**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет прикладной математики-процессов управления**

**Программа бакалавриата**

**“Большие данные и распределенная цифровая платформа”**

**ОТЧЕТ**

**по лабораторной работе №1**

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

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

**Вариант – 1**

**Студент гр. 23Б15-пу**

**Абрахин Е.Д.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc180013329)
2. [Описание задачи (формализация задачи) 3](#_Toc180013330)
3. [Теоретическая часть 4](#_Toc180013331)
4. [Основные шаги программы 7](#_Toc180013332)
5. [Блок схема программы 10](#_Toc180013333)
6. [Описание программы 12](#_Toc180013334)
7. [Рекомендации пользователя 15](#_Toc180013335)
8. [Рекомендации программиста 16](#_Toc180013336)
9. [Исходный код программы 16](#_Toc180013337)
10. [Контрольный пример 17](#_Toc180013338)
11. [Анализ результатов работы 22](#_Toc180013339)
12. [Вывод 25](#_Toc180013340)
13. [Источники 26](#_Toc180013341)

# Цель работы

Целью лабораторной работы является исследование хеш-функций с различными вводными условиями, включая применение соли при шифровании данных. В рамках работы будет проведена расшифровка набора данных, зашифрованного с использованием хеш-функций, и анализ различных факторов, влияющих на скорость расшифровки.

# Описание задачи (формализация задачи)

1. Изучение особенностей шифрования телефонных номеров:
   * Исследовать методы и алгоритмы, используемые для шифрования телефонных номеров с использованием хеш-функций и соли.
2. Разработка программы для деобезличивания датасета:
   * Написать программу, которая будет применять хеш-функции для шифрования телефонных номеров с использованием различных соль и исследовать их влияние на скорость расшифровки.
3. Тестирование на других хеш-функциях:
   * Программа должна быть протестирована на минимум трех различных хеш-функциях, представляющих разные семейства, чтобы сравнить эффективность и скорость расшифровки.
4. Анализ влияния различных факторов на скорость расшифровки:
   * Изучить, как различные виды соли (числовые, буквенные, комбинированные), их длина, а также выбор хеш-функции влияют на скорость расшифровки.
   * Определить, сколько телефонных номеров необходимо знать из предложенного датасета, чтобы гарантированно его расшифровать.

# Теоретическая часть

Для Хеш-функции

Хеш-функции представляют собой алгоритмы, которые преобразуют входные данные произвольной длины в строку фиксированной длины, известную как хеш-значение или хеш-код. Эти функции широко используются в криптографии и информационной безопасности для обеспечения целостности данных, а также для аутентификации и хранения паролей. Основная идея хеширования заключается в том, что, даже не имея доступа к исходным данным, можно проверить их целостность, сравнив хеш-значения.

Одной из ключевых характеристик хеш-функций является уникальность: для любых двух различных входных значений не может быть получено одинаковое хеш-значение. Это свойство называется коллизией, и его отсутствие является одним из критериев оценки качества хеш-функции. Классическими примерами хеш-функций являются MD5, SHA-1, SHA-256 и SHA-512.

* MD5: Хотя раньше считался стандартом для хеширования, в настоящее время он признан небезопасным из-за уязвимостей, позволяющих находить коллизии.
* SHA-1: Этот алгоритм также устарел и больше не рекомендуется для использования в новых системах из-за его уязвимости к атакам на коллизии.
* SHA-256: Часть семейства SHA-2, данный алгоритм обеспечивает высокий уровень безопасности и считается надежным для использования в современных приложениях.
* SHA-512: Также часть семейства SHA-2, SHA-512 использует более длинное хеш-значение (512 бит) по сравнению с SHA-256, что обеспечивает дополнительный уровень безопасности и устойчивости к атакам. Он подходит для сценариев, где требуется высокая степень защиты, например, для хранения паролей и цифровых подписей.

