Санкт-Петербургский государственный университет

Факультет прикладной математики – процессов управления

**Отчет**

по лабораторной работе №3

по дисциплине «Алгоритмы и структуры данных»

на тему «Применение соли в шифровании данных»

Вариант 6

Автор работы: Добренкова Л.С.

Группа: 22.Б15-пу

Преподаватель: Дик А.Г.

Санкт-Петербург, 2023

**Оглавление**

[**1.** **Введение** 3](#_Toc154441659)

[**2.** **Цель работы** 3](#_Toc154441660)

[**3.** **Задачи** 3](#_Toc154441661)

[**4.** **Теоретическая часть** 3](#_Toc154441662)

[**5.** **Алгоритм выполнения** 4](#_Toc154441663)

[**6.** **Влияние других типов соли** 6](#_Toc154441664)

[**7.** **Вывод** 7](#_Toc154441665)

# **Введение**

Алгоритмы шифрования играют ключевую роль в современном мире безопасности данных. Они обеспечивают защиту информации путем преобразования ее в непонятный для посторонних вид, что делает доступ к ней возможным только с помощью специальных ключей или методов. Эти алгоритмы являются фундаментальной основой для безопасной передачи, хранения и обработки конфиденциальной информации.

# **Цель работы**

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

# **Задачи**

1. Исследовать алгоритмы шифрования.
2. Определить тип шифрования в данном дата-сете.
3. Расшифровать дата-сет, используя ПО Hashcat.
4. Написать программу для определения соли.
5. Изучить влияние разных типов соли на скорость дешифрации данных.

# **Теоретическая часть**

"Брутфорс" (Brute Force) - это метод атаки, который заключается в попытке нахождения правильного значения, обычно пароля или ключа, путем систематического перебора всех возможных комбинаций. Этот подход не использует какие-либо специфические уязвимости или предположения о пароле, а просто перебирает все варианты до тех пор, пока не будет найдено правильное значение. Такие атаки могут быть очень ресурсоемкими и занимать продолжительное время, особенно если используются сложные или длинные пароли.

"Соль" (Salt) - это случайные данные, которые добавляются к входным данным перед их хэшированием. В криптографии "соль" используется для усложнения процесса хэширования и предотвращения использования таблиц радужных хэшей для взлома паролей. Каждый пользователь имеет свою уникальную соль, которая добавляется к их паролю перед хэшированием. Это увеличивает уровень безопасности, поскольку даже для одинаковых паролей хэши будут отличаться из-за уникальной соли.

Хэш-функция — это математическая функция, которая принимает входные данные произвольной длины и преобразует их в фиксированную строку определенной длины, называемую 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.

# **Алгоритм выполнения**

В первую очередь требуется декодировать дата-сет с зашифрованными номера, где известно 5 случайных номер. Для этого используем следующую 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 получаем файл output.txt, в котором сохранены расшифрованные номера в формате: <hash> : <номер>.

Далее нужно установить соль, для этого я написала программу на python, блок-схема 5.1. Зная несколько расшифрованных номеров, мы можем установить исходную соль, ведь она была добавлена к исходным номерам путем сложения номера и соли. Для это для всех расшифрованных номеров произведем следующее:

1. Вычитаем из расшифрованных номеров известные
2. Добавляем получившееся значения в словарь
3. Выводим то значение, которое вошло в словарь 5 раз

Запустив это программу с полученным ранее файлом output.txt получим 34000319– это и есть искомая соль.

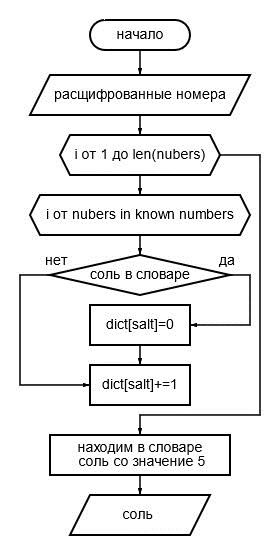


Рисунок 5.1

# **Влияние других типов соли**

В данном примере приведена зависимость времени от различных алгоритмов шифрования (md5, sha256, sha1) и соли (no salt, alphabetic, numeric). Атака производится методом «грубой силы» по маске из 11 цифр.

Таблица 4.1 No salt

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | 2м 11с |
| SHA1 | 3м 28с |
| SHA256 | 6м 3с |

# 

Таблица 4.2 Numeric Salt

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | 8м 31с |
| SHA1 | 12м 37с |
| SHA256 | 22м 46с |

Таблица 4.3 Alphabetic Salt

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | ~300ч |
| SHA1 | ~479ч |
| SHA256 | ~2000ч |

В таблице 4.1 мы видим, что сравнительно небольшое время расшифровки, с учетом того, что расшифровка происходила на встроенной графике процессора, а не на дискретной видеокарте.

В таблице 4.2 время расшифровки возрастает, ведь мы добавляем арифметическую соль из 3 цифр. Подобное затрудняет атаку грубой силы, но она все равно остается возможной.

В таблице 4.3 невозможно точно определить время расшифровки, ибо добавление даже сравнительно небольшой буквенной соли значительно усложняет процесс.

Лучше всего себя показал алгоритм шифрования SHA256, что не удивительно. Среди всех представленных алгоритмов у него наибольшая длина ключа, 256bit против 160 и 128 у sha1 и md5 соответственно. Использование буквенной соли является наиболее надежным, нежели арифметической. Это объясняется тем, что при для буквенной соли количество вариантов перебора при брутфорсе значительно возрастает.

# **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.