

САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики-процессов управления

Программа бакалавриата

“Большие данные и распределенная цифровая платформа”

ОТЧЕТ

по лабораторной работе №2

по дисциплине «Алгоритмы и структуры данных»

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

Вариант – 2

**Студент гр. 23Б15-пу
Антонян А. А.**

**Преподаватель
Дик А.Г.**

**Санкт-Петербург
2024 г.**

Оглавление

1. Цель работы	3
2. Теоретическая часть	3
3. Описание задачи	5
4. Основные шаги программы	5
5. Описание схемы пошагового выполнения алгоритма	6
6. Описание программы	8
7. Исходный код программы:.....	9
8. Рекомендации для пользователя	9
9. Рекомендации для программиста.....	10
10. Контрольный пример	10
11. Анализ	13
12. Вывод.....	13
13. Источники	13

Цель работы

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

Теоретическая часть

Хеш-функция

Хеш-функция — это алгоритм, который преобразует данные произвольного размера в уникальный код фиксированной длины, называемый хешем или «отпечатком». Этот хеш можно считать уникальным идентификатором исходных данных. Основные свойства хеш-функций включают:

1. **Односторонность:** создание хеша из данных легко, однако восстановить оригинал из хеша практически невозможно.
2. **Фиксированная длина:** длина хеша всегда постоянна.
3. **Уникальность:** небольшое изменение в исходных данных вызывает значительные изменения в значении хеша.
4. **Детерминированность:** одинаковые входные данные всегда дают один и тот же хеш.

Без дополнительных мер, таких как использование соли, хеши уязвимы к атакам, основанным на методе перебора (Brute Force) и радужных таблицах.

Техника Brute Force для взлома

Brute Force — это метод, при котором программа перебирает все возможные комбинации символов, пока не найдет значение, которое соответствует заданному хешу. Этот метод не требует знаний о содержимом данных, но из-

за необходимости проверки большого количества комбинаций он крайне медленный для длинных или сложных паролей.

Для повышения скорости brute force используют оптимизированные стратегии:

- **Атака по маске:** если известна структура пароля (например, только цифры), можно ограничить диапазон возможных комбинаций.
- **Словарная атака:** программа проверяет значения из заранее подготовленного списка (словаря) вероятных паролей вместо полного перебора всех комбинаций.

Основной недостаток метода brute force — высокая временная сложность, которая увеличивается с ростом числа символов.

Соль

Соль (или salt) — это случайное значение, добавляемое к данным перед их хешированием, что усложняет взлом методом brute force. Без соли одинаковые данные всегда приводят к одинаковым хешам, что делает их уязвимыми для атак. Благодаря добавлению соли одинаковые значения дают уникальные хеши.

Принцип работы соли:

1. При хешировании к данным (например, паролю) добавляется случайное значение, называемое солью.
2. Соль вместе с данными хешируется, и полученный хеш сохраняется.
3. При проверке пароля извлекается соль и добавляется к введенным данным перед хешированием, чтобы сверить результат с сохраненным значением.

Описание задачи

Задача состоит в расшифровке датасета с номерами телефонов, с последующим анализом аналогичной задачи с различными входными данными:

- 1) Программа должна считывать входной файл (scoring_data_v.1.2.xlsx) с хешами и расшифрованными номерами телефонов без соли и разбить его на два файла hashes.txt и phones.txt.
- 2) Расшифровка датасета с последующим нахождением соли.
- 3) Зашифровать датасет с номерами функциями разных семейств, и протестировать расшифровку с разными солями.
- 4) Найти минимальное количество телефонов для нахождения соли.

Основные шаги программы

1. **Запуск программы:** Пользователь запускает скрипт для считывания входного файла, и загружаются два файла hashes.txt и phones.txt.
2. **Расшифровка хэшей:** С помощью утилиты hashcat[\[1\]](#) расшифровать хэши и сохранить их в файл cracked.txt.
3. **Нахождение соли:** С помощью python скрипта находится соль и сохраняется файл с датасетом без соли.
4. **Тестирование задачи:** Задача тестируется с различными солями и хеш-функциями.