Техника взлома Brute Force

Техника взлома Brute Force предполагает перебор всех возможных комбинаций в поиске правильного пароля или ключа. Этот метод не требует знания о структуре данных и может использоваться для расшифровки данных, защищенных хеш-функциями. Brute Force является медленным, но достаточно эффективным методом, особенно при отсутствии дополнительных мер безопасности.

В контексте хеширования, атака методом Brute Force может быть применена для нахождения исходных данных, соответствующих конкретному хеш-значению. Если хеш-функция не использует соли или другие защитные меры, злоумышленник может быстро протестировать множество вариантов и найти соответствующее значение.

Соль

"Соль" представляет собой случайное значение, добавляемое к исходным данным перед хешированием. Использование соли значительно усложняет задачу злоумышленников, так как даже если два пользователя имеют одинаковые пароли, их хеш-значения будут различаться из-за уникальной соли. Это устраняет проблему одинаковых хеш-значений для одинаковых входных данных, что делает атаку методом подбора Brute Force более сложной и ресурсоемкой.

Каждый уникальный входной набор получает свою уникальную соль, что значительно усиливает безопасность хеширования. Соль помогает предотвратить предрассчитанные атаки, такие как атаки с использованием радужных таблиц, которые направлены на быстрое нахождение соответствия между хеш-значениями и исходными данными.

Хеширование телефонных номеров

При шифровании телефонных номеров с использованием хеш-функций необходимо учитывать особенности хранения и обработки данных. В данной лабораторной работе телефонные номера подвергаются хешированию с использованием соли, что увеличивает сложность атаки, основанной на анализе хеш-значений.

Использование хеш-функций и соли в данной задаче позволяет не только защитить данные от несанкционированного доступа, но и анализировать влияние различных параметров (таких как длина соли и выбор хеш-функции) на скорость расшифровки данных. Таким образом, задача лабораторной работы включает в себя как теоретическое, так и практическое исследование методов обеспечения безопасности данных через хеширование.

# Основные шаги программы

Основные шаги программы для трех модулей: Decryption.py, Encryption.py и SaltEncry

Decryption.py

1. Запуск программы.
2. Пользователь выбирает файл с хешами в формате Excel.
3. Пользователь выбирает файл для сохранения расшифрованных данных в формате TXT.
4. Извлечение данных:
   * Загружается файл Excel и извлекаются хеши.
   * Если файл пуст, отображается сообщение об ошибке.
5. Определение типа хеша:
   * На основе длины первого хеша определяется его тип (MD5, SHA-1, SHA-256, SHA-512).
   * Если тип не удается определить, отображается сообщение об ошибке.
6. Запуск Hashcat:
   * Формируется команда для выполнения Hashcat с указанными параметрами.
   * Проверяется наличие файлов, необходимых для расшифровки.
   * Если возникает ошибка при выполнении Hashcat, отображается сообщение об ошибке.
7. Пользователь выбирает файл с расшифрованными данными, содержащий хеши и номера телефонов.
8. Пользователь выбирает файл для сохранения номеров без соли в формате TXT.
9. Поиск и применение соли:
   * Загружается файл с расшифрованными данными и известные номера для вычисления соли.
   * Производится вычисление соли и применение её к расшифрованным номерам.
   * Результаты сохраняются в указанный файл.

Encryption.py

1. Запуск программы.
2. Пользователь выбирает файл с номерами телефонов в формате TXT.
3. Пользователь выбирает файл для сохранения хешей в формате Excel.
4. Пользователь выбирает метод хеширования (MD5, SHA-1, SHA-256, SHA-512).
5. Хеширование номеров телефонов:

* Читаются номера телефонов из входного файла.
* Каждый номер хешируется с использованием выбранного метода.
* Хеши сохраняются в Excel файл.

1. Обработка ошибок:

* Если файлы не выбраны или возникает ошибка при чтении/записи, выводится сообщение об ошибке.

