**МИНОБРНАУКИ РОССИИ**

**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 «Применение соли в шифровании данных»**

**Вариант – 9**

**Студент гр. 23Б15-пу**

**Кубякин Н.А.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

***Оглавление***

[1. Цель работы 3](#_Toc178797423)

[2. Описание задачи (формализация задачи) 3](#_Toc178797424)

[3. Теоретическая часть 4](#_Toc178797431)

[4. Основные шаги программы 5](#_Toc178797432)

[5. Блок-схема программы 6](#_Toc178797433)

[6. Описание программы 7](#_Toc178797442)

[7. Рекомендации пользователя 7](#_Toc178797443)

[8. Рекомендации программиста 8](#_Toc178797444)

[9. Исходный код программы 8](#_Toc178797445)

[10. Контрольный пример 8](#_Toc178797447)

[11. Вывод 11](#_Toc178797448)

[12. Источники 12](#_Toc178797450)

# 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 состоит в расшифровке набора данных, зашифрованного с помощью хэш-функции и соли и проанализировать решение при разных функциях и солях.

# Порядок выполнения:

1. Изучить особенности шифрования телефонных номеров.
2. Расшифровать данный датасет с помощью программы Hashcat [[1]](#hashcat).
3. Зашифровать обезличенный датасет тремя разными функциями и различными вариантами солей.
4. Написать от чего меняется скорость расшифровки. Влияние вида соли, длины соли и хеш-функции на скорость расшифровки датасета. А так же ответить на вопрос – сколько нужно знать телефонов из предложенного датасета, чтоб 100% его взломать?

# Теоретическая часть

"Брутфорс" (Brute Force) - это метод, который заключается в попытке нахождения решения задачи, путем перебора возможных вариантов. В контексте криптографии это обычно пароль или ключ. Этот подход может включать в себя не только простой перебор, но и использование специализированных алгоритмов, таких как атаки словаря, для ускорения процесса взлома. Такие атаки могут быть очень ресурсоемкими и занимать продолжительное время, особенно если используются сложные или длинные пароли.

"Соль" (Salt) - это дополнительные данные, добавляемые к входным данным перед их хэшированием. В криптографии соль используется для усложнения процесса взлома паролей и предотвращения использования таблиц радужных хэшей. Каждый пароль может снабжаться своей уникальной солью, генерируемой определенным образом образом, что повышает безопасность системы. В этом случае соль называется динамической.

Хэш-функция — это математическая функция, которая принимает входные данные произвольной длины и преобразует их в фиксированную строку или числовое значение определенной длины, называемое хэш-значением или просто хэшем. Основная цель хэш-функций — создать уникальное представление входных данных, чтобы даже небольшие изменения в исходных данных приводили к значительным изменениям в хэш-значении.

MD5 (Message Digest Algorithm 5) - это криптографический хэш-алгоритм, созданный для создания хэш-сумм из данных. Он генерирует фиксированную длину хэша (128 бит), используя входные данные произвольной длины. Однако из-за обнаруженных уязвимостей к коллизиям (когда два разных входа дают одинаковый хэш) он считается устаревшим для криптографических целей.

SHA-1 (Secure Hash Algorithm 1) - также криптографический хэш-алгоритм, создающий фиксированную хэш-сумму (160 бит) из входных данных. Он также стал уязвимым к коллизиям из-за развития вычислительных методов, что делает его не рекомендуемым для криптографических задач.

SHA-256 (Secure Hash Algorithm 256) - это один из членов семейства SHA-2, который генерирует более длинную и безопасную хэш-сумму (256 бит) в сравнении с MD5 и SHA-1. SHA-256 используется широко для обеспечения безопасности данных и криптографических протоколов в современных системах благодаря своей устойчивости к коллизиям и хорошей криптографической стойкости.

# Основные шаги программы

1. Декодировать датесет с помощью Hashcat: прописать в cmd команду hashcat -m0 -a3 -o output.txt txt\_h.txt ?d?d?d?d?d?d?d?d?d?d?d,

где -m0 – это ключ для md5, -a3 ключ для атаки методом грубой силы, ?d?d?d?d?d?d?d?d?d?d?d – маска из 11 цифр, -о – ключ для файла вывода. На выходе будет получен файл output.txt, в котором сохранены расшифрованные номера в формате: <hash> : <номер>.

