**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 – 10**

**Студент гр. 23Б16-пу**

**Кривошеин А.Е.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

**Содержание**

[1. Цель работы 4](#_1fob9te)

[2. Описание задачи (формализация задачи) 4](#_3znysh7)

[3. Теоретическая часть 5](#_2et92p0)

[4. Основные шаги программы 7](#_26in1rg)

[5. Описание программы 9](#_lnxbz9)

[6. Рекомендации пользователя 1](#_35nkun2)0

[7. Рекомендации программиста 1](#_44sinio)1

[8. Контрольный пример 1](#_z337ya)2

[9. Анализ результатов работы](#_2bn6wsx) 17

[10. Вывод 1](#_2p2csry)8

# **Цель работы**

Целью данной лабораторной работы является исследование хеш-функций с учетом различных входных параметров, в том числе использование соли для шифрования данных. В рамках работы планируется дешифровка набора данных, зашифрованного с использованием хеш-функций, и анализ факторов, влияющих на скорость этого процесса.

# **Описание задачи (формализация задачи)**

1. Изучение особенностей шифрования телефонных номеров**:**  
   • Изучить способы шифрования номеров телефонов, используя хеш-функции и добавление уникального значения (соли) для повышения безопасности данных.  
   • Ознакомиться с принципами работы хеш-функций и их применением в контексте шифрования телефонных номеров.
2. Разработка программы для деобезличивания датасета**:**  
   • Написать программу, которая будет использовать хеш-функции для шифрования номеров телефонов с различной солью и изучить, как это влияет на скорость расшифровки.
3. Тестирование на других хеш-функциях**:**  
   • Программа должна быть протестирована на минимум двух различных хеш-функциях, представляющих разные семейства (например, MD-5, SHA-1, SHA-256), чтобы сравнить эффективность и скорость расшифровки.
4. Анализ влияния различных факторов на скорость расшифровки**:**  
   • Изучить влияние различных видов солей (числовых, буквенных, комбинированных), длины и выбора хеш-функции на скорость расшифровки данных.

• Определить минимальное количество телефонных номеров из датасета, необходимых для успешного расшифрования.

# **Теоретическая часть**

### **Хеш-функции**

Хеш-функция — это математический алгоритм, который принимает на вход данные произвольной длины (сообщение) и возвращает фиксированной длины строку битов, называемую хеш-значением или дайджестом. Хеш-функции широко используются в криптографии, базах данных, системах контроля версий и других областях для обеспечения целостности данных, аутентификации и хранения паролей.

### Свойства хеш-функций

1. Детерминированность: Для одного и того же входа хеш-функция всегда возвращает одинаковый хеш. Это свойство позволяет использовать хеш-функции для проверки целостности данных.
2. Быстрота вычисления: Хеш-функция должна быть достаточно быстрой для вычисления, чтобы не замедлять работу системы. Однако скорость вычисления не должна быть слишком высокой, чтобы не сделать возможным быстрый перебор всех возможных входных данных (атака Brute Force).
3. Необратимость: Из хеш-значения невозможно восстановить исходные данные. Это свойство делает хеш-функции полезными для хранения паролей, так как даже если злоумышленник получит доступ к хешам, он не сможет восстановить исходные пароли.
4. Устойчивость к коллизиям: Коллизия возникает, когда две разные входные строки дают одинаковый хеш. Хорошая хеш-функция должна минимизировать вероятность возникновения коллизий. Однако, согласно принципу Дирихле, коллизии неизбежны для любой хеш-функции с конечным выходным пространством.

В данном коде используются следующие хеш-функции:

1. SHA-1: Хеш-функция, которая создает 160-битный хеш. Хотя SHA-1 считается устаревшим и уязвимым к коллизиям, он все еще используется в некоторых приложениях.
2. SHA-256: Часть семейства SHA-2, которая создает 256-битный хеш. SHA-256 считается более безопасным и широко используется в современной криптографии.
3. MD5: Хеш-функция, которая создает 128-битный хеш. Хотя MD5 когда-то широко использовался для защиты данных и проверки целостности, в настоящее время он считается устаревшим. Тем не менее, MD5 все еще используется в некоторых приложениях, где не требуется высокая криптографическая стойкость, например, для быстрой проверки данных и контроля целостности файлов.