SaltEncryption.py

1. Запуск программы.
2. Чтение номеров из файла.
3. Генерация соли:
   * Создаются разные типы соли (числовая, буквенная, комбинированная) разной длины.
4. Хеширование с солью:
   * Для каждого номера создается хеш с применением соли.
   * Хеши сохраняются в файлы для каждого типа хеширования.
5. Генерация команд для Hashcat:
   * Формируются команды для выполнения Hashcat на основе сохраненных хешей.
   * Все команды сохраняются в файл hashcat\_commands.txt.
6. Обработка ошибок:
   * Если возникли ошибки при чтении файла или создании директорий, отображаются сообщения об ошибках.

# Блок схема программы

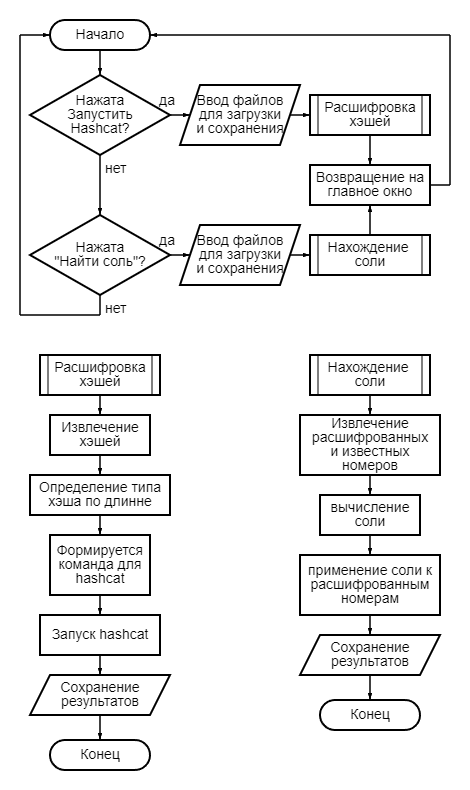


Рис 1. Блок-схема Decryption.py

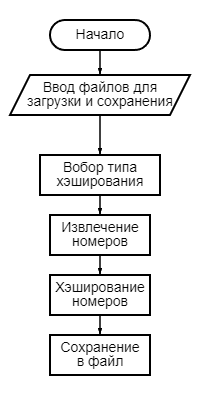


Рис 2. Блок-схема Encryption.py

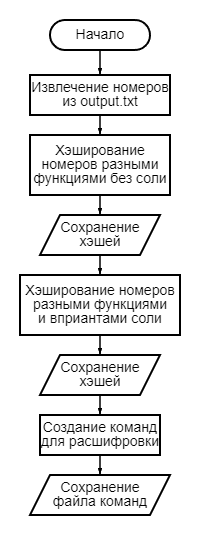


Рис 3. Блок-схема SaltEncryption.py

# Описание программы

Программная реализация написана на языке Python 3.12.6 с использованием следующих библиотек: pandas [[1]](#_Источники), tkinter [[2]](#_Источники) и hashlib [[3]](#_Источники). Программа организована через модульную структуру с акцентом на генерацию данных для железнодорожных билетов. В процессе разработки программы использовались 11 функций, каждая из которых имеет чётко определённое назначение и 4 основных модуля:

Таблица 1. Decryption.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| selectEncryptedFile | Открывает диалоговое окно для выбора зашифрованного файла Excel. | None |
| selectDecryptedSaveFile | Открывает диалог для сохранения расшифрованных данных в файл формата TXT. | None |
| selectDecryptedFile | Открывает диалог для выбора файла с расшифрованными данными (формат TXT). | None |
| selectSaltRemovedFile | Открывает диалог для сохранения результата удаления соли в TXT файл. | None |
| extractData | Извлекает хеши из Excel файла и сохраняет их в текстовом файле для Hashcat. | tuple |
| determineHashType | Определяет тип хеша по его длине (MD5, SHA-1, SHA-256, SHA-512). | str |
| runHashcat | Запускает процесс расшифровки хешей с использованием Hashcat. | None |
| findAndApplySalt | Ищет соль и применяет её к расшифрованным данным для восстановления номеров. | None |

