**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЁТ**

**по лабораторной работе №3**

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

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

**Студент гр. 23Б15-пу**

**Беляева А.П.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

**Оглавление**

Цель работы...................................................................................................................3

Описание задачи............................................................................................................3

Теоретическая часть......................................................................................................4

Основные шаги программы...........................................................................................4

Блок-схема программы..................................................................................................5

Описание программы.....................................................................................................7

Рекомендации пользователя........................................................................................7

Рекомендации для программиста................................................................................8

Исходный код программы..............................................................................................8

Контрольный пример...................................................................................................8

Вывод.............................................................................................................................10

Источники......................................................................................................................10

**Цель работы**

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

**Описание задачи (формализация задачи)**

Задача заключается в расшифровке набора данных, зашифрованного с помощью хеш-функций и соли, и определении зависимости времени расшифровки от выбранного метода хеширования и длины соли. Для успешного взлома необходимо протестировать программу на заданном наборе данных, а также с другими хеш-функциями, и выяснить минимальное количество известных значений, необходимых для расшифровки всего датасета.

**Теоретическая часть**

Хеш-функции преобразуют произвольные данные в зашифрованное значение фиксированной длины. Включение соли увеличивает уникальность результата, добавляя случайные данные перед хешированием. Это усложняет задачу обратного восстановления, так как один и тот же телефонный номер с разными солями будет преобразован в разные хеши. В данной работе используется hashcat для выполнения перебора возможных значений и восстановления исходных данных.

**Основные шаги программы**

· Загрузка файла с зашифрованными данными.

· Применение hashcat для попытки восстановления исходных данных.

· Сохранение расшифрованного набора данных.

· Вычисление соли для полученного результата.

# Блок схема программы



Рис 1. Блок-схема обезличивания

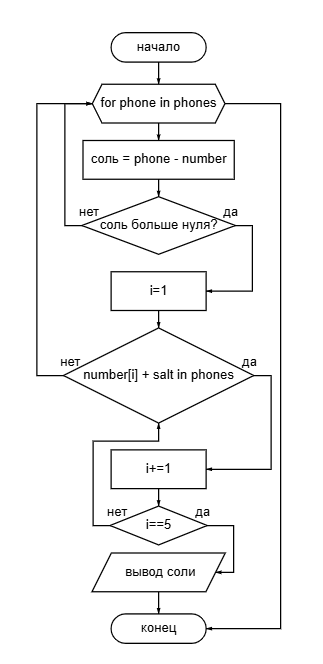


Рис 2. Блок-схема поиска соли

# Описание программы

Программа разработана на языке Python с использованием библиотек pandas, os, hashlib, tkinter для работы с данными. Пользователю предоставляется возможность выбора хеш-функции, после чего программа автоматически применяет выбранные действия и высчитываются результаты.

Таблица 1. main.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| salting | Вычисление соли | Int |
| find\_salt | Нахождение соли из файла | None |
| Load\_file | Загрузка файла | None |
| hashing | Хеширование файла и деобезличивание | None |

# Рекомендации пользователя

· Перед запуском программы поместите файл main.py в каталог с hashcat

· Перед началом работы убедитесь, что hashcat установлен на вашем устройстве.

· Загрузите файл с зашифрованными номерами через интерфейс программы.

· Выберите алгоритм шифрования, используемый для расшифровки, и запустите процесс.

· После завершения программы проверьте файл с расшифрованными данными и соль.

# Рекомендации программиста

# Убедитесь, что все пути к файлам корректны и hashcat установлен по указанному пути.

# Можно добавить дополнительные алгоритмы, расширив поддержку других хеш-функций.

# В дальнейшем для улучшения безопасности добавьте возможность изменения параметров hashcat для более гибкого подхода к перебору.

