**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

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

**ОТЧЕТ**

**по лабораторной работе №3**

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

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

**Вариант – 1**

**Студент гр. 23Б16-пу**

**Крылов А. С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

[Цель работы 3](#_Toc183115431)

[Описание задачи 3](#_Toc183115433)

[Теоретическая часть 3](#_Toc183115434)

[Основные шаги работы 4](#_Toc183115435)

[Описание функций: 5](#_Toc183115436)

[Рекомендации программиста 8](#_Toc183115437)

[Исходный код программы 8](#_Toc183115438)

[Исследование 10](#_Toc183115439)

[Понятие "Соли": 11](#_Toc183115442)

[Вывод 14](#_Toc183115443)

# **Цель работы**

Целью лабораторной работы является расшифровка набора данных, зашифрованного с помощью хеш-функции с использованием модификатора входа – соли, а также анализ решения аналогичной задачи при различных условиях.

# **Описание задачи**

1. Изучить особенности шифрования телефонных номеров.
2. Написать программу для деобезличивания датасета.
3. Протестировать программу на выданном варианте.
4. Протестировать с еще минимум 3-мя различными хеш-функциями, которыми зашифрован исходный деобезличенный набор.
5. Написать от чего меняется скорость расшифровки. Влияние вида соли, длины соли и хеш-функции на скорость расшифровки датасета.
6. Написать отчет.

# **Теоретическая часть**

Для выполнения работы использован HashCat и код на python.

# 

# **Основные шаги работы**

Дешифровка номеров с примененной солью с помощью атаки по маске с помощью hashcat. Далее, так как известно несколько заведомо необезличенных номеров из списка, с их помощью вычисляем соль. Затем применяем обратный ход модификатора ко всем номерам, получая деобезличенный датасет. Далее проверка 3-мя различными хеш-функциями, которыми зашифрован исходный деобезличенный набор.

**Блок схема программы**



Рис 1. Блок-схема программы.

**Описание програмы:**

Программная реализация написана на языке Python 3.11.9 с использованием следующих библиотек: os, time, hashlib. С помощью os в код интегрирована работа с консольной версией программы HashCat.

## Описание функций:

Таблица 1. Описание функции 'clean'

|  |
| --- |
| Читает каждую строку из входного файла, удаляет непечатаемые символы и записывает очищенные строки в выходной файл. |
| Тип возвращаемого значения: None |

Таблица 2. Параметры 'clean'

|  |  |
| --- | --- |
| Параметр | Описание |
| input\_file (str) | Путь к входному файлу, содержащему текст для очистки. |
| output\_file (str) | Путь к выходному файлу, куда записывается очищенный текст. |

Таблица 3. Описание функции 'remove\_duplicates'

|  |
| --- |
| Читает каждую строку из входного файла, удаляет непечатаемые символы и дубликаты, записывает уникальные строки в выходной файл. |
| Тип возвращаемого значения: None |

Таблица 4. Параметры функции 'remove\_duplicates'

|  |  |
| --- | --- |
| Параметр | Описание |
| input\_file (str) | Путь к входному файлу, содержащему текст с возможными дубликатами строк. |
| output\_file (str) | Путь к выходному файлу, куда записывается текст без дубликатов. |

Таблица 5. Описание функции 'found\_salt'

|  |
| --- |
| Описание функции: |
| Определяет значение соли, добавленной к 'phones', сравнивая с 'phones\_with\_salt' через проверку множеством. Возвращает определенное значение соли. |
| Тип возвращаемого значения: int (значение соли) |

Таблица 6. Описание функции 'found\_salt'

|  |  |
| --- | --- |
| Параметр | Описание |
| phones (list[int]) | Список заранее известных номеров из датасета |
| phones\_with\_salt (list[int]) | Список номеров телефонов с добавленной солью. |