Таблица 2. Encryption.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| SelectInputFile | Открывает диалоговое окно для выбора текстового файла с номерами телефонов. | None |
| SelectOutputFile | Открывает диалог для сохранения файла с хешами в формате Excel. | None |
| HashPhoneNumbers | Генерирует хеши номеров телефонов с использованием выбранной хеш-функции. | None |
| HashUsingMD5 | Хеширует номера телефонов с использованием MD5. | None |
| HashUsingSHA1 | Хеширует номера телефонов с использованием SHA-1. | None |
| HashUsingSHA256 | Хеширует номера телефонов с использованием SHA-256. | None |
| HashUsingSHA512 | Хеширует номера телефонов с использованием SHA-512. | None |

Таблица 3. SaltEncryption.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| readNumbersFromFile | Читает номера телефонов из текстового файла. | list |
| generateNumericSalt | Генерирует цифровую соль заданной длины. | str |
| generateAlphaSalt | Генерирует буквенную соль заданной длины. | str |
| generateCombinedSalt | Генерирует комбинированную соль (цифры и буквы) заданной длины. | str |
| hashNumberWithSalt | |  | | --- | |  |  |  | | --- | | Генерирует хеш номера телефона с солью. | | str |
| saveHashesToFile | Сохраняет хеши в файл для использования Hashcat. | None |
| generateHashcatCommand | |  | | --- | |  |  |  | | --- | | Создает команду для запуска Hashcat с определенным типом хешей и маской. | | str |
| processHashing | Генерирует хеши для номеров с различными типами солей и записывает команды Hashcat. | None |

# 

# Рекомендации пользователя

Программа позволяет расшифровывать хешированные телефонные номера с использованием утилиты Hashcat [[4]](#_Источники) и производить хеширование номеров с добавлением соли.

Процесс расшифровки:

1. Запустите скрипт Decryption.py.
2. Выберите Excel-файл с хешами телефонных номеров и укажите место для сохранения результата в текстовый файл.
3. Нажмите кнопку "Запустить Hashcat", и программа попытается расшифровать номера, сохраняя результат в выбранный файл.
4. Для удаления соли, выберите файл с расшифрованными данными, и нажмите “Найти и применить соль” и программа найдет и применит соль для восстановления исходных номеров.

Процесс хеширования:

1. Запустите скрипт Encryption.py.
2. Выберите текстовый файл с номерами телефонов для хеширования и укажите место сохранения Excel-файла с хешами.
3. Выберите метод хеширования (MD5, SHA-1, SHA-256, SHA-512) для генерации хешей и сохранения результата.

Процесс хеширования с солью:

1. Запустите скрипт SaltEncryption.py, и он сделает несколько тестовых вариантов файлов номеров, захэшированных с разными вариантами случайных.
2. Текстовые файлы хэшей и команды к их расшифровке можно будет найти в директории hashcat рядом с исполняемыми скриптами.

# Рекомендации программиста

Для корректной работы программы убедитесь, что у вас установлен Python версии не ниже 3.12.0 и необходимые библиотеки: pandas [[1]](#_Источники), tkinter [[2]](#_Источники) и hashlib [[3]](#_Источники).

Проверьте, что все файлы программы находятся в одной директории, затем скачайте hashcat [[4]](#_Источники) и распакуйте его в директорию проекта, назвав эту папку “hashcat”. Проверьте работоспособность программы запустив, например, example0.cmd в папке с hashcat.exe.

Проведите тестирование модулей с различными файлами и форматами данных для проверки корректности работы всех функций: расшифровка, хеширование с разными алгоритмами и применением соли.

# Исходный код программы

<https://github.com/FasterXaos/Algorithms_and_Data_Structures>

# Контрольный пример

Decryption.py

1. Запуск программы расшифровки хешей.

Для запуска программы используйте файл Decryption.py. Программа отвечает за расшифровку хешей телефонных номеров, используя инструмент Hashcat [[3]](#_Источники).

1. Выбор файлов и запуск Hashcat.

