиям.

*SHA-256 (Secure Hash Algorithm 256-bit)*:

* Описание: SHA-256 — часть семейства SHA-2, созданного Национальным институтом стандартов и технологии (NIST). SHA-256 производит 256-битный хэш, что делает его более криптографически стойким по сравнению с MD5 и SHA-1.
* Применение: широко используется для безопасности данных, включая цифровые подписи, блокчейн и другие области, где требуется высокая стойкость к коллизиям.

*Брутфорс (brute force)* представляет собой метод взлома, при котором злоумышленник осуществляет попытки нахождения правильного пароля, путем перебора всех возможных комбинаций. Этот метод является прямолинейным и требовательным к ресурсам, поскольку включает в себя систематический перебор всех вариантов паролей. Применение брутфорса обосновано в ситуациях, когда отсутствуют альтернативные методы взлома, и может быть эффективен при работе с простыми паролями или недостаточными политиками безопасности.

*Соль (salt)* представляет собой случайную или уникальную строку, которая добавляется к паролю перед его хэшированием. Использование соли решает проблему коллизий (возможности различных данных создавать одинаковый хэш) и повышает стойкость к атакам, таким как использование таблиц радужных хэшей (rainbow tables). Применение соли имеет важное значение для предотвращения использования предварительно вычисленных таблиц, в которых содержатся хэши для всех возможных паролей. Она также усиливает уровень защиты от брутфорс-атак, даже при использовании слабых паролей. Однако стоит отметить, что для полноценной реализации соли требуется уникальная соль для каждого пользователя. Это также вносит дополнительные сложности в управление данными о соли. Несмотря на применение соли, она сама по себе не исключает возможность брутфорс-атак, но делает их более ресурсоемкими.

# **Описание алгоритма**

1. Деобезличивание датасета номеров телефонов с помощью инструмента hashcat.
2. Анализ номеров телефонов для выявления соли с использованием эталонных номеров.
3. Запись номеров телефонов без соли, полученных после деобезличивания, в файл для последующего использования.
4. Применение хеш-функций MD5, SHA1, SHA256 для хеширования номеров телефонов с различными значениями.
5. Проведение деобезличивания для каждого нового полученного датасета номеров телефонов с последующим анализом полученных результатов.

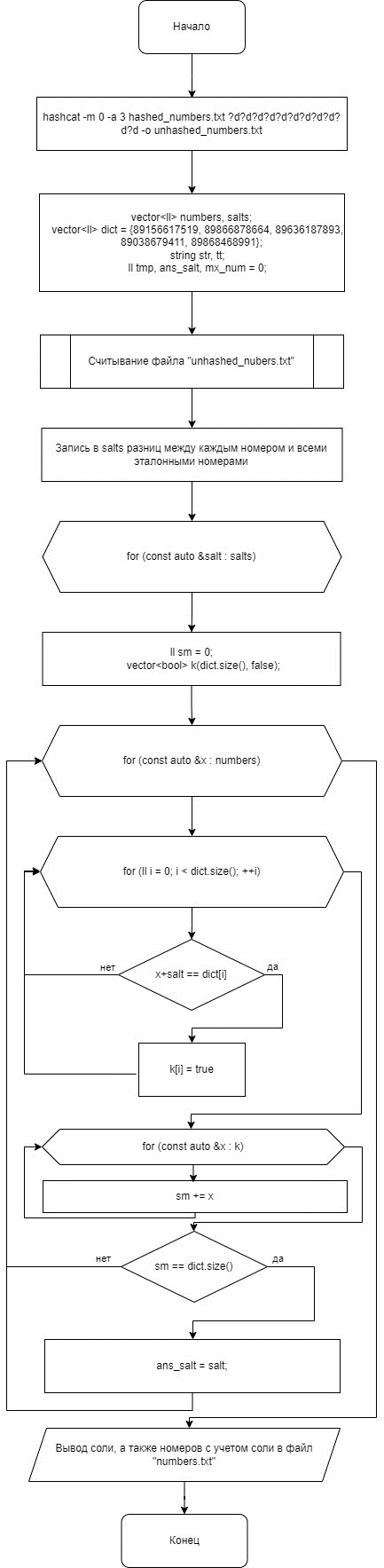


Рисунок 4.1. Блок-схема алгоритма деобезличивания датасета и нахождения соли.