Описание схемы пошагового выполнения алгоритма

- 1) Загрузка зашифрованного датасета
- 2) Применение алгоритмов для расшифровки данных с использованием метода Brute force
- 3) Тестирование программы с помощью хеш-функции MD5, вследствие чего получаем телефонные номера с солью
- 4) С помощью скрипта на python находим соль для этих номеров
- 5) Далее, используя скрипт на python зашифровываем номера через SHA1, SHA3-256
- 6) Применение алгоритмов для расшифровки данных с использованием метода Brute force

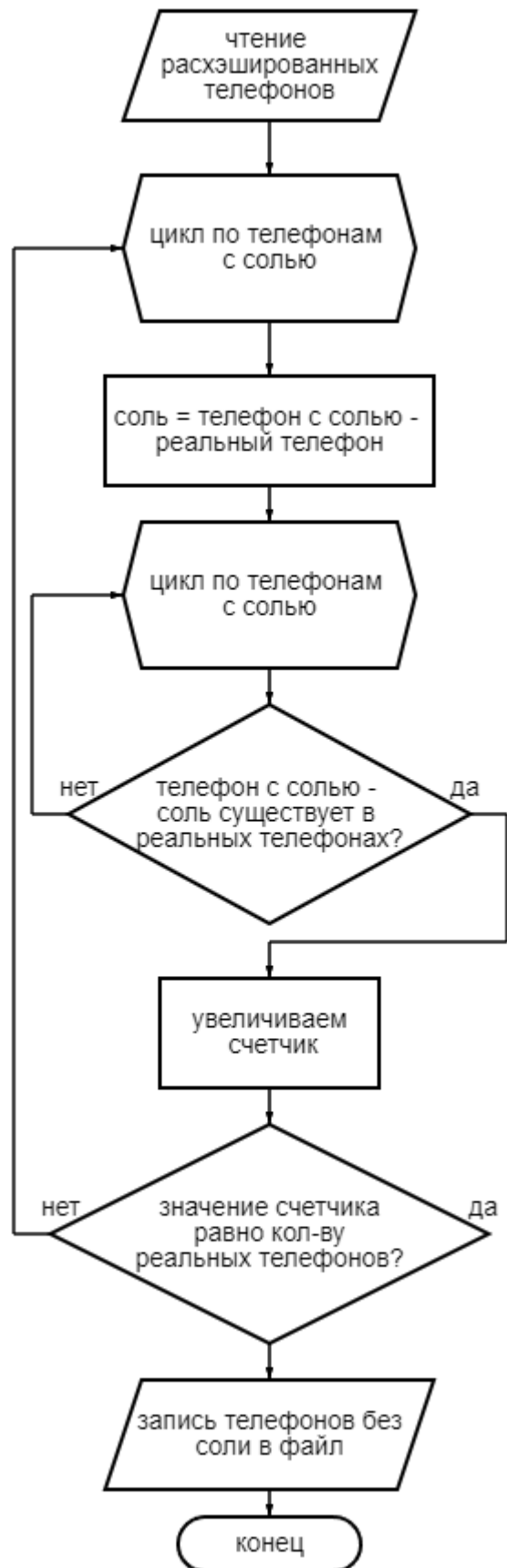


Рис 1. Блок-схема нахождения соли

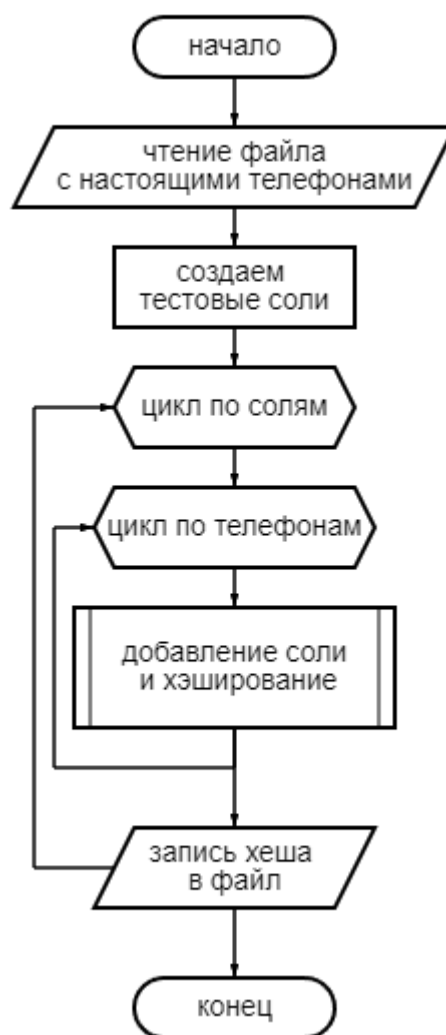


Рис 2. Блок-схема хеширования телефонов

Описание программы

Программная реализация написана на языке Python 3.12.2 с использованием библиотеки [hashlib](#)[\[2\]](#) и утилиты [hashcat](#)[\[1\]](#). Задача выполняется посредством последовательного выполнения скриптов и команд [hashcat](#)[\[1\]](#).

Таблица1 Скрипты

Скрипт	Описание	Возвращает
script.py	Принимает входной .xlsx файл и разбивает его на два файла .txt	phones.txt, hashes.txt
find_salt.py	Находит соль для расшифрованного файла с помощью известных номеров телефона и записывает сырые номера телефонов в файл.	Соль, real_phones.txt