**Описание переменных:**

* phones (list[int]) — список номеров телефонов, загруженных из файла phones.txt. Это исходные номера без добавленной соли.
* phones\_with\_salt\_a (list[str]) — список строк из файла cracked\_hashes.txt, который содержит хешированные телефоны с добавленной солью. Каждая строка в этом списке содержит хеш и последние 11 цифр (номер телефона с солью).
* phones\_with\_salt (list[int]) — список телефонов с солью, извлеченных из последних 11 символов каждой строки в phones\_with\_salt\_a. Все значения приведены к целочисленному типу.
* salt (int) — найденное значение соли, которое используется для анонимизации номеров телефонов. Вычисляется функцией found\_salt, сравнивая оригинальные номера с номерами, к которым добавлена соль.
* phones\_wo\_salt (list[int]) — список номеров телефонов без соли, который получается путем вычитания найденной соли из номеров телефонов с солью.
* lines (list[str]) — список строк, которые будут записаны в файл phones\_out.txt. Каждая строка содержит хеш телефона без соли, соответствующий оригинальному номеру.
* hash (str) — строка для хранения хеша, сгенерированного для каждого номера телефона в форматах SHA-1, SHA-256 и SHA3-256.
* main\_start\_time, main\_end\_time, main2\_end\_time, main3\_end\_time (float) — временные метки начала и окончания различных этапов выполнения программы, включая расшифровку данных, вычисление соли и деанонимизацию номеров. Время фиксируется с помощью time.time(), который возвращает число с плавающей запятой.
* sha1\_start\_time, sha1\_end\_time (float) — временные метки для вычисления времени, затраченного на расшифровку хешей в формате SHA-1.
* sha256\_start\_time, sha256\_end\_time (float) — временные метки для вычисления времени расшифровки хешей в формате SHA-256.
* sha3\_start\_time, sha3\_end\_time (float) — временные метки для вычисления времени расшифровки хешей в формате SHA3-256.

**Рекомендации пользователя**

* Для запуска программы, убедитесь, что у вас установлен Python, HashCat и необходимые библиотеки. Запустите код в среде разработки или командной для его выполнения.
* Запуск производится файлом laba3.py.
* Периодически проверяйте корректность данных перед использованием программы.

# **Рекомендации программиста**

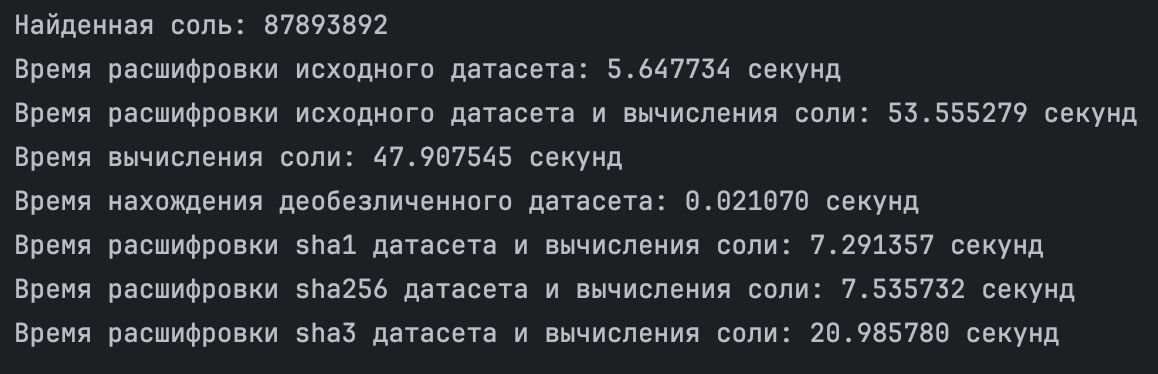
* Поддерживайте актуальность используемых библиотек и версии Python для сохранения актуальности и работоспособности кода на современных системах.
* Периодически проводите тестирование на различных входных данных для обеспечения надежности и корректности программы.

# **Исходный код программы**

[**https://github.com/akryloff/spbu-algorithms-and-data-structures**](https://github.com/akryloff/spbu-algorithms-and-data-structures)

**Контрольный пример**

При выполнении программа выведет результат работы: соль с применением которой хешировался датасет, время выполнения разделов программы.

****Рис 1. Результаты работы программы.

Исследование

## Хеш-функции:

Хеш-функция — это математический алгоритм, преобразующий входные данные произвольной длины в выходную строку фиксированной длины, называемую хешем или дайджестом. Основные свойства хеш-функций:

Необратимость. Из хеша нельзя получить исходные данные даже теоретически. Слишком много информации отбрасывается в процессе; это не зашифровка информации.

Детерминированность. Если подать хеш-функции одинаковые данные, то и хеш у них будет одинаковым. Именно это свойство позволяет использовать хеши для проверки подлинности информации.

Уникальность. Идеальная хеш-функция выдает стопроцентно уникальный результат для каждого возможного набора данных. В реальности такое невозможно, и иногда случаются коллизии — одинаковые хеши для разных сведений. Но существующие хеш-функции достаточно сложны, поэтому вероятность коллизии сводится к минимуму.

Разнообразие. Даже если два набора информации различаются одним-двумя символами, их хеши будут кардинально разными. У них не будет общих блоков, по ним невозможно будет понять, что исходные данные схожи.

Высокая скорость генерации. Это в целом свойство любых хешей: в отличие от зашифрованных версий файлов, они генерируются быстро, даже если входной массив данных большой.