### **Техника взлома Brute Force**

### Brute Force (грубая сила) — это метод атаки, при котором перебираются все возможные варианты паролей или ключей до тех пор, пока не будет найден правильный. Этот метод требует больших вычислительных ресурсов и времени, но он гарантирует нахождение правильного ответа, если у атакующего достаточно ресурсов.

### В контексте хеширования, Brute Force используется для восстановления исходных данных из хешей. Атакующий генерирует все возможные варианты входных данных, хеширует их и сравнивает полученные хеши с хешами, которые он пытается взломать. Если хеши совпадают, атакующий находит исходные данные.

### **Понятие "Соли"**

Соль (Salt) — это случайное значение, которое добавляется к исходным данным перед хешированием. Соль делает хеширование более безопасным, так как она предотвращает использование предварительно вычисленных таблиц хешей для атак.

В данном коде соль вычисляется на основе телефонных номеров и используется для проверки соответствия хешей.

### **Хеширование телефонных номеров**

Хеширование телефонных номеров — это процесс преобразования телефонных номеров в хеши, которые могут быть использованы для аутентификации, безопасного хранения и передачи данных. Хеширование телефонных номеров позволяет защитить личную информацию пользователей, так как хеши не могут быть обратно преобразованы в исходные номера.

В данном коде телефонные номера хешируются с использованием алгоритмов MD-5, SHA-1 и SHA-256, а затем сохраняются в файлы для дальнейшего использования.

# **Основные шаги программы**

* 1. **Считывание данных:**
     + Программа считывает данные из файла hashes.txt и given\_numbers.txt.
  2. **Вычисление соли:**
     + Программа вычисляет соль на основе телефонных номеров и полученного хеша.
     + Выводит значение соли.
  3. **Хеширование данных:**
     + Программа хеширует телефонные номера с использованием алгоритмов MD-5, SHA-1 и SHA-256.
     + Сохраняет результаты хеширования в файлы md5.txt, sha1.txt и sha256.txt.
  4. **Дехеширование данных:**
     + Программа дехеширует хеш, полученный на предыдущем этапе и вычисляет время, которое ушло на MD-5, SHA-1 и SHA-256.
     + Делает тоже самое для различных солей и выводит время, затраченное для определенного алгоритма и соли.

**Блок схема программы**

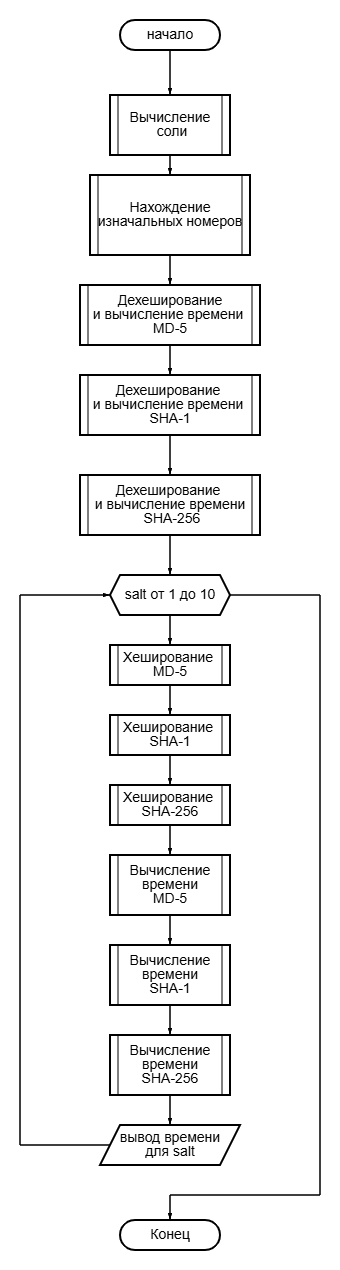


Рисунок 1. Блок-схема основной программы

# **Описание программы**

Программная реализация написана на языке Python 3.11.8 с использованием следующих библиотек: hashlib, os, time. Программа организована в одном модуле, который содержит все функции для чтения данных, их деобезличивания, вычисления соли, хеширования данных и управления процессом. В таблице 5 представлено описание функций, используемых в программе.

