**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: 5**

**Студент гр. 23Б15-пу**

**Бек В.А.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc180747658)
2. [Описание задачи (формализация задачи) 4](#_Toc180747659)
3. [Теоретическая часть 5](#_Toc180747660)

[**Шифрование данных** 5](#_Toc180747661)

[**Хеширование** 5](#_Toc180747662)

[**Атака brute-force (грубой силы)** 5](#_Toc180747663)

[**Понятие соли в хешировании** 6](#_Toc180747664)

1. [Основные шаги программы 7](#_Toc180747665)
2. [Блок схема программы 8](#_Toc180747666)
3. [Описание программы 10](#_Toc180747667)
4. [Рекомендации пользователя 12](#_Toc180747668)
5. [Рекомендации программиста 13](#_Toc180747669)
6. [Контрольный пример 14](#_Toc180747670)
7. [Анализ 17](#_Toc180747671)
8. [Анализ на практике 19](#_Toc180747672)
9. [Вопрос о минимальном количестве телефонов 31](#_Toc180747673)
10. [Вывод 32](#_Toc180747674)
11. [Источники 33](#_Toc180747675)

# Цель работы

Провести исследование хеш-функций с использованием различных вводных условий для деобезличивания данных. Основная задача заключается в расшифровке датасета, зашифрованного с использованием хеш-функции и модификатора входа (соли). В процессе работы требуется проанализировать, как изменяются параметры скорости расшифровки в зависимости от вида и длины соли, а также от типа используемой хеш-функции. Дополнительно, необходимо определить, сколько телефонных номеров необходимо знать для успешного взлома всего набора данных при данных условиях.

# Описание задачи (формализация задачи)

Задача заключается в деобезличивании набора данных, содержащего хешированные телефонные номера, зашифрованные с использованием соли. Основная цель — восстановить оригинальные телефонные номера, используя метод полного перебора (Brute Force) с учетом особенностей примененной хеш-функции и соли.

Для этого необходимо реализовать программу, которая способна:

1. Расшифровывать телефонные номера, зашифрованные с помощью хеш-функций MD5, добавляя соль к исходным данным.
2. Реализовать алгоритм полного перебора для восстановления исходных данных, ориентируясь на известные телефонные номера, чтобы ускорить процесс.
3. Выполнить шифрование расшифрованных данных с использованием выбранного метода.

Также программа должна быть протестирована на:

1. Исходном наборе данных, зашифрованном хеш-функцией.
2. Зашифровке при помощи трех дополнительных алгоритмов.
3. Различных видах и длинах соли для анализа влияния этих факторов на скорость расшифровки.

# Теоретическая часть

### **Шифрование данных**

Шифрование – это процесс преобразования исходных данных в зашифрованный формат, чтобы обеспечить их защиту от несанкционированного доступа. Основной целью шифрования является обеспечение конфиденциальности информации.

### **Хеширование**

Хеширование – это процесс преобразования входных данных любой длины в строку фиксированной длины, которая называется хешем. Хеширование используется для проверки целостности данных, создания уникальных идентификаторов и аутентификации паролей. Ключевые свойства хеш-функций:

* **Определённость**: одинаковый входной набор данных всегда возвращает один и тот же хеш.
* **Односторонность**: из хеша невозможно восстановить исходные данные.
* **Устойчивость к коллизиям**: сложность подбора двух различных наборов данных, дающих одинаковый хеш.

Примеры популярных хеш-функций: MD5, SHA-1, SHA-256.

### **Атака brute-force (грубой силы)**

Атака методом brute-force (перебора) заключается в последовательной проверке всех возможных комбинаций паролей или ключей для того, чтобы найти правильное значение. Такой метод гарантирует нахождение ответа, но является крайне ресурсоёмким и неэффективным при больших ключевых пространствах. Однако, если длина пароля короткая или используется простой алгоритм шифрования, brute-force может оказаться успешным. Основные способы защиты от brute-force атак:

* Увеличение длины ключа или пароля.
* Ограничение количества попыток ввода пароля.
* Введение задержки между попытками.

