**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Кафедра фундаментальной информатики и информационных технологий**

**ОТЧЕТ**

**по лабораторной работе №3**

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

**на тему «Применение соли в шифровании данных»**

**Вариант – 4**

|  |  |  |
| --- | --- | --- |
| Студент гр. 22Б15-пу |  | Добренкова Л.С. |
| Преподаватель |  | Дик А.Г. |

**Санкт-Петербург**

**2023 г**

Оглавление

[1. Цель работы 3](#_heading=h.30j0zll)

[2. Задачи 3](#_heading=h.1fob9te)

[3. Теоретическая часть 3](#_heading=h.3znysh7)

[4. Алгоритм метода 4](#_heading=h.2et92p0)

[5. Контрольный пример 5](#_heading=h.tyjcwt)

[6. Анализ результатов работы алгоритма и вводных условий 8](#_heading=h.3dy6vkm)

[7. Вывод 8](#_heading=h.1t3h5sf)

# 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, 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.

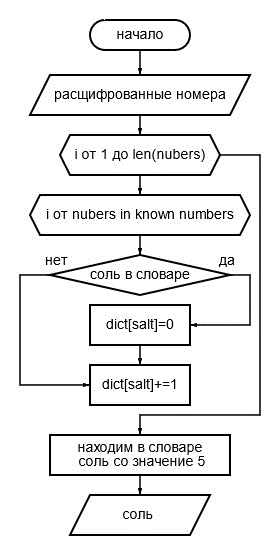
# Алгоритм метода

В первую очередь требуется декодировать дата-сет с зашифрованными номерами, где известно 5 случайных номеров. Для этого можно использовать утилиту hashcat, а именно 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 будет получен файл output.txt, в котором сохранены расшифрованные номера в формате: <hash> : <номер>.

Далее нужно 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:

1. Вычесть из расшифрованных номеров известные
2. Добавить получившееся значения в словарь
3. Вывести то значение, которое вошло в словарь 5 раз

Запустив это программу с полученным ранее файлом output.txt, была получена искомая соль - 32941766.



*Рис. 5.1 Блок-схема поиска соли*

# Контрольный пример

В данном примере приведена зависимость времени от различных алгоритмов шифрования (md5, sha256, sha1), типа соли (no salt, alphabetic, numeric), а также длины соли (3-5 символов или разрядов). 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 (3-значное число)*

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

*Таблица 4.3 Numeric Salt (4-значное число)*

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | 13м 27с |
| SHA1 | 19м 21с |
| SHA256 | 46м 1c |

*Таблица 4.4 Numeric Salt (5-значное число)*

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | 18м 55с |
| SHA1 | 37м 20с |
| SHA256 | 1ч 37м |

*Таблица 4.5 Alphabetic Salt (3-символьная строка)*

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | 1ч 47м |
| SHA1 | 2ч 24м |
| SHA256 | ~4ч |

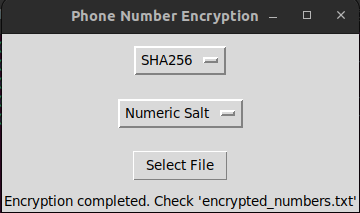
*Таблица 4.6 Alphabetic Salt (4-символьная строка)*

|  |  |
| --- | --- |
| Алгоритм | Время |
| MD5 | ~8ч |
| SHA1 | ~12ч |
| SHA256 | ~29ч |

В таблице 4.1 можно увидеть сравнительно небольшое время расшифровки, с учетом того, что вычисления проводились на cpu, а не на дискретной видеокарте.

В таблице 4.2 время расшифровки возрастает, из-за добавления числовой соли из 3 цифр. Подобное затрудняет атаку грубой силы, но она все равно остается возможной. Сравнивая с солью из 4 и 5 цифр в таблицах 4.3 и 4.4 соответственно, можно увидеть экспоненциальный рост времени вычисления в зависимости от длины.

В таблицах 4.5 и 4.6 невозможно точно определить время расшифровки, так как добавление даже сравнительно небольшой буквенной соли значительно усложняет процесс. Однако некоторые измерения все же возможны и на их основе получаем снова существенное увеличение времени вычислений от длины соли.

*Рис. 5.1 Пример работы enctypt.py*

# Анализ результатов работы алгоритма и вводных условий

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 буквенной соли является наиболее надежным, нежели арифметической. Это объясняется тем, что при для буквенной соли количество вариантов перебора при брутфорсе значительно возрастает. Так же увеличение длины соли даже на 1 символ или разряд существенно замедляет время перебора.

# 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.