**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: 1**

**Студент гр. 23Б15-пу**

**Черевко М.Е.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

Оглавление

1. [Цель работы 3](#_Toc182511904)
2. [Описание задачи (формализация задачи) 4](#_Toc182511905)

[**Анализ особенностей шифрования телефонных номеров** 4](#_Toc182511906)

1. [Теоретическая часть 5](#_Toc182511907)

[**Техника Brute Force для взлома** 5](#_Toc182511908)

[**Использование соли** 6](#_Toc182511909)

[**Хеширование телефонных номеров** 6](#_Toc182511910)

1. [Основные шаги программы 7](#_Toc182511911)
2. [Блок схема программы 9](#_Toc182511912)
3. [Описание программы 12](#_Toc182511913)
4. [Рекомендации пользователя 14](#_Toc182511914)
5. [Рекомендации программиста 15](#_Toc182511915)
6. [Контрольный пример 16](#_Toc182511916)
7. [От чего зависит время расшифровки? Анализ. 19](#_Toc182511917)
8. [Практический анализ 22](#_Toc182511918)
9. [Сколько нужно знать телефонов из предложенного датасета, чтоб 100% его взломать? 24](#_Toc182511919)
10. [Вывод 25](#_Toc182511920)
11. [Источники 26](#_Toc182511921)

# Цель работы

Провести исследование особенностей и производительности различных хеш-функций при попытках деобезличивания данных с учетом ряда вводных условий. Основная цель эксперимента — расшифровка набора данных, который был зашифрован с использованием хеш-функции и модификатора ввода, известного как "соль". Необходимо проанализировать, как на скорость и эффективность расшифровки влияют различные параметры: длина и тип соли, а также вид выбранной хеш-функции (например, MD5, SHA-1, SHA-256, SHA-512).

Дополнительно, требуется оценить, какое минимальное количество известных телефонных номеров нужно для успешного взлома всего набора данных в заданных условиях. Это поможет определить зависимость успешности расшифровки от объема известной информации в зашифрованном массиве данных.

План исследования включает выбор нескольких различных хеш-функций и экспериментирование с разными длинами соли, что позволит выявить оптимальные параметры для расшифровки в условиях, приближенных к реальным сценариям. Особое внимание также будет уделено времени выполнения операций расшифровки и влиянию каждого параметра на общую производительность алгоритма, что особенно важно для анализа больших наборов данных.

# Описание задачи (формализация задачи)

## **Анализ особенностей шифрования телефонных номеров**

1. **Анализ методов и алгоритмов шифрования телефонных номеров**  
   Провести детальное исследование современных подходов и алгоритмов, применяемых для шифрования телефонных номеров с использованием хеш-функций и добавлением соли. Изучить, как различные хеш-алгоритмы и виды соли могут улучшить защиту данных и повысить уникальность хеш-значений при различных исходных данных.
2. **Создание программы для расшифровки набора данных**  
   Разработать программу, использующую хеш-функции для шифрования телефонных номеров. Программа должна поддерживать применение различных типов соли и предоставлять возможности для анализа их влияния на процесс и скорость расшифровки данных. Этот этап позволит исследовать, как изменения параметров шифрования влияют на стойкость к попыткам восстановления данных.
3. **Проведение тестирования с использованием различных хеш-функций**  
   Программа должна быть протестирована на основе, как минимум, трех различных хеш-функций, которые представляют разные классы алгоритмов. Сравнение этих функций позволит выявить, как каждая из них влияет на скорость расшифровки и оценить их устойчивость к различным видам атак.
4. **Изучение влияния разных факторов на скорость расшифровки**  
   Провести анализ влияния различных типов соли (числовые, буквенные, комбинированные), их длины, а также выбранной хеш-функции на скорость расшифровки. Определить минимальное количество известных телефонных номеров, которое нужно иметь, чтобы гарантировать успешную расшифровку остальной части набора данных.