hashing.py	Хеширует телефоны различными функциями с различными солями	<hash>_hashes_salt_<salt>.txt

Исходный код программы:

<https://github.com/ArseniiAntonin/spbu-algorithms-and-data-structures>

Рекомендации для пользователя

1. Для запуска программы убедитесь, что у вас установлен Python наиболее актуальной версии. Код можно запустить в среде разработки или через командную строку.
2. Запускайте скрипты только в очередности, указанной в алгоритме выполнения задания.

Рекомендации для программиста

1. Использование соли при хешировании: Соль должна быть безопасной и уникальной для каждого хешируемого значения.
2. Для поддержания актуальности и работоспособности программы используйте последние версии python и hashcat. Применяйте практики надлежащего именования переменных и функций для улучшения читаемости кода.

Контрольный пример

1. Считывание входного файла

Для считывания входного файла запустите скрипт script.py.

2. Расшифровка

Расшифровка хешей утилитой hashcat[\[1\]](#) с помощью атаки по маске (Рис. 3 и 4).

```
C:\Users\zvn\ln\Downloads\hashcat-6.2.6\hashcat-6.2.6>hashcat -m 0 -a 3 C:\Users\zvn\ln\IT\algorithms_and_data_structures\assignment3\hashes.txt 89?d?d?d?d?d?d?d?d -o C:\Users\zvn\ln\IT\algorithms_and_data_structures\assignment3\cracked.txt
```