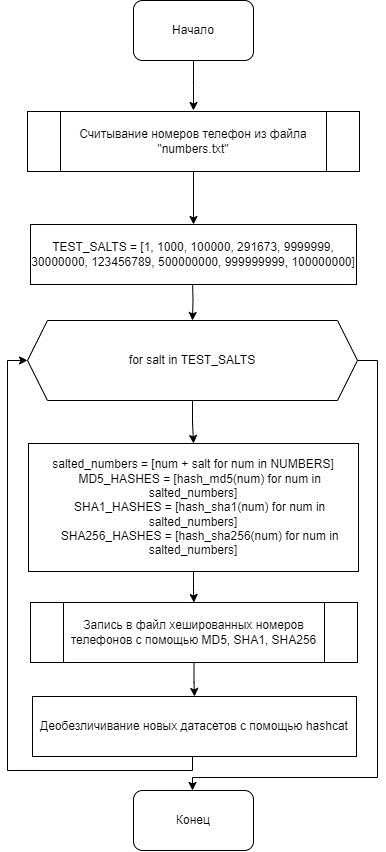


Рисунок 4.2. Блок-схема алгоритма деобезличивания датасета и нахождения соли.

Соль: 88720184.

# **Описание программы**

## **Описание функций**

В программе используется 5 функций. В таблице 5.1 представлено описание функций «main.cpp».

*Таблица 5.1. Описание функций «main.cpp»*

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| main | — | Деобезличивание датасета, поиск соли |

В таблице 5.2 представлено описание функций «hash.py»

*Таблица 5.2. Описание функций «hash.py»*

|  |  |  |
| --- | --- | --- |
| Имя функции | Входные данные | Описание функции |
| hash\_md5 | Номер телефона с солью | Хеширование с помощью md5 |
| hash\_sha1 | Номер телефона с солью | Хеширование с помощью sha1 |
| hash\_sha256 | Номер телефона с солью | Хеширование с помощью sha256 |
| write\_to\_file | Список хешированных номеров телефонов | Запись данных в файл |

# **Исследование по использованию соли**

В таблице 6.1 представлена зависимость времени необходимого на расшифровку таблицы, содержащей номера телефонов от разных хеш-функций и соли:

|  |  |  |  |
| --- | --- | --- | --- |
| Значение соли | MD5, мин:сек | SHA1, мин:сек | SHA256, мин:сек |
| 1 | 9:43 | 19:21 | 38:29 |
| 1000 | 10:08 | 19:32 | 38:40 |
| 100000 | 10:54 | 20:06 | 39:01 |
| 291673 | 11:36 | 20:59 | 39:40 |
| 9999999 | 11:57 | 21:32 | 40:21 |
| 30000000 | 11:19 | 20:21 | 39:53 |
| 100000000 | 11:01 | 20:07 | 39:22 |
| 123456789 | 11:51 | 21:15 | 40:02 |
| 500000000 | 11:14 | 20:22 | 39:49 |
| 999999999 | 12:10 | 22:07 | 41:44 |

*Таблица 6.1. Время расшифровки хеш-функций в зависимости от размера соли.*

На основании предоставленной таблицы зависимости времени, затраченного на расшифровку хеш-значений с использованием hashcat, можно сделать следующие выводы. Во-первых, видно, что время взлома увеличивается с увеличением значения соли. Это подтверждает эффективность использования соли для повышения уровня безопасности паролей.

Во-вторых, наблюдается различная динамика времени взлома в зависимости от типа хеш-функции. Например, хеш-функция MD5 требует меньше времени на взлом по сравнению с SHA1 и SHA256. Это может быть обусловлено различиями в алгоритмах хеширования и их вычислительной сложности.

Кроме того, стоит отметить, что время взлома увеличивается с увеличением длины и сложности пароля. Например, в случае использования числа 999999999, время взлома значительно выше, чем при использовании числа 1. Это свидетельствует о необходимости использования более длинных и сложных паролей для обеспечения надежной защиты.

Дополнительно стоит отметить, что чем больше значение числа в пароле, тем больше времени требуется на его взлом. В частности, наблюдается, что при увеличении количества цифр в числе, например, при использовании значения 999999999, время взлома продолжает увеличиваться. Этот тренд подчеркивает важность использования не только сложных, но и достаточно длинных паролей с целью обеспечения эффективной защиты от попыток взлома.

# **Заключение**

В рамках данной работы была успешно достигнута поставленная цель - разработка алгоритма и программы, которая способна расшифровать набор данных, зашифрованный с помощью хеш-функции с использованием модификатора входа – соли. Данная лабораторная работа позволила не только приобрести, но и закрепить знания в области шифровании номеров телефонов. Также было проанализировано время расшифровки хеш-функций в зависимости от размера соли.