### Примеры хеш-функций:

MD5 и SHA-1: Устаревшие функции, не рекомендуемые из-за уязвимостей к коллизиям.

SHA-256 и SHA-3: Современные криптографические хеш-функции, обеспечивающие высокую степень безопасности.

bcrypt, scrypt, Argon2: Хеш-функции, специально разработанные для хеширования паролей, учитывающие фактор времени и памяти для усложнения атак.

Техника взлома Brute Force:

Brute Force (грубый перебор) — метод взлома, при котором злоумышленник систематически перебирает все возможные комбинации входных данных до тех пор, пока не найдет совпадение с целевым хешем. Этот метод гарантирует нахождение исходных данных при достаточном времени и ресурсах, но его эффективность снижается с увеличением сложности и длины входных данных.

Особенности:

Вычислительная затратность: требует значительных ресурсов и времени, особенно для длинных и сложных паролей.

Полная переборка: проверяются все возможные комбинации символов в заданном диапазоне (буквы, цифры, спецсимволы).

Применение автоматизации: используются специализированные программы и скрипты для автоматизации процесса перебора.

Способы ускорения Brute Force:

Словарные атаки: Использование списков распространенных паролей или слов для сокращения количества перебираемых комбинаций.

Распределенные вычисления: Распределение задачи между несколькими компьютерами или использование облачных сервисов.

Аппаратное ускорение: Применение графических процессоров (GPU) или специализированного оборудования (ASIC) для ускорения вычислений.

## Понятие "Соли":

Соль — это случайная или псевдослучайная строка данных, которая добавляется к паролю или другим входным данным перед их хешированием. Основная цель использования соли — усложнить задачу злоумышленнику при попытке восстановить исходные данные из хеша.

Преимущества использования соли:

Защита от предварительно вычисленных атак: Соль делает менее эффективными атаки с использованием заранее подготовленных таблиц хешей (rainbow tables), так как для каждой соли требуется отдельная таблица.

Уникальность хешей: Даже при использовании одинаковых паролей у разных пользователей их хеши будут различаться благодаря разным солям.

Усложнение Brute Force атак: Увеличивает количество комбинаций, которые необходимо перебрать злоумышленнику.

Оптимальное использование соли:

Уникальная соль для каждой записи: Каждому паролю должна соответствовать своя уникальная соль.

Достаточная длина соли: Рекомендуется использовать соли длиной не менее 16 байт для повышения безопасности.

Хранение соли: Соль обычно хранится вместе с хешем в базе данных, но это не снижает ее эффективности.

Использование вместе с замедленными хеш-функциями: Такие функции, как bcrypt или Argon2, дополнительно усложняют задачи взлома.

Вид соли:

Отсутствие соли: Если соль не используется, одинаковые входные данные будут приводить к одинаковым хешам. Это делает систему уязвимой для атак с использованием предварительно вычисленных таблиц (например, rainbow tables), что значительно ускоряет расшифровку.

Статическая соль: Использование одной и той же соли для всех записей лучше, чем ее отсутствие, но все же представляет риск. Злоумышленник, узнав эту соль, сможет применить ее ко всем хешам одновременно, что ускорит атаку.

Уникальная соль на запись: наиболее безопасный вариант. Каждая запись имеет свою собственную соль, что делает предварительные атаки практически невозможными, так как злоумышленнику придется обрабатывать каждую запись отдельно, существенно замедляя процесс.

Длина соли:

Короткая соль: меньше вариантов комбинаций, что упрощает перебор возможных значений.

Длинная соль: увеличивает энтропию и количество возможных комбинаций, что усложняет и замедляет атаки методом перебора.

Влияние на скорость: Увеличение длины соли может незначительно увеличить время вычисления хеша, но это пренебрежимо мало по сравнению с существенным выигрышем в безопасности.

## Вывод

В рамках данной работы были исследованы хеш-функции и их применение. Отработано использование программы HashCat в комплексе с использованием python. В ходе выполнения работы было выяснено: если соль отсутствует или одинакова для всех записей, то, зная достаточное количество пар "телефон-хеш", злоумышленник может вычислить алгоритм или соль и применить их ко всему датасету. Теоретически, знания одного телефона может быть достаточно, если злоумышленник знает используемую соль и хеш-функцию, для точного результата нужно два значения. При уникальной соли на каждую запись - знание телефонов поможет взломать остальные записи только косвенно, то есть может помочь установить алгоритм, по которому генерируется уникальная соль. Было выяснено: при одинаковой длине самой безопасной является смешанная соль (из букв и цифр), наименее безопасной – соль, состоящая только из цифр. При увеличении длины соли безопасность растет.