После запуска программы, пользователю будет предложено выбрать файл с хешами и указать место для сохранения расшифрованных данных (Рис. 4). Затем нужно нажать на кнопку “Запустить Hashcat”. Программа автоматически определяет тип хеша (MD5, SHA-1, SHA-256, SHA-512) и запускает процесс расшифровки через Hashcat, сохранив вывод в выбранную директорию (Рис. 5).

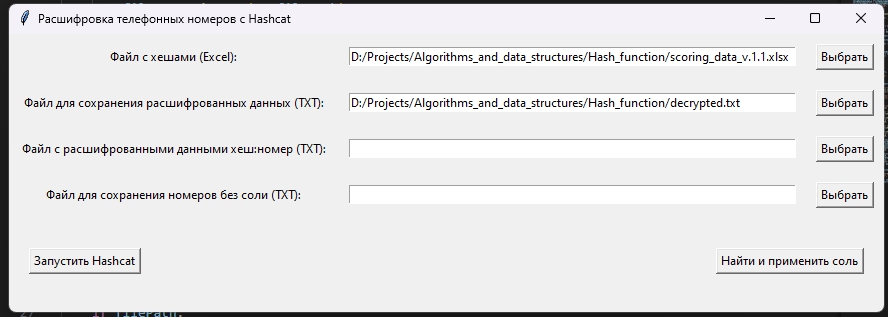


Рис 4. пример ввода файла хэшей и файла сохранения

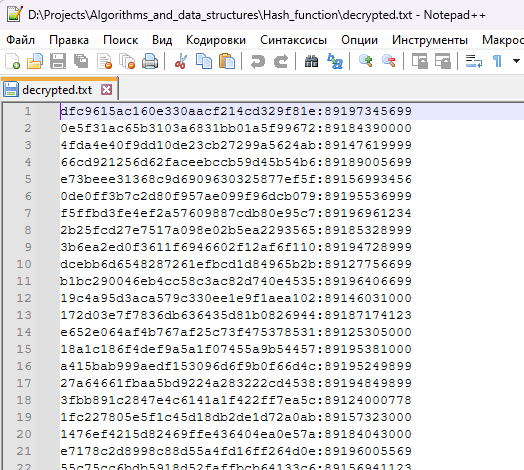


Рис 5. пример вывода в файл после hashcat

1. Нахождение и применение соли.

Для нахождения соли расшифрованного файла нужно выбрать его в третьем поле и выбрать куда сохранить итоговый вариант без соли в четвертом (Рис. 6). Затем нужно нажать на “Найти и применить соль”, после чего программа выполнит задачу и сохранит результат в выбранный файл (Рис. 7).

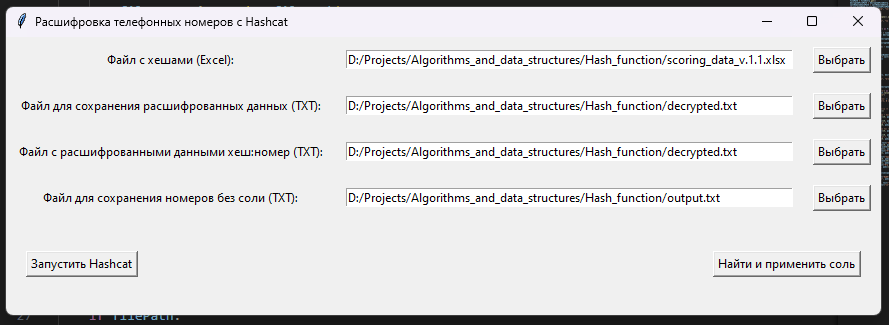


Рис 6. пример ввода файла номеров с солью и файла сохранения

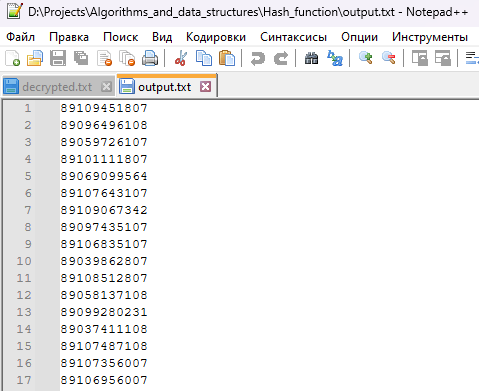


Рис 7. пример файла вывода номеров без соли

Encryption.py

1. Запуск программы шифровки номеров без соли.