### **Понятие соли в хешировании**

Соль – это случайная строка данных, которая добавляется к паролю перед его хешированием. Основная цель использования соли – предотвратить создание заранее подготовленных таблиц для подбора паролей. Даже если два пользователя имеют одинаковый пароль, благодаря добавлению разных солей их хеши будут отличаться. Пример использования соли:

1. Генерация случайной строки (соли).
2. Добавление соли к паролю перед хешированием.
3. Хранение соли вместе с хешем в базе данных.

Использование соли значительно усложняет взлом паролей путём предсказания их хешей или применения атак, основанных на таблицах хешей.

# Основные шаги программы

* **Импорт библиотек**: Программа использует библиотеки для работы с файловой системой (os), хешированием (hashlib), рандомизацией (random), и взаимодействием с Excel-файлами (pandas, openpyxl). Также используются элементы графического интерфейса (tkinter) для выбора и обработки файлов.
* **Функции выбора файлов**:
  + selectEncryptedFile(): Открывает диалоговое окно для выбора файла Excel, содержащего зашифрованные данные для расшифровки.
  + selectFileForEncryption(): Открывает диалог для выбора файла, который будет зашифрован, и активирует кнопки для выбора алгоритма хеширования.
* **Запуск программы Hashcat**:
  + runHashcat(): Запускает Hashcat для взлома паролей, используя предварительно извлеченные хеши. Команда включает в себя указание на файл хешей и маску для подбора паролей.
* **Извлечение хешей**:
  + extractPasswordHashes(): Извлекает хеши паролей из Excel-файла и сохраняет их в отдельный файл для использования Hashcat.
* **Применение соли для расшифровки**:
  + applyDecryptionSalt(): Сравнивает зашифрованные номера с известными, вычисляет соль и применяет её для расшифровки номеров, а затем сохраняет результат в Excel.
* **Хеширование номеров телефонов**:
  + hashPhoneNumbers(): Хеширует телефонные номера с использованием выбранного алгоритма хеширования и соли. Результат сохраняется в файл Excel.
* **Графический интерфейс (GUI)**:
  + Программа использует tkinter для создания интерфейса, позволяющего пользователю выбирать файлы и алгоритмы хеширования, а также запускать процессы расшифровки и хеширования.