Рисунок 3. Пример команды для расшифровки хэшей MD5

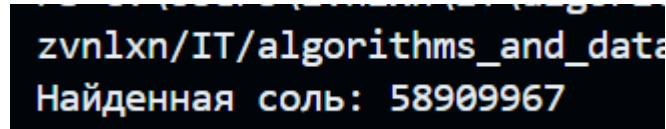
```
Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 0 (MD5)
Hash.Target.....: C:\Users\zvn\ln\IT\algorithms_and_data_structures\assignment3\hashes.txt
Time.Started.....: Wed Oct 23 22:04:59 2024 (6 mins, 1 sec)
Time.Estimated...: Wed Oct 23 22:11:00 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: 89?d?d?d?d?d?d?d [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2772.9 kH/s (9.24ms) @ Accel:128 Loops:100 Thr:64 Vec:1
Recovered.....: 48870/49940 (97.86%) Digests (total), 48870/49940 (97.86%) Digests (new)
Remaining.....: 1070 (2.14%) Digests
Recovered/Time...: CUR:8281,N/A,N/A AVG:8215.49,N/A,N/A (Min,Hour,Day)
Progress.....: 1000000000/1000000000 (100.00%)
Rejected.....: 0/1000000000 (0.00%)
Restore.Point...: 10000000/10000000 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-100 Iteration:0-100
Candidate.Engine.: Device Generator
Candidates.#1...: 89123184973 -> 89396497383
Hardware.Mon.#1..: Util: 19% Core: 400MHz Mem:1600MHz Bus:16

Started: Wed Oct 23 22:04:51 2024
Stopped: Wed Oct 23 22:11:02 2024
```

Рисунок 4. Результат расшифровки входного датасета

3. Нахождение соли

Запуск скрипта find_salt.py, который находит соль и записывает номера телефонов без соли в отдельный файл real_phones.txt.

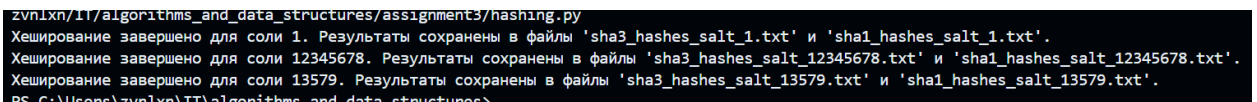


```
zvn1xn/IT/algorithms_and_data
Найденная соль: 58909967
```

Рисунок 5. Соль, найденная скриптом

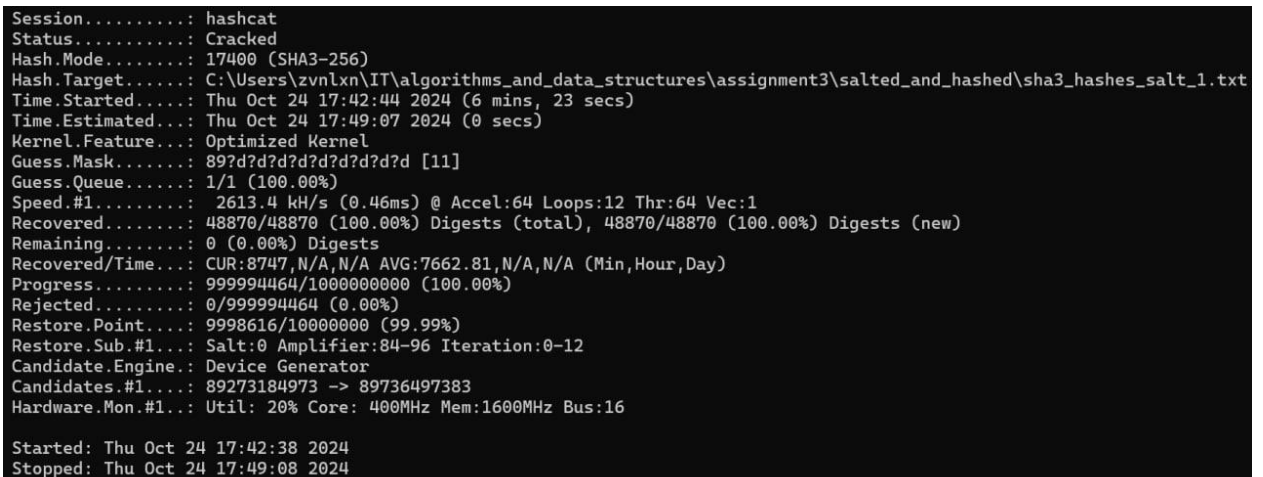
4. Тестирование задачи с различными солями и хеш-функциями