# Исходный код программы

**https://github.com/hysterria/Depersonalization.git**

# Контрольный пример

# 1. Запуск программы

# Для запуска программы используйте файл **main.py**.

2. Загрузка файла

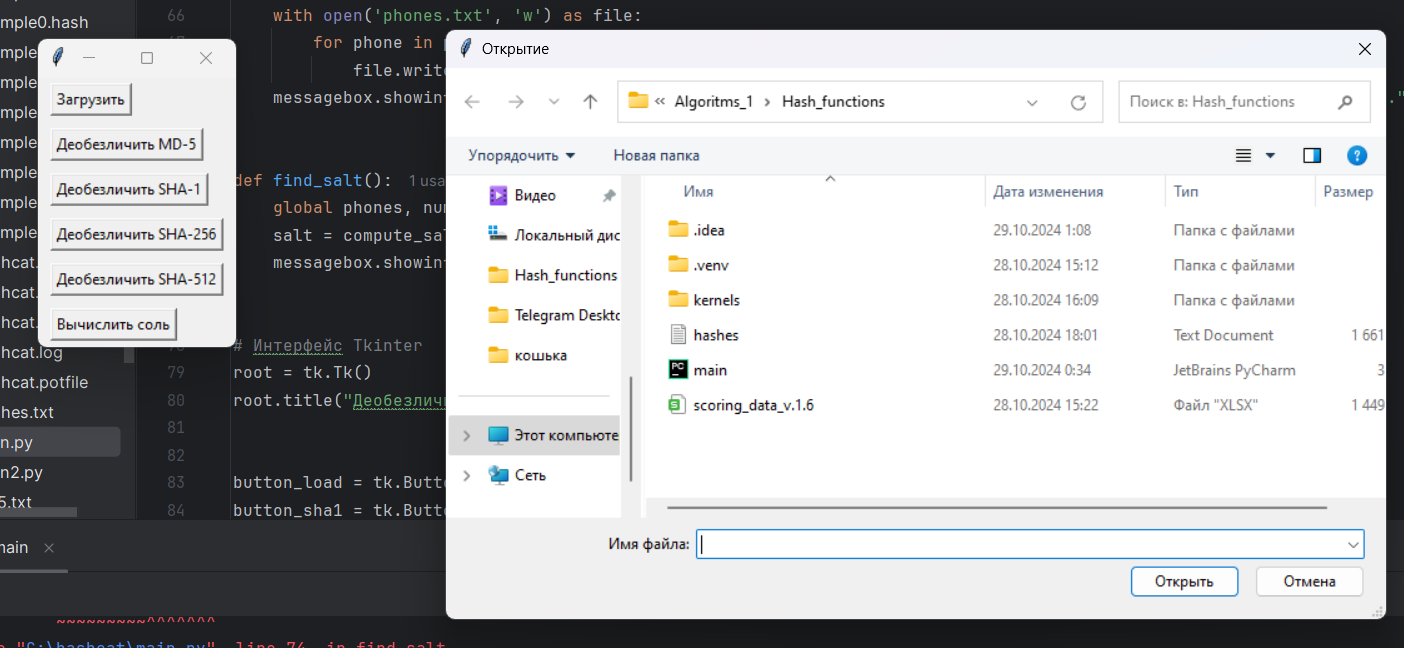


Рис 3. пример ввода

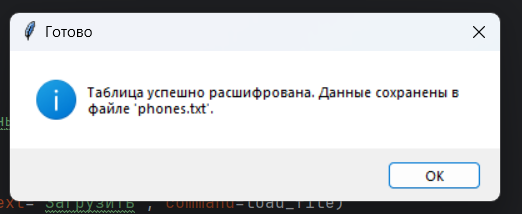
1. Деобезличивание  
   

Рис 4. сообщение о готовности

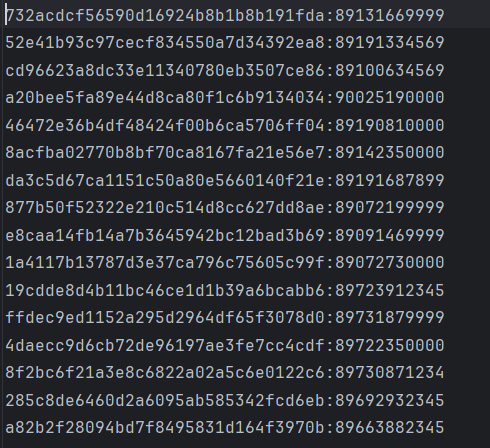


Рис 5. пример деобезличенного датасета

1. Вычисление соли

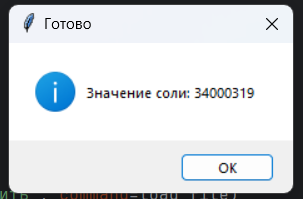


Рис 6. вывод соли

# Факторы, влияющие на скорость расшифровки:

# Вид соли:

* Отсутствие соли: Когда данные хешируются без соли, то один и тот же вход (например, одинаковый номер телефона) всегда будет давать одинаковый хеш. Это делает данные более уязвимыми для атак, основанных на радужных таблицах, и ускоряет взлом, поскольку достаточно лишь одного раза подобрать исходное значение для одного хеша.
* Фиксированная соль для всех записей: При использовании одной и той же соли для всех записей, требуется меньше вычислений, чем при использовании уникальной соли для каждого номера. В этом случае можно повторно использовать найденную пару "номер + соль" для других хешей, тем самым ускоряя взлом.
* Уникальная соль для каждой записи: Уникальные соли для каждого номера значительно замедляют взлом, так как каждый хеш требует отдельного вычисления с его индивидуальной солью, исключая возможность использовать радужные таблицы и увеличивая нагрузку на систему.

# Длина соли:

* Короткая соль: Более короткая соль (например, 4–5 символов) потребует меньше времени для обработки, поскольку хеш-функции обрабатывают меньше данных. Однако, короткая соль также более уязвима к атакам полного перебора.
* Длинная соль: Увеличение длины соли делает взлом сложнее, особенно при использовании уникальных солей для каждой записи. Более длинные строки требуют больше вычислений, увеличивая время, затраченное на расшифровку.

# Выбор хеш-функции:

* Быстрые функции (например, MD5, SHA-1): Эти функции сравнительно быстрее, поскольку были разработаны для производительности и имеют более короткий цикл вычислений. В результате их легче взломать методом перебора.
* Медленные функции (например, bcrypt, scrypt, Argon2): Эти функции разработаны для устойчивости к атакам перебором и используют намеренное замедление с помощью дополнительных параметров (например, "cost factor" в bcrypt). Для взлома таких хешей требуется гораздо больше времени, особенно при увеличении значений настроек.

# Сколько нужно знать телефонов, чтобы 100% взломать весь датасет?

Чтобы с вероятностью 100% расшифровать хеши и найти соль нужно знать 3 телефона.

# 

# Вывод

# В результате лабораторной работы было исследовано влияние хеш-функций и соли на скорость расшифровки данных. Было установлено, что скорость расшифровки зависит как от используемого алгоритма, так и от длины соли. Программа доказала свою эффективность в деобезличивании данных и позволяет наглядно видеть, как различные условия влияют на восстановление информации.

# Источники

1. Pandas – pandas documentation // Pandas URL:  
    <https://pandas.pydata.org/>.
2. Hashlib - hashlib documentation // Hashlib URL:  
   <https://sky.pro/media/kak-rabotat-s-modulem-hashlib-v-python/>
3. Hashcat - hashcat documentation // Hashcat URL:  
   <https://hashcat.net/hashcat/>