Для запуска программы используйте файл Encryption.py. Программа отвечает за шифровку хешей телефонных номеров, используя инструмент hashlib [[3]](#_Источники).

1. Хэширование номеров без соли.

Для для шифровки номеров нужно выбрать файл номеров и файл сохранения (Рис. 8). Затем нужно нажать на одну из кнопок, на которых написан тип хэширования и программы произведет шифровку соответствующем типом (Рис. 9).

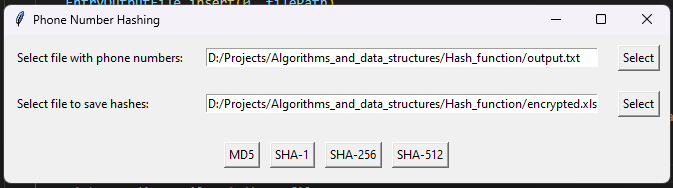


Рис 8. пример ввода файла номеров и файла сохранения

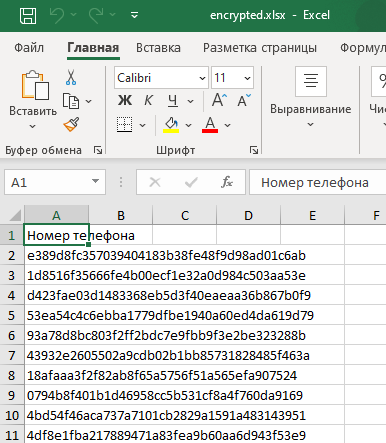


Рис 9. пример файла хэшированных номеров без соли

SaltEncryption.py

1. Запуск программы шифровки номеров c солью.

Для запуска программы используйте файл SaltEncryption.py с существующем файлом номеров output.txt в директории скрипта. Программа отвечает за шифровку хешей телефонных номеров, используя инструмент hashlib [[3]](#_Источники) c применением различных вариантов соли. После запуска программа автоматически сохранит несколько вариантов хэширования в директорию hashcat (Рис. 10) и файл с командами для их расшифровки (Рис. 11)



Рис 10. Пример файлов хэшированных номеров различными методами с солью

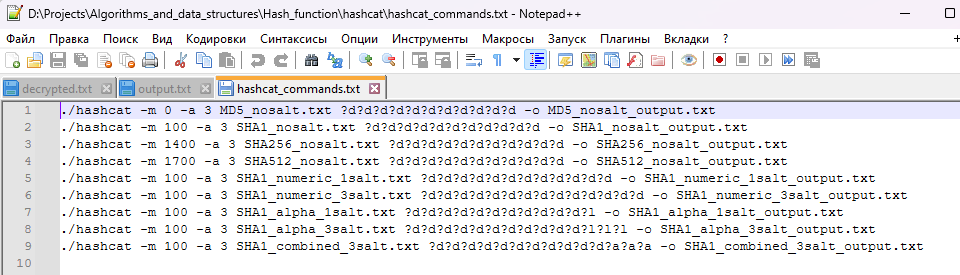


Рис 11. пример файла команд для расшифровки номеров

# Анализ результатов работы

Для тестирования исходного деобезличенного набора данных можно использовать следующие хеш-функции:

MD5 (Message Digest Algorithm 5):

* Основной алгоритм. Производит 128-битную хэш-сумму (32 символа шестнадцатеричного кода).
* Преимущества: быстрая скорость работы и простота вычисления.
* Недостатки: устаревший и подвержен коллизиям, что делает его уязвимым для криптоанализа.