1. Определить соль. Зная несколько расшифрованных номеров, мы можем установить исходную соль, ведь она была добавлена к исходным номерам путем сложения номера и соли. Для всех расшифрованных номеров выполняется следующее:
2. Вычесть из расшифрованных номеров известные
3. Добавить получившееся значения в словарь
4. Вывести то значение, которое вошло в словарь 5 раз

Запустив программу main.py с полученным ранее файлом output.txt, была получена искомая соль - **56998744**.

1. Зашифровать список номеров используя одну из трёх хэш-функций и одну из различных солей с помощью encrypt.py (выбор пользователя).
2. Проанализировать разницу во времени, нужном программе Hashcat, чтобы расшифровать различные варианты.
3. Рассмотреть минимальное количество известных номеров чтобы точно расшифровать исходный датасет **(3 номера)**.

# Блок-схема программы

# На рисунке представлена блок-схема алгоритма.

# 

# Рис. 5 Блок-схема поиска соли.

# Описание программы

Программа реализована на языке Python 3.10 с использованием библиотек hashlib [[2]](#hashlib), tkinter [[3]](#tkinter).

В программе используются 2 модуля для вычисления соли и шифровки файлов.

Таблица 6.1 Описание модулей

|  |  |  |
| --- | --- | --- |
|  | Имя модуля | Назначение |
|  | main.py | Поиск соли в расшифрованном файле. |
|  | encrypt.py | Шифровка номеров по указанной хэш-функции и соли. |

# Рекомендации пользователя

1. Для расшифровки датасета нужно:
   1. Открыть командную строку (Win+R, прописать cmd)
   2. Прописать cd “путь папки где находится hashcat”
   3. Прописать hashcat -m0 -a3 -o “название файла вывода” “название файла с датасетом” ?d?d?d?d?d?d?d?d?d?d?d
   4. Дождаться окончания работы программы.
2. Для поиска соли необходимо расположить файл с расшифрованным датасетом в папку с main.py и переименовать его в “decrypted.txt”, после чего запустить программу. Соль будет выведена в консоль.
3. Для шифровки датасета необходимо запустить encrypt.py, выбрать в интерфейсе вариант хэш-функции и вариант соли, выбрать файл с датасетом и запустить программу.
4. Для расшифровки надо повторить шаг 1, но вместо –m0 вписать –m100 для SHA1 и –m1400 для SHA256 и ?d для чисел, ?l для букв, ?a для обоих типов столько раз, сколько всего символов в каждом пункте (11 из номера + соль, без пробелов)

# Рекомендации программиста

Для запуска программы необходима 64-битная операционная система Windows, Linux или macOS. Для работы с кодом необходима среда разработки, совместимая с python 3.1 и библиотеки hashlib [[2]](#hashlib), tkinter [[3]](#tkinter).

Для запуска программы необходимо установить Python версии не ниже 3.10, а так же библиотеки hashlib [[2]](#hashlib), tkinter [[3]](#tkinter).

Папка с hashcat должна находиться в директории с кодом.

# Исходный код программы

# Исходный код программы и необходимые текстовые файлы доступны по ссылке: <https://github.com/NikiTaku1/spbu_alg/tree/main/lab3>

# Контрольный пример

В данном разделе представлен пример, демонстрирующий работу программы.

1) Запуск hashcat для расшифровки исходного датасета.

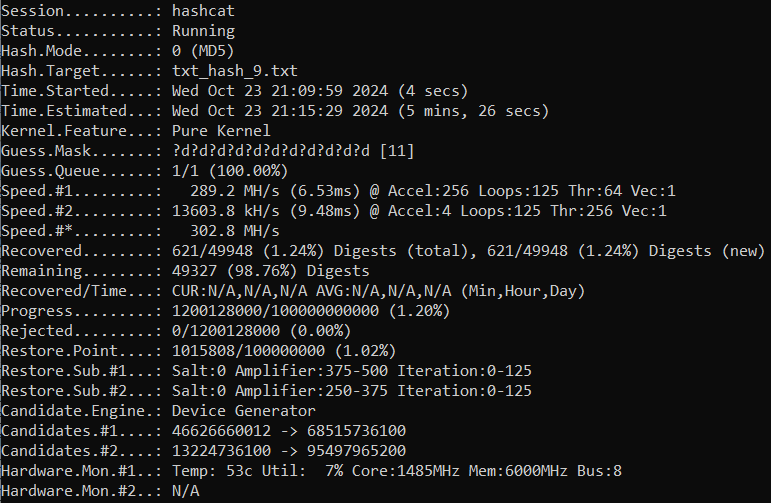


Рис 10.1 Работа hashcat по расшифровке исходного датасета.