Запуска скрипта hashing.py, который хэширует исходный датасет с различными солями различными хеш-функциями и сохраняет зашифрованные номера телефонов в отдельные файлы. Далее с помощью утилиты hashcat[1] происходит расшифровка хешей. (Рис. 6 – 10)



```
zvn1xn/IT/algorithms_and_data_structures/assignment3/hashing.py
Хеширование завершено для соли 1. Результаты сохранены в файлы 'sha3_hashes_salt_1.txt' и 'sha1_hashes_salt_1.txt'.
Хеширование завершено для соли 12345678. Результаты сохранены в файлы 'sha3_hashes_salt_12345678.txt' и 'sha1_hashes_salt_12345678.txt'.
Хеширование завершено для соли 13579. Результаты сохранены в файлы 'sha3_hashes_salt_13579.txt' и 'sha1_hashes_salt_13579.txt'.
```

Рисунок 6. Вывод скрипта hashing.py



```
Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 17400 (SHA3-256)
Hash.Target.....: C:\Users\zvn1xn\IT\algorithms_and_data_structures\assignment3\salted_and_hashed\sha3_hashes_salt_1.txt
Time.Started.....: Thu Oct 24 17:42:44 2024 (6 mins, 23 secs)
Time.Estimated...: Thu Oct 24 17:49:07 2024 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Mask.....: 89?d?d?d?d?d?d?d [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2613.4 kH/s (0.46ms) @ Accel:64 Loops:12 Thr:64 Vec:1
Recovered.....: 48870/48870 (100.00%) Digests (total), 48870/48870 (100.00%) Digests (new)
Remaining.....: 0 (0.00%) Digests
Recovered/Time...: CUR:8747,N/A,N/A AVG:7662.81,N/A,N/A (Min,Hour,Day)
Progress.....: 999994464/1000000000 (100.00%)
Rejected.....: 0/999994464 (0.00%)
Restore.Point...: 9998616/100000000 (99.99%)
Restore.Sub.#1...: Salt:0 Amplifier:84-96 Iteration:0-12
Candidate.Engine.: Device Generator
Candidates.#1...: 89273184973 -> 89736497383
Hardware.Mon.#1..: Util: 20% Core: 400MHz Mem:1600MHz Bus:16

Started: Thu Oct 24 17:42:38 2024
Stopped: Thu Oct 24 17:49:08 2024
```

Рисунок 7. Результат расшифровки датасета с хеш-функцией sha3-256 и солью 1

```

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 17400 (SHA3-256)
Hash.Target.....: C:\Users\zvn\xn\IT\algorithms_and_data_structures\assignment3\salted_and_hashed\sha3_hashes_salt_13579.txt
Time.Started.....: Thu Oct 24 21:15:07 2024 (6 mins, 36 secs)
Time.Estimated...: Thu Oct 24 21:21:43 2024 (0 secs)
Kernel.Feature...: Optimized Kernel
Guess.Mask.....: 89?d?d?d?d?d?d?d [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2521.9 kH/s (0.69ms) @ Accel:64 Loops:12 Thr:64 Vec:1
Recovered.....: 48870/48870 (100.00%) Digests (total), 48811/48870 (99.88%) Digests (new)
Remaining.....: 0 (0.00%) Digests
Recovered/Time...: CUR:8018,N/A,N/A AVG:7385.23,N/A,N/A (Min,Hour,Day)
Progress.....: 999977856/1000000000 (100.00%)
Rejected.....: 0/999977856 (0.00%)
Restore.Point...: 9998616/100000000 (99.99%)
Restore.Sub.#1...: Salt:0 Amplifier:72-84 Iteration:0-12
Candidate.Engine.: Device Generator
Candidates.#1...: 89963184973 -> 89846497383
Hardware.Mon.#1..: Util: 23% Core: 400MHz Mem:1600MHz Bus:16

Started: Thu Oct 24 21:14:57 2024
Stopped: Thu Oct 24 21:21:45 2024