SHA-1 (Secure Hash Algorithm 1):

* Производит 160-битную хэш-сумму (40 символов шестнадцатеричного кода).
* Преимущества: использовался в безопасности, но сейчас считается небезопасным из-за уязвимости к атакам с коллизиями.
* Недостатки: медленнее MD5, но всё еще подвержен взлому.

SHA-256:

* Принадлежит к семейству SHA-2 и генерирует 256-битную хэш-сумму (64 символа).
* Преимущества: значительно более безопасный, чем MD5 и SHA-1, и широко используется в современных криптографических приложениях.
* Недостатки: медленнее MD5 и SHA-1, требует больше вычислительных ресурсов.

SHA-512:

* Также из семейства SHA-2, генерирует 512-битную хэш-сумму (128 символов).
* Преимущества: максимальная безопасность из представленных алгоритмов, используется для высокоуровневой защиты данных.
* Недостатки: наибольшие вычислительные требования и время выполнения.

Влияние вида соли на дешифровку данных:

1. Численная соль:

Добавляет дополнительные числа перед хэшированием, но диапазон значений ограничен. Это слегка увеличивает количество комбинаций, но незначительно замедляет атаки перебора (Рис. 12).

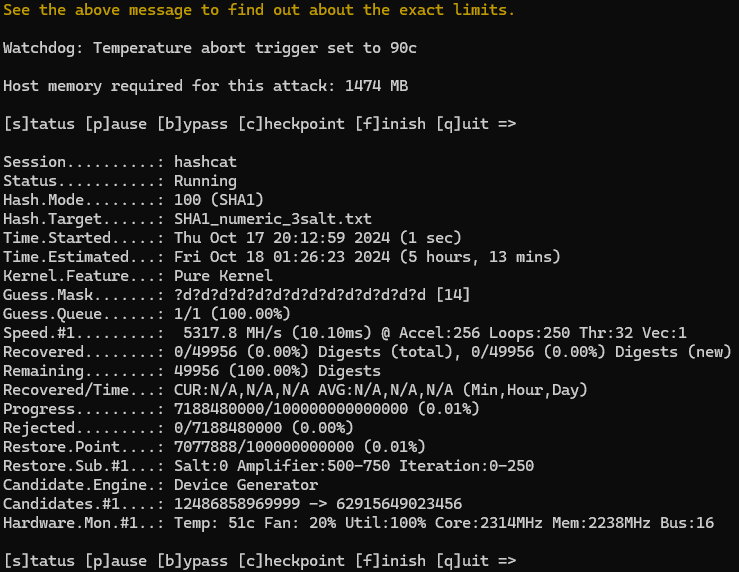


Рис 12. Пример расшифровки с численной солью

1. Буквенная соль:

Каждая добавленная буква существенно увеличивает число возможных комбинаций, что замедляет взлом паролей, хэшированных с использованием буквенной соли (Рис. 13).

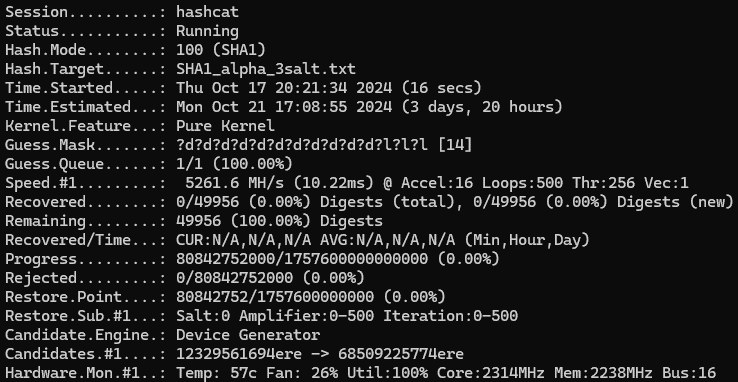


Рис 13. Пример расшифровки с буквенной солью

1. Комбинированная соль:

Комбинация чисел и букв создает значительно большее количество возможных значений, чем численная или буквенная соль отдельно. Это значительно усложняет атаки и замедляет их, так как необходимо вычислить хэш для каждой уникальной комбинации соли (Рис. 14).