# Блок схема программы

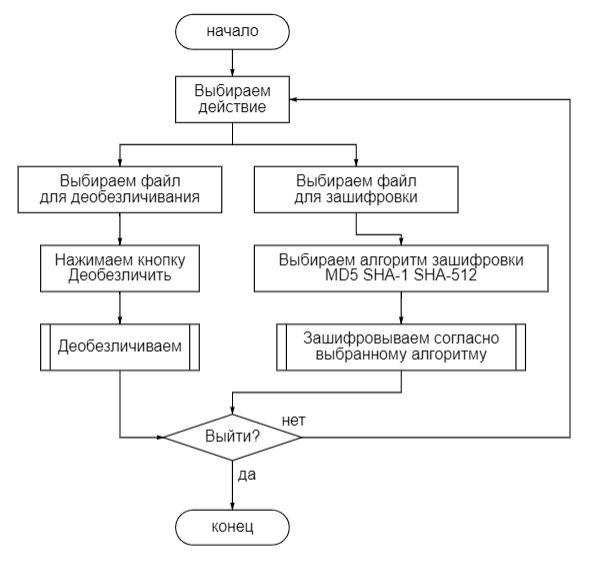


Рис 1. Блок-схема основной программы

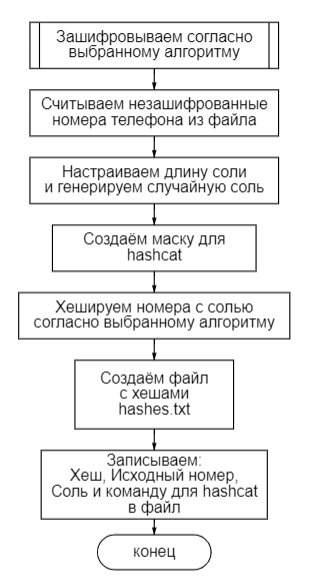
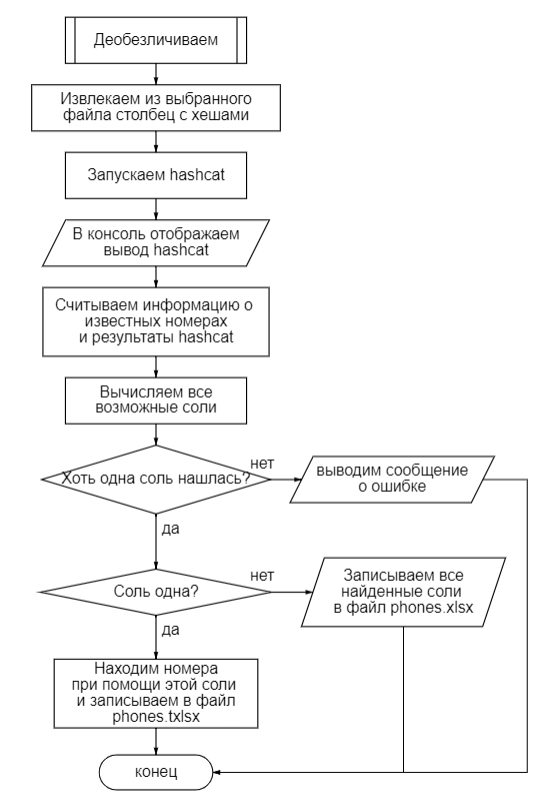


Рис 2. Блок-схема подпрограммы 1 Рис 3. Блок-схема подпрограммы 2

# Описание программы

Программная реализация написана на языке Python 3.12.7 с использованием следующих библиотек: os, string, subprocess, hashlib, random, pandas, tkinter, openpyxl. Программа организована в едином модуле, который фокусируется на деобезличивании файлов и их зашифровке. В процессе разработки программы использовалось 13 функций, каждая из которых имеет четко определенное назначение:

Таблица 1. decryption\_and\_encryption\_tool.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| selectEncryptedFile | Открывает диалоговое окно для выбора файла, предназначенного для деобезличивания, и обновляет метку. | None |
| selectFileForEncryption | Открывает диалоговое окно для выбора файла, предназначенного для шифрования, и обновляет метку. | None |
| runHashcat | Запускает Hashcat для извлечения паролей из указанного файла и возвращает путь к выходному файлу. | Путь к выходному файлу (str) |
| extractPasswordHashes | Извлекает хеши паролей из Excel файла и сохраняет их в текстовый файл. | Количество хешей и путь к файлу |
| applyDecryptionSalt | Находит и применяет соль к расшифрованным номерам и сохраняет результат в Excel файл. | None |
| startDeobfuscationProcess | Запускает процесс деобезличивания, включая извлечение паролей и применение соли. | None |
| generate\_salt | |  | | --- | |  |  |  | | --- | | Генерирует случайную соль заданной длины. | | Случайная соль (str) |
| hashPhoneNumbers | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Хеширует номера телефонов с использованием заданной функции хеширования и сохраняет их в файл. | | | None |
| |  | | --- | | hashMD5 |  |  |  | | --- | --- | |  |  | | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Хеширует номера телефонов с использованием алгоритма MD5. | | | None |
| hashSHA1 | Хеширует номера телефонов с использованием алгоритма SHA-1. | None |
| hashSHA512 | Хеширует номера телефонов с использованием алгоритма SHA-512. | None |
| centerWindow | Центрирует заданное окно на экране. | None |

# Рекомендации пользователя

Для корректного выполнения программы следуйте приведённым шагам:

1. **Выбор файла для деобезличивания**:
   * Нажмите кнопку **"Выбрать файл для деобезличивания"**. Откроется диалоговое окно, в котором нужно выбрать файл Excel с телефонными номерами, которые необходимо деобезличить.
   * После выбора файла его имя отобразится в интерфейсе.
2. **Деобезличивание**:
   * После выбора файла для деобезличивания нажмите кнопку **"Деобезличить"**. Это запустит процесс деобезличивания, который будет использовать выбранный файл и производить необходимые вычисления. Результаты будут сохранены в файл phones.xlsx.
3. **Выбор файла для зашифровки**:
   * Нажмите кнопку **"Выбрать файл для зашифровки"**. В этом диалоговом окне выберите файл, который является результатом работы деобезличивания (то есть, файл phones.xlsx, который был создан на предыдущем шаге).
   * После выбора файла его имя отобразится в интерфейсе, и кнопки для выбора метода хеширования (MD5, SHA-1, SHA-512) станут активными.
4. **Хеширование телефонных номеров**:
   * Выберите один из методов хеширования, нажав соответствующую кнопку:
     + **MD5**
     + **SHA-1**
     + **SHA-512**
   * После выбора метода хеширования программа создаст хешированные значения телефонных номеров и сохранит их в новый файл Excel, имя которого будет указано в сообщении.

Следуя этим шагам, вы сможете успешно использовать программу для деобезличивания и хеширования телефонных номеров.

# Рекомендации программиста

Для корректного функционирования программы рекомендуется выполнить следующие действия:

1. **Установите необходимые библиотеки**:
   * Убедитесь, что установлены следующие библиотеки Python:
     + openpyxl: для работы с файлами Excel.



* + - hashlib: встроенная библиотека Python для хеширования (не требует установки).
    - tkinter: стандартная библиотека для создания графического интерфейса (входит в стандартную библиотеку Python).

1. **Установите Hashcat**:
   * Скачайте последнюю версию Hashcat с официального сайта.
   * Следуйте инструкциям по установке, указанным на сайте, в зависимости от вашей операционной системы (Windows, Linux или macOS).
   * Убедитесь, что вы имеете доступ к командной строке и можете запускать Hashcat из неё.
2. **Проверьте наличие необходимых файлов**:
   * Убедитесь, что файлы:
     + phones.xlsx (результат деобезличивания) создаются в процессе работы программы. Если файл отсутствует после выполнения шага деобезличивания, программа не сможет продолжить работу.
   * Если вы используете собственные файлы для тестирования, убедитесь, что они имеют правильный формат и содержат ожидаемые данные.

Следуя этим рекомендациям, вы сможете обеспечить корректную работу программы для деобезличивания и хеширования телефонных номеров.

**Код программы:**

**<https://github.com/Kliooo/Algorithms-and-data-structures>**

# Контрольный пример

* Запуск программы: Для запуска программы используйте файл **decryption\_and\_encryption\_tool.py.** Программа запустит графический интерфейс (Рис. 4), в котором можно будет выбрать “Выбрать файл для деобезличивания” или “ Выбрать файл для зашифровки”.

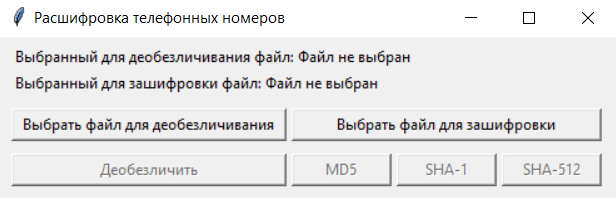


Рис 4. Графический интерфейс программы

* Выбор файла: Первым делом нам надо выбрать файл для работы. При нажатии на копку: “Выбрать файл для деобезличивания” программа откроет проводник, где пользователю предложено выбрать xlsx файл (Рис.5).