*Таблица 5. Описание функций*

| Функция | Описание |
| --- | --- |
| find\_salt() | Вычисляет соль на основе телефонных номеров и сохраненного хеша. |
| hash\_sha1() | Хеширует телефонные номера с использованием алгоритма SHA-1 и сохраняет результаты в файл. |
| hash\_sha256() | Хеширует телефонные номера с использованием алгоритма SHA-256 и сохраняет результаты в файл. |
| put\_correct\_phones\_into\_file() | На основе полученной соли вычисляет изначальные телефонные номера и создает с ними файл. |
| hash\_md5() | Хеширует телефонные номера с использованием алгоритма MD-5 и сохраняет результаты в файл. |
| get\_time\_hashcat() | Запускает hashcat с заданным алгоритмом и вычисляет время затраченное на расшифровку хеша. |
| main() | Основная функция программы, которая управляет процессом загрузки файла, деобезличиванием данных, вычислением соли и хешированием данных. |

# **Рекомендации пользователя**

Убедитесь, что у вас установлен Python 3.11.8 или более поздняя версия. Установите необходимые библиотеки: hashlib, os, time.

Проверьте, что все текстовые файлы находятся в одной директории с файлом скрипта.

Запустите код в среде разработки (например, PyCharm, VSCode) или через командную строку.

После завершения работы скрипта проверьте файлы output\_md5.txt, output\_sha1.txt, output\_sha256.txt на наличие ошибок и корректность данных. Если программа обнаружила проблемы с вычислением соли или хешированием данных, проверьте вывод в консоли для получения дополнительной информации.

Убедитесь, что файлы с хешами и восстановленными данными хранятся в безопасном месте и недоступны для неавторизованных пользователей. Используйте соль для обеспечения дополнительной безопасности хеширования данных.

Если вы работаете с большими объемами данных, рассмотрите возможность использования более мощных вычислительных ресурсов или оптимизации алгоритмов.

Перед использованием программы в реальных условиях, протестируйте ее на тестовых данных, чтобы убедиться в корректности работы всех функций.

# **Рекомендации программиста**

Регулярно обновляйте используемые библиотеки и версию Python, чтобы обеспечить актуальность и работоспособность кода.

Организуйте код в соответствии с best practices, используя понятные имена для переменных и функций. Разделите код на модули и функции, чтобы улучшить читаемость и поддерживаемость.

### Используйте конфигурационные файлы или переменные окружения для хранения настроек. Это позволит легко изменять параметры программы без изменения кода.

Используйте переменные окружения для хранения чувствительных данных, таких как пути к файлам или настройки безопасности.

Убедитесь, что все входные данные проверяются и очищаются, чтобы избежать атак через внедрение данных. Используйте соль для обеспечения дополнительной безопасности хеширования данных.

Оптимизируйте алгоритмы и используйте эффективные структуры данных для улучшения производительности. Используйте многопоточность или асинхронное программирование для обработки больших объемов данных.

Используйте юнит-тесты для проверки корректности работы функций. Предоставьте подробную документацию для каждой функции и модуля, чтобы облегчить понимание и использование кода. Используйте комментарии для описания логики работы кода.

# **Контрольный пример**

### 1. Запуск программы

* Откройте исходный xlsx файл и скопируйте первый столбец в файл hashes.txt и третий столбец в в файл given\_numbers.txt.
* Откройте командную строку или среду разработки.
* Перейдите в директорию, где находится файл decryption\_hash.py.
* Нажмите кнопку для запуска программы.

### 2. Программа автоматически считывает данные из given\_numbers.txt и hashes.txt.

* Пример исходного датасета показан на рисунке 2.