```

Рисунок 8. Результат расшифровки датасета с хеш-функцией sha3-256 и солью 13579

```

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 100 (SHA1)
Hash.Target.....: C:\Users\zvn\xn\IT\algorithms_and_data_structures\assignment3\salted_and_hashed\sha1_hashes_salt_1.txt
Time.Started.....: Thu Oct 24 22:30:43 2024 (3 mins, 25 secs)
Time.Estimated...: Thu Oct 24 22:34:08 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: 89?d?d?d?d?d?d?d [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 4882.4 kH/s (5.37ms) @ Accel:4 Loops:100 Thr:256 Vec:1
Recovered.....: 48870/48870 (100.00%) Digests (total), 25753/48870 (52.70%) Digests (new)
Remaining.....: 0 (0.00%) Digests
Recovered/Time...: CUR:8192,N/A,N/A AVG:7542.73,N/A,N/A (Min,Hour,Day)
Progress.....: 1000000000/1000000000 (100.00%)
Rejected.....: 0/1000000000 (0.00%)
Restore.Point...: 9999360/100000000 (99.99%)
Restore.Sub.#1...: Salt:0 Amplifier:0-100 Iteration:0-100
Candidate.Engine.: Device Generator
Candidates.#1...: 89121517383 -> 89396497383
Hardware.Mon.#1..: Util: 9% Core: 400MHz Mem:1600MHz Bus:16

Started: Thu Oct 24 22:30:39 2024
Stopped: Thu Oct 24 22:34:10 2024

```

Рисунок 9. Результат расшифровки датасета с хеш-функцией sha1 и солью 1

```

Session.....: hashcat
Status.....: Cracked
Hash.Mode.....: 100 (SHA1)
Hash.Target.....: C:\Users\zvn\xn\IT\algorithms_and_data_structures\assignment3\salted_and_hashed\sha1_hashes_salt_12345678.txt
Time.Started.....: Thu Oct 24 22:35:50 2024 (6 mins, 5 secs)
Time.Estimated...: Thu Oct 24 22:41:55 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Mask.....: 89?d?d?d?d?d?d?d [11]
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 2736.6 kH/s (5.73ms) @ Accel:8 Loops:100 Thr:128 Vec:1
Recovered.....: 48870/48870 (100.00%) Digests (total), 48866/48870 (99.99%) Digests (new)
Remaining.....: 0 (0.00%) Digests
Recovered/Time...: CUR:8326,N/A,N/A AVG:8023.16,N/A,N/A (Min,Hour,Day)
Progress.....: 1000000000/1000000000 (100.00%)
Rejected.....: 0/1000000000 (0.00%)
Restore.Point...: 9999360/100000000 (99.99%)
Restore.Sub.#1...: Salt:0 Amplifier:0-100 Iteration:0-100
Candidate.Engine.: Device Generator
Candidates.#1...: 89121517383 -> 89396497383
Hardware.Mon.#1..: Util: 9% Core: 400MHz Mem:1600MHz Bus:16

Started: Thu Oct 24 22:35:42 2024
Stopped: Thu Oct 24 22:41:57 2024

```

Рисунок 10. Результат расшифровки датасета с хеш-функцией sha1 и солью 12345678

Анализ

Время работы расшифровки хешей утилитой hashcat[1] зависит напрямую от вида функции, но в контексте задачи время работы не зависит от вида и длины соли. Минимальное количество телефонов для нахождения соли – 3.

Вывод

В рамках данной работы был разработан алгоритм для расшифровки датасета с номерами телефонов. Реализованный алгоритм обеспечивает возможность анализа выполнения задачи на различных данных, что помогает лучше понять работу хеш-функций и метода brute-force.

Источники

1. hashcat's documentation // hashcat URL: <https://hashcat.net/wiki/> (дата обращения: 20.10.2024).
2. Python hashlib documentation // hashlib URL: <https://docs.python.org/3/library/hashlib.html> (дата обращения: 20.10.2024).