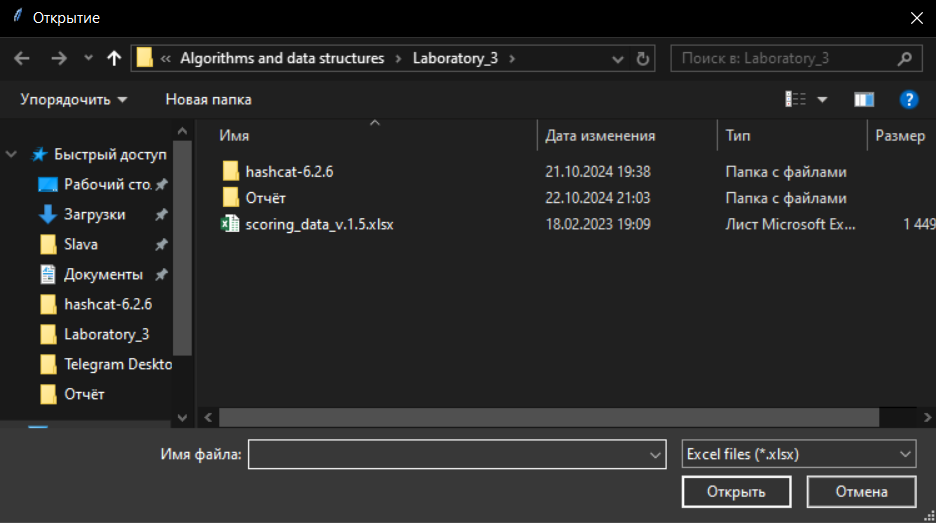


Рис 5. Выбор файла для деобезличивания

* После выбора файла для деобезличивания разблокируется кнопка: “Деобезличить” (Рис.6). Нажимаем её, ждём, и на выходе получаем файл: phones.xlsx.

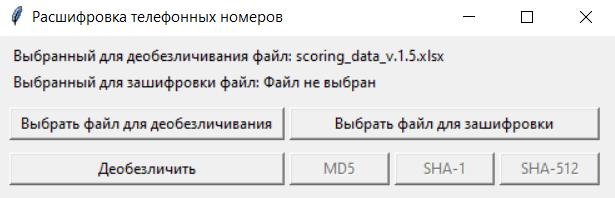


Рис 5. Разблокировка кнопки: “Деобезличить”

* Зашифровка: Первым делом нам надо снова выбрать файл, с которым будем работать на кнопку “Выбрать файл для зашифровки”. Нам нужен: phones.xlsx (Рис.6). Теперь, будут разблокированы кнопки: “MD5” “SHA-1” “SHA-512”.(Рис. 7).

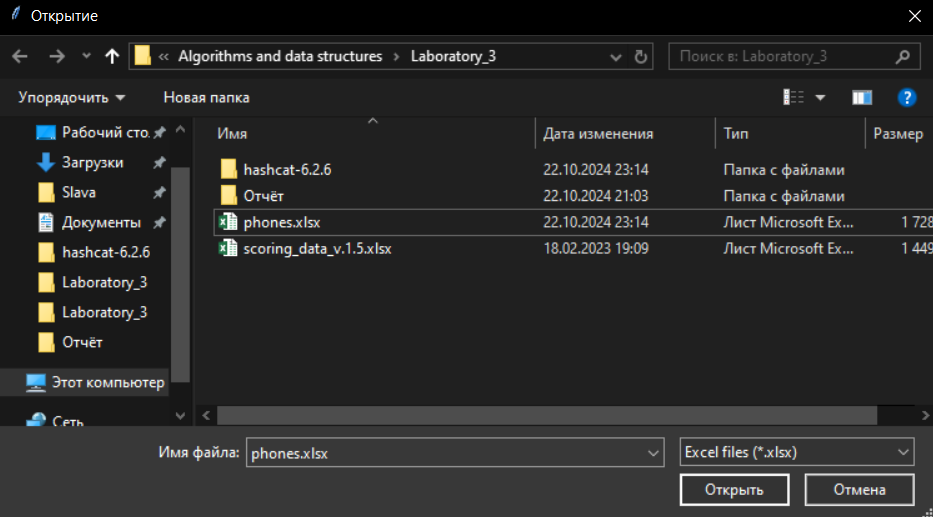


Рис 6. Выбор файла для зашифровки

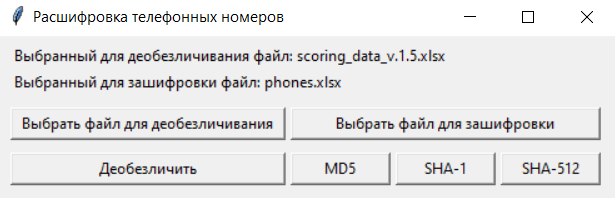


Рис 7. Разблокировка кнопок

* Зашифровка: Осталось выбрать требующуюся зашифровку и подождать завершения программы. (Рис. 8)