# Теоретическая часть

**Хеш-функции** представляют собой алгоритмы, которые преобразуют данные произвольной длины в строку фиксированной длины, называемую хеш-значением или хеш-кодом. Они широко используются в криптографии для проверки целостности данных, а также в аутентификации и хранении паролей. Ключевая идея хеширования заключается в том, что при сравнении хеш-значений можно проверить целостность данных, не имея доступа к исходным данным.

Одним из важнейших свойств хеш-функций является уникальность: для любых двух различных входных значений не должно получаться одинаковое хеш-значение. Отсутствие коллизий — ситуации, когда две разные входные строки имеют одинаковое хеш-значение — является важным критерием качества хеш-функции. Наиболее распространёнными примерами хеш-функций являются MD5, SHA-1, SHA-256 и SHA-512.

* **MD5**: ранее широко применялся, однако в настоящее время считается небезопасным из-за возможности возникновения коллизий.
* **SHA-1**: также устарел и уязвим к атакам на коллизии, что делает его непригодным для современных приложений.
* **SHA-256**: часть семейства SHA-2, он обеспечивает высокую степень безопасности и рекомендуется для использования в современных приложениях.
* **SHA-512**: также принадлежит к SHA-2 и использует более длинное хеш-значение (512 бит), что повышает его устойчивость к атакам и делает его особенно подходящим для защиты паролей и цифровых подписей.

## **Техника Brute Force для взлома**

Метод Brute Force предполагает полный перебор всех возможных комбинаций значений для нахождения правильного ключа или пароля. Этот метод не требует информации о структуре данных и часто используется для расшифровки хеш-значений. Хотя Brute Force является достаточно медленным, он остаётся эффективным при недостаточной защите данных.

При применении к хеш-функциям, метод Brute Force используется для нахождения исходных данных, соответствующих заданному хеш-значению. Если для защиты данных не используется соль или другие меры безопасности, злоумышленник может быстро проверить множество вариантов и найти совпадение с исходным значением.

## **Использование соли**

Соль представляет собой случайное значение, добавляемое к данным перед их хешированием. Применение соли делает одинаковые значения уникальными, что значительно затрудняет атаки на основе радужных таблиц и подбор паролей. Даже при одинаковых данных, уникальная соль гарантирует различные хеш-значения, что делает метод Brute Force более сложным и трудоёмким.

Добавление соли усиливает безопасность данных, поскольку каждый уникальный набор данных получает свою собственную соль. Соль предотвращает подготовленные атаки с использованием радужных таблиц, так как злоумышленнику приходится тестировать все возможные значения соли, чтобы найти совпадение с искомым хеш-значением.

## **Хеширование телефонных номеров**

При шифровании телефонных номеров с использованием хеш-функций важно учитывать особенности обработки этих данных. В рамках данной лабораторной работы телефонные номера хешируются с применением соли, что усложняет анализ хеш-значений и предотвращает несанкционированный доступ.

Использование хеш-функций и соли позволяет защитить телефонные номера, анализируя влияние таких параметров, как длина соли и выбор конкретной хеш-функции, на скорость расшифровки.

# Основные шаги программы

Основные шаги программы для шифрования и расшифровки телефонных номеров:

1. Выбор файлов для шифрования и расшифровки:
   * Программа позволяет пользователю выбрать Excel-файлы, содержащие данные для шифрования или расшифровки.
   * При выборе файла для расшифровки его путь сохраняется, и активируется кнопка для начала процесса расшифровки.
   * При выборе файла для шифрования программа отображает его имя и активирует кнопки выбора алгоритма хеширования: MD5, SHA-1, SHA-256, SHA-512. Пользователь может выбрать любой из этих алгоритмов для последующего использования в процессе шифрования.
2. Шифрование телефонных номеров:
   * Шифрование выполняется функцией encryptPhoneNumbers, которая применяет выбранный алгоритм хеширования к каждому телефонному номеру в выбранном файле.
   * Сначала программа считывает данные из Excel-файла и извлекает список телефонных номеров.
   * Затем для каждого телефонного номера генерируется соль. Соль может быть случайной строкой заданной длины или конкретным значением (в зависимости от выбранного типа соли), что усложняет процесс расшифровки и добавляет дополнительный уровень защиты.
   * Телефонные номера объединяются с солью, и к каждому полученному значению применяется выбранная хеш-функция. Хеш-код каждого номера сохраняется вместе с исходным номером, алгоритмом хеширования и командой hashcat, которая потребуется для расшифровки.
   * Все зашифрованные номера, вместе с солью и командой для hashcat, записываются в новый Excel-файл, чтобы результаты шифрования можно было легко просмотреть и использовать для дальнейших шагов.
3. Запуск команды hashcat для расшифровки:
   * Процесс расшифровки начинается с выполнения команды hashcat, предназначенной для подбора исходных значений телефонных номеров на основе их хеш-кодов.
   * Функция executeHashcat запускает hashcat, используя файл с хешами, и применяет маску (например, одиннадцатизначные числовые комбинации). Результаты подбора сохраняются в текстовый файл для последующей обработки.
   * Маска указывает hashcat, какие варианты подбора следует использовать, исходя из известных характеристик данных (например, что номера состоят из 11 цифр). Подбор выполняется методом Brute Force, перебирая возможные значения и пытаясь найти исходные номера, соответствующие хешам.
4. Поиск возможных значений соли:
   * Функция applySaltForDecryption загружает список расшифрованных номеров из файла, созданного hashcat, и проверяет, какие из них могут соответствовать известным значениям в исходном файле.
   * Затем программа определяет возможные значения соли, которые могли быть использованы при шифровании. Этот процесс включает попытку вычесть значения известных номеров из расшифрованных, чтобы выявить значение соли.
   * Если программа обнаруживает только одно возможное значение соли, оно считается истинным, и программа может использовать его для корректного восстановления всех телефонных номеров.
5. Расшифровка и сохранение результатов:
   * Если для расшифровки подходит только одно значение соли, программа пересчитывает зашифрованные номера, используя найденную соль. Каждый номер декодируется и сохраняется в Excel-файле вместе с хеш-кодом, исходным номером и значением соли.
   * В случае, если найдено несколько значений соли, точное расшифрование становится невозможным, так как существует неоднозначность. В этом случае программа сохраняет список возможных солей в отдельном Excel-файле и выводит сообщение о том, что корректная расшифровка невозможна.
   * Программа также форматирует таблицы в созданных Excel-файлах, задавая ширину колонок и выделяя значение соли для удобного просмотра и дальнейшего использования.
6. Графический интерфейс:
   * Программа построена на основе библиотеки tkinter, предоставляя пользователю удобный интерфейс для взаимодействия.
   * Основное окно интерфейса позволяет пользователю выбирать файлы для шифрования или расшифровки, выбирать алгоритм хеширования и запускать процессы шифрования и расшифровки.
   * При выборе файлов и нажатии кнопок программа показывает информационные сообщения о статусе выполнения действий. Сообщения об ошибках отображаются при отсутствии необходимых файлов или неверном ходе работы, а успешные действия завершаются сообщениями о сохранении результатов.
   * Благодаря интерфейсу, работа с программой становится удобной даже для пользователей без навыков программирования, так как все действия выполняются через кнопки и визуальные элементы.