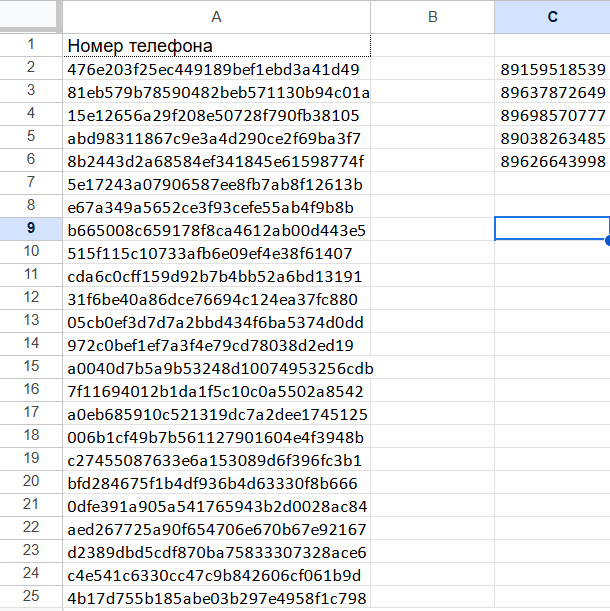


Рисунок 2. Пример исходного датасета

### 3. Деобезличивание данных

* Программа автоматически деобезличивает данные и сохраняет восстановленные телефонные номера в файл dehashed\_given\_hash.txt
* Пример вывода восстановленных номеров показан на рисунке 3.

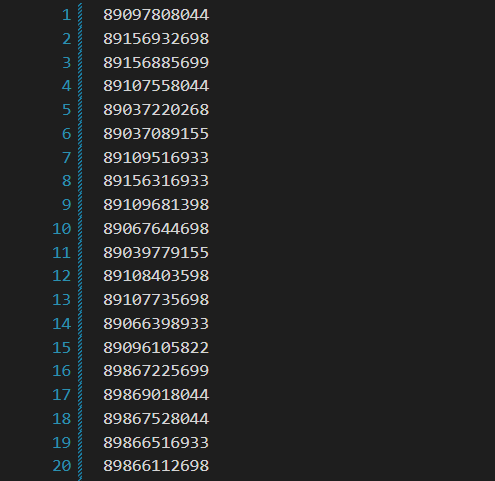


Рисунок 3. Пример вывод восстановленных номеров

### 4. Вычисление соли

* Программа вычисляет соль на основе телефонных номеров и сохраненных чисел.
* Пример вывода значения соли показан на рисунке 4.



Рисунок 4. Пример вывода значения соли

### 5. Хеширование данных

* Программа хеширует телефонные номера с использованием алгоритмов MD-5, SHA-1, SHA-256 и сохраняет результаты в файлы output\_md5.txt, output\_sha1.txt и output\_sha256.txt.
* Пример вывода хешей показаны на рисунках 5, 6, 7 соответственно.

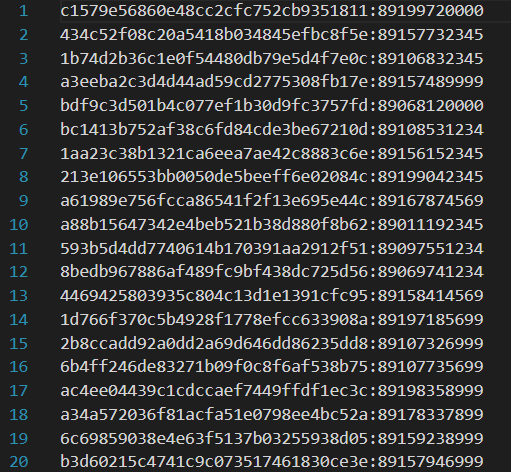


Рисунок 5. Пример файла output\_md5.txt



Рисунок 6. Пример файла output\_sha1.txt



Рисунок 7. Пример файла output\_sha256.txt

### 7. Проверка результатов

* Проверьте файлы output\_md5.txt output\_sha1.txt и output\_sha256.txt на наличие ошибок и корректность данных.
* Убедитесь, что восстановленные телефонные номера соответствуют исходным данным, а хеши соответствуют ожидаемым значениям.

# **Анализ результатов работы**

Влияние вида соли на скорость расшифровки датасета.