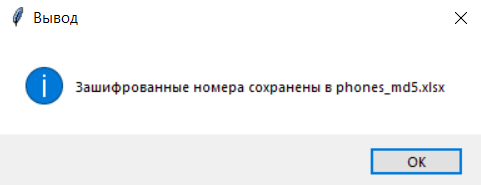


Рис 8. Завершение программы

* Анализ: Для анализа времени расшифровки получившегося файла надо открыть терминал и зайти в директорию где расположен hashcat.exe, а далее запустить код(Он будет меняться в зависимости от расшифровки, а найти его можно в файле: phones\_###.xlsx):  
  
* Анализ: Выведется статус работы, в нём можно будет посмотреть сколько времени займёт расшифровка. (Рис. 9)

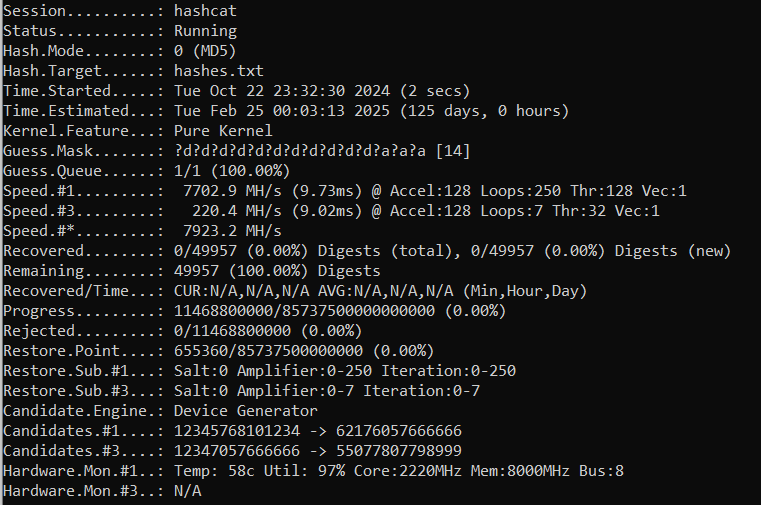


Рис 9. Статус работы hashcat

# Анализ

Скорость расшифровки данных (например, при атаке на хеши) зависит от нескольких факторов, включая вид соли, длину соли, а также используемую хеш-функцию. Рассмотрим каждый из этих факторов подробнее:

* **Вид соли**

Соль — это случайная строка, добавляемая к данным перед их хешированием, чтобы предотвратить использование радужных таблиц (precomputed hash tables) для взлома паролей. Вид соли может влиять на скорость взлома за счет увеличения сложности и числа возможных комбинаций.

* + **Цифровая соль** (только цифры) легче и быстрее для перебора, так как возможных вариантов меньше (всего 10 символов).
  + **Соль из букв** (строчные и прописные) даёт 52 возможных символа для каждой позиции, что уже увеличивает время на перебор всех комбинаций.
  + **Алфавитно-цифровая соль** (цифры + буквы) ещё сложнее, так как каждая позиция может содержать 62 варианта символов.
  + **Соль из всего набора символов** (буквы, цифры, спецсимволы) значительно увеличивает число возможных комбинаций. Такой вид соли делает перебор существенно медленнее за счёт использования всех доступных символов (до 94 на одну позицию), что значительно затрудняет взлом.
* **Длина соли**

Чем длиннее соль, тем больше возможных уникальных комбинаций, которые нужно проверить злоумышленнику при атаке. Длина соли напрямую влияет на время, необходимое для подбора хеша.