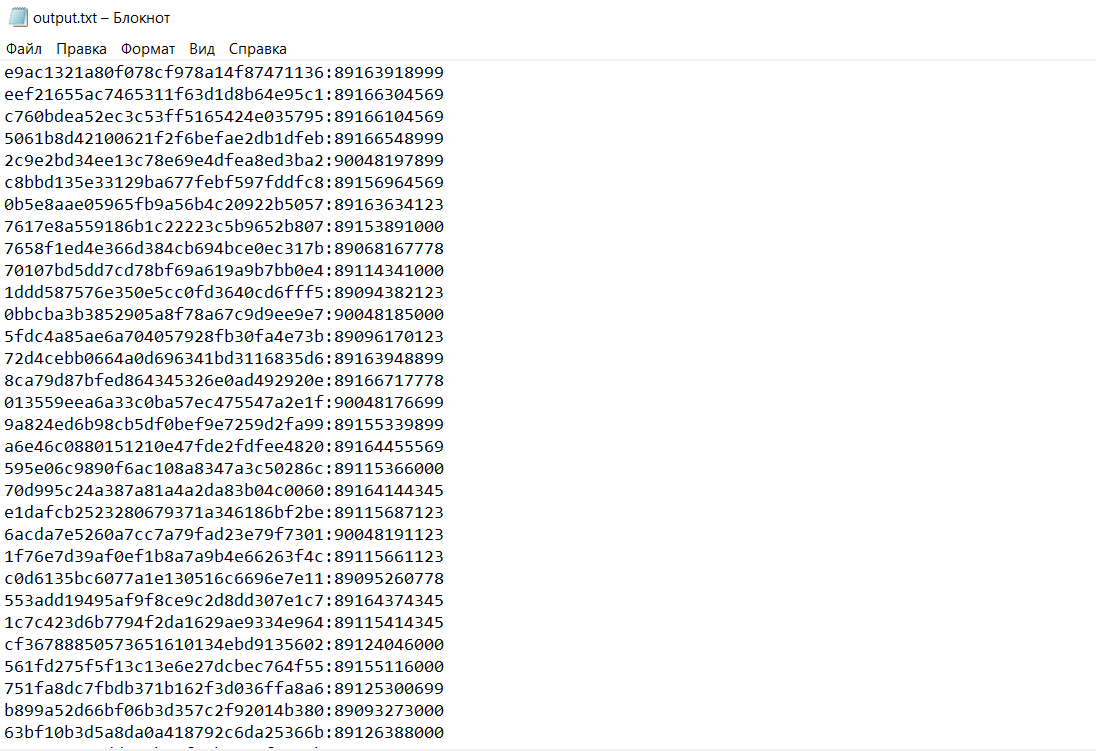


Рис 10.2 Результат расшифровки

2) Запуск main.py для поиска соли и создания файла с полностью расшифрованным датасетом.

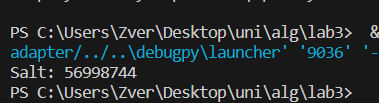


Рис 10.3 Найденная соль.