### Данные для анализа:

1. **Соль длиной 1**

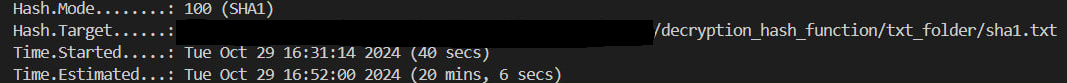


Рисунок 8. Время расшифровки sha1 с солью длиной 1

1. **Соль длиной 5**

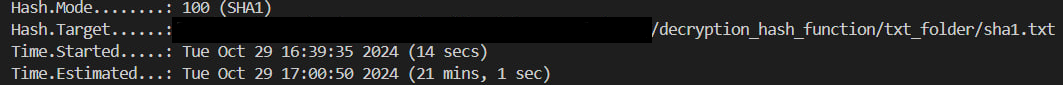


Рисунок 9. Время расшифровки sha1 с солью длиной 5

1. **Соль длиной 10**

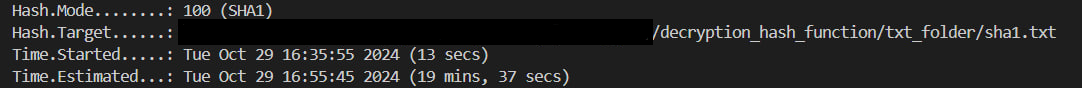


Рисунок 10. Время расшифровки sha1 с солью длиной 10

Из представленных данных видно, что увеличение длины соли незначительно влияет на скорость расшифровки датасета. Основное влияние на время расшифровки оказывает количество возможных комбинаций. В данном случае, количество возможных комбинаций остается неизменным, поэтому увеличение длины соли не приводит к значительному увеличению времени расшифровки.

Влияние хеш-функции на скорость расшифровки датасета.

### Данные для анализа:

1. **SHA1**

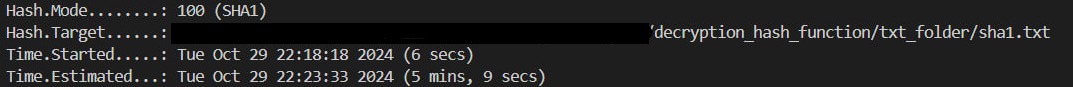


Рисунок 15. Данные для анализа SHA-1

1. **SHA-256**

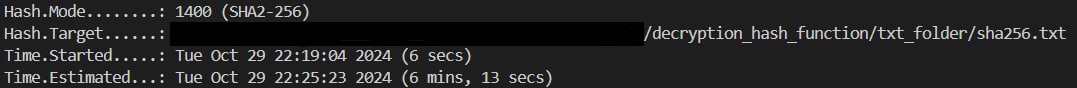


Рисунок 16. Данные для анализа SHA-256

1. **MD-5**

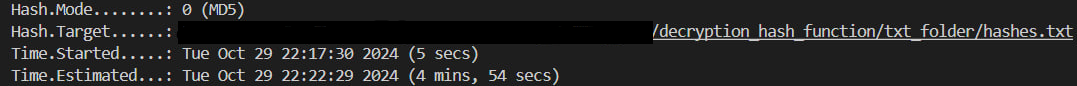


Рисунок 17. Данные для анализа MD-5

### Анализ

* MD-5 — создает 128-битный хеш, хотя MD5 когда-то широко использовался для защиты данных и проверки целостности, в настоящее время он считается устаревшим. Тем не менее, MD5 все еще используется в некоторых приложениях, где не требуется высокая криптографическая стойкость, например, для быстрой проверки данных и контроля целостности файлов
* SHA-1 — это относительно быстрый алгоритм хеширования, что объясняет короткое время расшифровки. Широко используется, но считается менее безопасным по сравнению с SHA-256 и SHA-512. Из-за его низкой сложности и потенциальных уязвимостей, рекомендуется использовать более современные алгоритмы.
* SHA-256 требует больше вычислительных ресурсов по сравнению с SHA1, что приводит к увеличению времени расшифровки. Однако, этот алгоритм считается достаточно безопасным и широко применяется в различных областях.

Из представленных данных видно, что более сложные и ресурсоемкие алгоритмы хеширования, такие как SHA-1 и SHA-256, не значительно увеличивают время расшифровки датасета. Это связано с тем, что эти алгоритмы используют более сложные математические операции и генерируют хеши большей длины. SHA-1, будучи более простым алгоритмом, позволяет быстрее получить результаты, но при этом имеет более низкий уровень безопасности.

# **Вывод**

В рамках данной работы были исследованы принципы дехеширования и хеширования телефонных номеров с использованием различных хеш-функций и методов защиты данных. Разработан алгоритм, который учитывает особенности телефонных номеров, методы хеширования и использование соли для повышения безопасности данных.