* **Хеш-функция**

Разные хеш-функции имеют разную сложность вычисления. Более сложные функции могут требовать большего времени на каждый цикл генерации хеша, что влияет на скорость взлома:

* **Быстрые хеш-функции** (например, MD5 или SHA-1) производят хеши быстрее, что делает их менее защищёнными от атак брутфорс. При этом злоумышленник может сгенерировать огромное количество хешей за короткий промежуток времени.
* **Медленные хеш-функции** (например, bcrypt, scrypt или Argon2) специально разработаны для замедления процесса хеширования. Они используют больше ресурсов (времени и памяти) на каждый цикл, что значительно усложняет атаку, даже если другие параметры (например, вид или длина соли) относительно просты.

# Анализ на практике

Попробуем на практике посмотреть, как будет меняться время расшифровки в зависимости от разных ситуаций.

Таблица 2. Анализ расшифровки

|  |  |  |  |
| --- | --- | --- | --- |
| **Алгоритм** | **Тип** **соли** | **Длина соли** | **Время** |
| **MD5** | Буквенная | 1 | 5 мин. |
| **MD5** | **Спецсимволы** | 1 | 6 мин. |
| **MD5** | Спецсимволы, цифры, буквы | 1 | 20 мин. |
| **MD5** | Буквенная | 2 | 2 ч. |
| **MD5** | **Спецсимволы** | 2 | 3 ч. |
| **MD5** | Спецсимволы, цифры, буквы | 2 | 1 д. |
| **MD5** | Буквенная | 3 | 2 д. |
| **MD5** | **Спецсимволы** | 3 | 5 д. |
| **MD5** | Спецсимволы, цифры, буквы | 3 | 124 д. |
| **MD5** | Буквенная | 4 | 1 г. |
| **MD5** | **Спецсимволы** | 4 | 3 г. |
| **MD5** | Спецсимволы, цифры, буквы | 4 | - |
| **MD5** | Буквенная | 5 | 4 г. |
| **MD5** | **Спецсимволы** | 5 | 15 лет |
| **MD5** | Спецсимволы, цифры, буквы | 5 | - |
| **MD5** | Буквенная | 6 | - |
| **MD5** | **Спецсимволы** | 6 | - |
| **MD5** | Спецсимволы, цифры, буквы | 6 | - |
| **SHA-1** | Буквенная | 1 | 9 мин. |
| **SHA-1** | **Спецсимволы** | 1 | 12 мин. |
| **SHA-1** | Спецсимволы, цифры, буквы | 1 | 35 мин. |
| **SHA-1** | Буквенная | 2 | 4 ч. |
| **SHA-1** | **Спецсимволы** | 2 | 6 ч. |
| **SHA-1** | Спецсимволы, цифры, буквы | 2 | 2 д. |
| **SHA-1** | Буквенная | 3 | 4 д. |
| **SHA-1** | **Спецсимволы** | 3 | 9 д. |
| **SHA-1** | Спецсимволы, цифры, буквы | 3 | 226 д. |
| **SHA-1** | Буквенная | 4 | 1 г. |
| **SHA-1** | **Спецсимволы** | 4 | 4 г. |
| **SHA-1** | Спецсимволы, цифры, буквы | 4 | - |
| **SHA-1** | Буквенная | 5 | 8 лет |
| **SHA-1** | **Спецсимволы** | 5 | 28 лет |
| **SHA-1** | Спецсимволы, цифры, буквы | 5 | - |
| **SHA-1** | Буквенная | 6 | - |
| **SHA-1** | **Спецсимволы** | 6 | - |
| **SHA-1** | Спецсимволы, цифры, буквы | 6 | - |
| **SHA-512** | Буквенная | 1 | 52 мин. |
| **SHA-512** | **Спецсимволы** | 1 | 1 ч. |
| **SHA-512** | Спецсимволы, цифры, буквы | 1 | 3 ч. |
| **SHA-512** | Буквенная | 2 | 23 ч. |
| **SHA-512** | **Спецсимволы** | 2 | 1 д. |
| **SHA-512** | Спецсимволы, цифры, буквы | 2 | 12 д. |
| **SHA-512** | Буквенная | 3 | 24 д. |
| **SHA-512** | **Спецсимволы** | 3 | 51 д. |
| **SHA-512** | Спецсимволы, цифры, буквы | 3 | 3 г. |
| **SHA-512** | Буквенная | 4 | 2 г. |
| **SHA-512** | **Спецсимволы** | 4 | 6 лет |
| **SHA-512** | Спецсимволы, цифры, буквы | 4 | - |
| **SHA-512** | Буквенная | 5 | 47 лет |
| **SHA-512** | **Спецсимволы** | 5 | 153 г. |
| **SHA-512** | Спецсимволы, цифры, буквы | 5 | - |
| **SHA-512** | Буквенная | 6 | - |
| **SHA-512** | **Спецсимволы** | 6 | - |
| **SHA-512** | Спецсимволы, цифры, буквы | 6 | - |