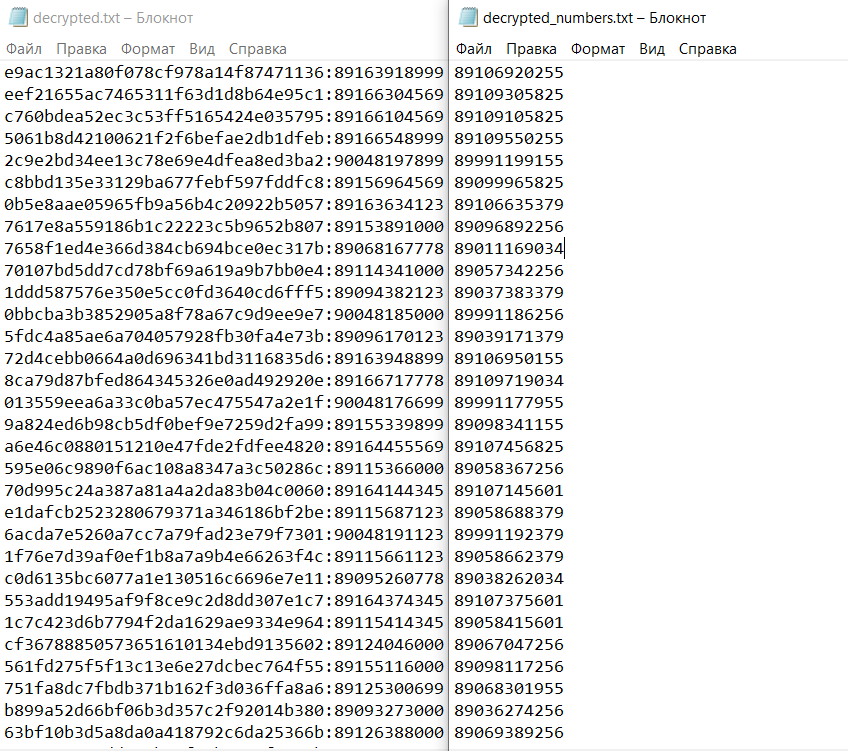


Рис 10.4 Сравнение файла – результата hashcat и полностью расшифрованного датасета.

3) Создание зашифрованных файлов с помощью encrypt.py

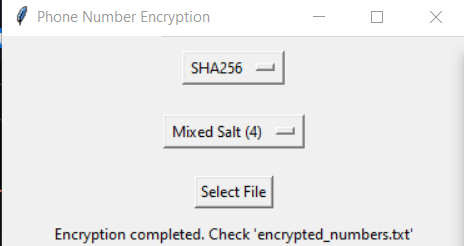


Рис 10.5 Работа encrypt.py

4) Сравнение времени работы Hashcat со всеми комбинациями

Таблица 10.6 Время работы Hashcat

|  |  |  |  |
| --- | --- | --- | --- |
|  | MD5 | SHA1 | SHA256 |
| Нет соли | 5м 33с | 6м 45с | 7м 39с |
| Числовая (3) | 4ч 43м | 8ч 48м | 18ч 23м |
| Числовая (4) | 1д 23ч | 3д 15ч | 7д 14ч |
| Буквенная (3) | 3д 11ч | 6д 7ч | 13д 14ч |
| Буквенная (4) | 91д 1ч | 165д 18ч | 361д 10ч |
| Смешанная (3) | 181д 3ч | 313д 19ч | 1г 323д |
| Смешанная (4) | 12л 50д | 23л 38д | 52л 176д |

# Вывод

В результате выполнения данной работы был успешно расшифрован исходный дата-сет и была найдена его соль. Также был проведен сравнительный анализ других алгоритмов шифрования на их устойчивость к взлому с применением различных типов соли. Лучше всего себя показал алгоритм шифрования SHA256. Среди всех представленных алгоритмов у него наибольшая длина хэша - 256bit против 160 и 128 у sha1 и md5 соответственно. Использование смешанных солей является гораздо более эффективным, по сравнению с буквенными солями, а численные соли еще менее эффективны, чем буквенные. Это объясняется тем, что при для буквенной соли количество вариантов перебора при брутфорсе значительно возрастает, а при смешанной, совмещает варианты и числовых и буквенных солей. Так же увеличение длины соли даже на 1 символ или разряд существенно замедляет время перебора.

# Источники

# hashcat — hashcat documentation // Documentation URL: <https://hashcat.net/wiki/doku.php?do=export_xhtml&id=hashcat> (дата обращения: 23.10.2024).

# hashlib — hashlib documentation // Documentation URL: <https://docs.python.org/3/library/hashlib.html> (дата обращения: 23.10.2024).

# tkinter — Python interface to Tcl/Tk // Documentation URL: <https://docs.python.org/3/library/tkinter.html> (дата обращения: 23.10.2024).