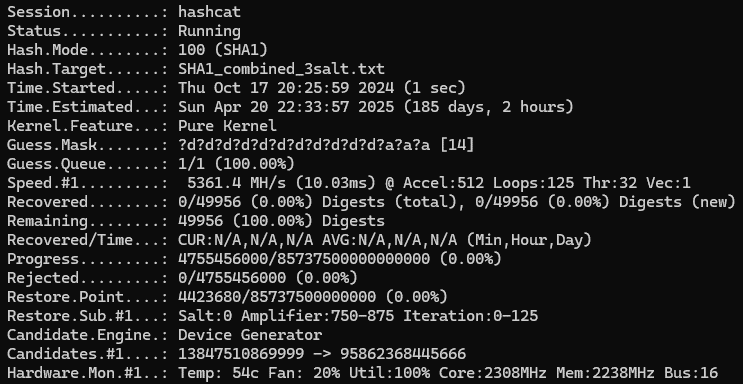


Рис 14. Пример расшифровки с комбинированной солью

Влияние длины соли:

Чем длиннее соль, тем больше количество возможных комбинаций, что экспоненциально увеличивает время, необходимое для расшифровки данных методом перебора. Например, соль длиной 5 символов даёт 62^5 (~916 млн) возможных вариантов.

Влияние хеш-функции:

MD5 работает быстрее всех остальных алгоритмов, но его безопасность крайне низка. Его можно взломать за короткое время при наличии слабой соли или вовсе без неё.

SHA-1 медленнее MD5, но, несмотря на увеличенную длину хэша, также не рекомендуется для защиты конфиденциальных данных.

SHA-256 и SHA-512 обеспечивают значительно большую стойкость к взлому за счёт длинных хэшей. Однако они работают медленнее, чем MD5 и SHA-1, особенно SHA-512.

Сколько нужно знать телефонов из предложенного датасета, чтобы 100% его взломать?

Тесты показали, что для полной дешифровки хватает 3 известных номера, тогда соль определяется однозначно (Рис. 15). Но, например, при двух известных номерах количество возможных вариантов становится 4 с одним лишь верным (Рис. 16).

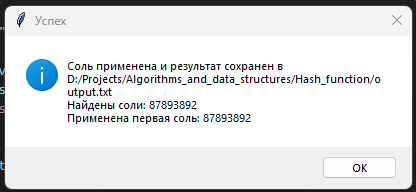


Рис 15. Пример успешного нахождения соли

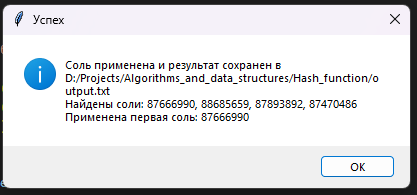


Рис 16. Пример неудачного нахождения соли

# Вывод

В рамках лабораторной работы была проведена исследовательская работа по анализу хеш-функций, использованию соли и изучению методов расшифровки хешированных телефонных номеров. Работа охватывала разработку программного обеспечения для хеширования и расшифровки данных, а также анализ влияния различных факторов на скорость расшифровки.

# Источники

1. tkinter — Python interface to Tcl/Tk // Tkinter URL:  [https://docs.python.org/3/library/tkinter.html/](file:///D:\Projects\Algorithms_and_data_structures\Dataset_anonymization\%20https:\docs.python.org\3\library\tkinter.html\) (дата обращения: 16.10.2024).
2. pandas // Pandas URL: <https://pandas.pydata.org/> (дата обращения: 16.10.2024).
3. hashlib // Hashlib URL: <https://docs.python.org/3/library/hashlib.html> (дата обращения: 16.10.2024).
4. hashcat // Hashcat URL: <https://hashcat.net/hashcat/> (дата обращения: 16.10.2024).