На основании таблицы можно сделать несколько выводов. Во-первых, увеличение длины соли заметно повышает стойкость к взлому, поскольку при большей длине соли увеличивается число возможных комбинаций, что усложняет и замедляет перебор. Например, если для хеша **MD5** с буквенными символами длиной 1 время взлома составляет несколько минут, то при длине 5 на это потребуется уже несколько лет.

Во-вторых, выбор алгоритма хеширования оказывает значительное влияние на сложность взлома. Более современные алгоритмы, такие как **SHA-512**, значительно повышают стойкость к атаке по сравнению с **MD5** и **SHA-1**, особенно при увеличении длины соли и использовании сложных комбинаций символов. Так, для **SHA-512** с длиной соли 5 и спецсимволами время взлома составит 153 года, в то время как для аналогичных условий с **MD5** — примерно 15 лет.

Наконец, тип символов в строке также влияет на сложность взлома. Хеши, содержащие только буквенные символы, поддаются взлому быстрее, чем комбинации из спецсимволов, цифр и букв, поскольку использование расширенного набора символов увеличивает число возможных комбинаций. Например, строка из спецсимволов длиной 3 для **SHA-1** взламывается за 9 дней, а та же строка, включающая буквы и цифры, — за 226 дней.

# Вопрос о минимальном количестве телефонов

В ходе эксперимента было установлено, что минимально число известных телефонов для 100%-ого взлома датасета равно трём, так как при двух известных телефонах у нас находится несколько вариантов соли: 27301097, 28872298, 19934617, 26341582, 27690959, 28015769, 28460703.

# Вывод

В ходе лабораторной работы было проведено исследование хеш-функций с использованием соли для расшифровки телефонных номеров. Сначала изучили особенности шифрования и написали программу для деобезличивания данных. Программа была протестирована на заданном датасете, а также на нескольких других хеш-функциях, включая **MD5**, **SHA-1** и **SHA-512**, что позволило сравнить их эффективность.

Анализ показал, что скорость расшифровки зависит от вида соли, её длины и выбранной хеш-функции. Например, использование более длинной соли значительно увеличивает время, необходимое для взлома. Процесс деобезличивания, в свою очередь, продемонстрировал, как хеширование может защитить личные данные, но также выявил уязвимости, при которых знание хотя бы нескольких телефонных номеров из датасета может существенно облегчить восстановление оригинальной информации.

# Источники

* Редактор блок-схем.

[*https://programforyou.ru/block-diagram-redactor*](https://programforyou.ru/block-diagram-redactor)

*дата обращения: (23.10.2024)*

* **os** — Работа с файловой системой.

[*https://docs.python.org/3/library/os.html*](https://docs.python.org/3/library/os.html)

*дата обращения: (19.10.2024)*

* tkinter — Библиотека для создания графических интерфейсов в Python.

[*https://docs.python.org/3/library/tkinter.html*](https://docs.python.org/3/library/tkinter.html%20)

*дата обращения: (19.10.2024)*

* **hashlib** — создание хешей различных алгоритмов.

[*https://docs.python.org/3/library/hashlib.html*](https://docs.python.org/3/library/hashlib.html)

*дата обращения: (19.10.2024)*

* **pandas** — анализ и манипуляция данными.

[*https://pandas.pydata.org/*](https://pandas.pydata.org/)

*дата обращения: (19.10.2024)*