# Блок схема программы

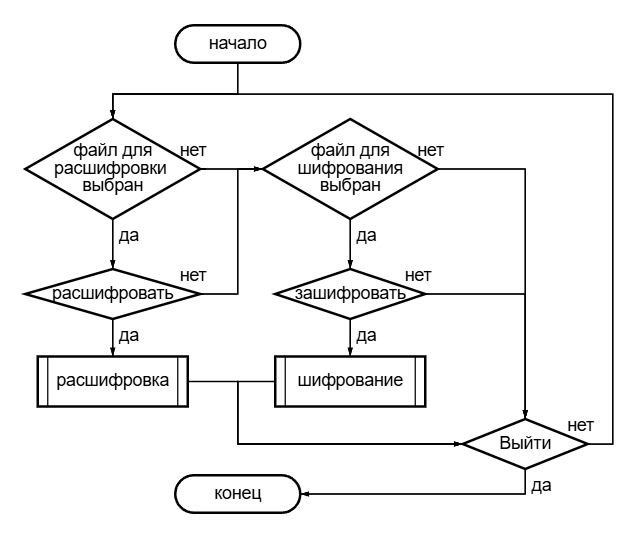


Рис 1. Блок-схема основной программы

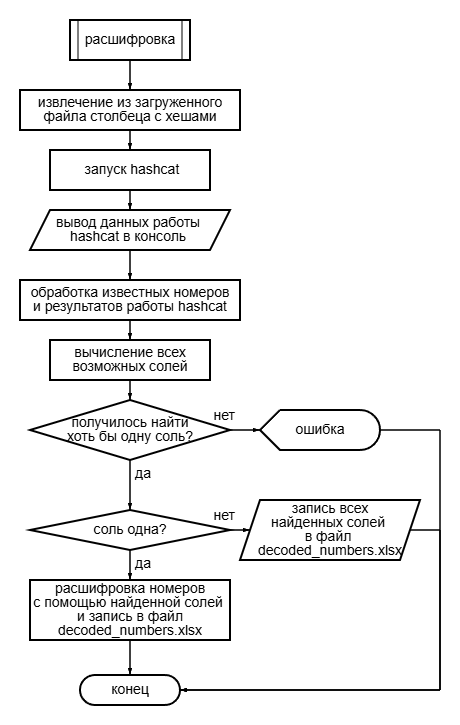


Рис 1. Блок-схема подпрограммы «расшифровка»

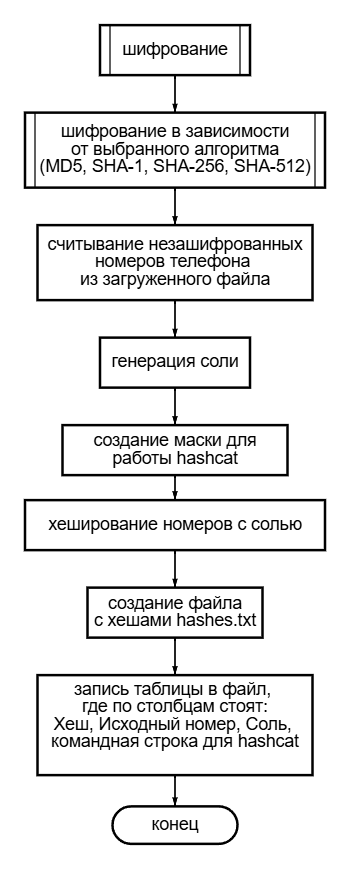


Рис 3. Блок-схема подпрограммы «шифрование»

# Описание программы

Программа реализована на языке Python версии 3.12.7 с применением следующих библиотек: **os, string, subprocess, hashlib, random, pandas, tkinter** и **openpyxl**. Вся логика сосредоточена в одном модуле, ориентированном на деобезличивание и шифрование файлов. В ходе разработки было создано 15 функций, каждая из которых выполняет строго определенную задачу:

Таблица 1. Decryption\_and\_Encryption.py

|  |  |  |
| --- | --- | --- |
| Функция | Описание | Возвращаемое значение |
| chooseFileForDecryption | Открывает диалог для выбора файла для расшифровки и активирует кнопку "Расшифровать". | None |
| chooseFileForEncryption | Открывает диалог для выбора файла для шифрования и активирует кнопки для выбора хэш-функций. | None |
| retrieveHashData | Считывает хэш-данные из Excel-файла, сохраняет их в файл и возвращает количество хэшей и путь к файлу. | (int, str) |
| executeHashcat | Запускает hashcat для извлечения паролей из указанного файла и возвращает путь к выходному файлу. | str |
| findSalts | Находит возможные значения соли на основе известных номеров и расшифрованных значений. | list |
| applySaltForDecryption | |  | | --- | |  |  |  | | --- | | Применяет соль для расшифровки данных, сохраняет результат и выводит информацию пользователю | | None |
| initiateDecryptionProcess | Запускает процесс расшифровки, вызывает функции для запуска Hashcat и применения соли. | None |
| generateSalt | |  | | --- | |  |  |  | | --- | |  |   Генерирует случайную соль заданной длины. | str |
| encryptPhoneNumbers | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Шифрует номера телефонов с использованием выбранной хэш-функции и соли, сохраняет результаты в Excel-файл. | | | None |
| |  | | --- | | hash\_MD5 |  |  |  | | --- | --- | |  |  | | |  | | --- | |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | | |  | | --- | |  |  |  |  |  | | --- | --- | --- | | |  | | --- | |  |  |  | | --- | | Вызывает encryptPhoneNumbers с использованием хэш-функции MD5. | | | | None |
| hash\_SHA1 | |  | | --- | |  |  |  | | --- | | Вызывает encryptPhoneNumbers с использованием хэш-функции SHA-1. | | None |
| hash\_SHA256 | |  | | --- | |  |  |  | | --- | | Вызывает encryptPhoneNumbers с использованием хэш-функции SHA-256. | | None |
| hash\_SHA512 | |  | | --- | |  |  |  | | --- | | Вызывает encryptPhoneNumbers с использованием хэш-функции SHA-512. | | None |
| setWindow | |  | | --- | |  |  |  | | --- | | Центрирует окно приложения на экране. | | None |
| mainloop (неявно в root.mainloop()) | Запускает главный цикл обработки событий для интерфейса. | None |

# Рекомендации пользователя

Для корректного выполнения программы следуйте приведённым шагам:

1. **Выбор файла для Расшифровки**:
   * Нажмите кнопку **"Выбрать файл для расшифровки"**. Откроется диалоговое окно, в котором нужно выбрать файл Excel с телефонными номерами, которые необходимо деобезличить.
   * После выбора файла его имя отобразится в интерфейсе.
2. **Расшифровка**:
   * После выбора файла для **расшифровки** нажмите кнопку **" Расшифровать"**. Это запустит процесс деобезличивания, который будет использовать выбранный файл и производить необходимые вычисления. Результаты будут сохранены в файл decoded\_numbers.xlsx.
3. **Выбор файла для зашифровки**:
   * Нажмите кнопку **"Выбрать файл для зашифровки"**. В этом диалоговом окне выберите файл, который является результатом работы **расшифровки** (то есть, файл decoded\_numbers.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-256, SHA-512) станут активными.
4. **Хеширование телефонных номеров**:
   * Выберите один из методов хеширования, нажав соответствующую кнопку:
     + **MD5**
     + **SHA-1**
     + **SHA-256**
     + **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 расшифровки и хеширования телефонных номеров.

# Рекомендации программиста

Чтобы программа работала корректно, выполните следующие шаги:

1. **Установите необходимые библиотеки:**
   * Убедитесь, что в системе установлены следующие библиотеки Python (для установки используйте pip install **название**):
     + **openpyxl** — для работы с файлами Excel;
     + **hashlib** — встроенная библиотека Python для хеширования (устанавливать не нужно);
     + **tkinter** — стандартная библиотека для создания графического интерфейса (входит в стандартную библиотеку Python);
     + **pandas** — для работы с данными и таблицами.
2. **Установите Hashcat:**
   * Загрузите последнюю версию Hashcat с официального сайта.
   * Выполните установку, следуя инструкциям для вашей операционной системы (Windows, Linux или macOS).
   * Проверьте, что вы можете запускать Hashcat из командной строки.
3. **Проверьте наличие необходимых файлов:**
   * Убедитесь, что файл **decoded\_numbers.xlsx** (результат расшифровки) создаётся в процессе работы программы. Если файл не создаётся на этапе деобезличивания, программа не сможет продолжить работу.
   * При использовании собственных файлов для тестирования проверьте, что они имеют правильный формат и содержат ожидаемые данные.

Следуя этим рекомендациям, вы обеспечите корректную работу программы для расшифровки и хеширования телефонных номеров.

**Код программы:**

<https://github.com/FraaaM/Decryption_and_Encryption>

# Контрольный пример

1. Запуск программы: для запуска программы используйте файл **decryption\_and\_encryption.py**.Появится графический интерфейс (Рис. 4), в котором можно будет выбрать кнопку «Выбрать файл для расшифровки» или «Выбрать файл для шифрования».

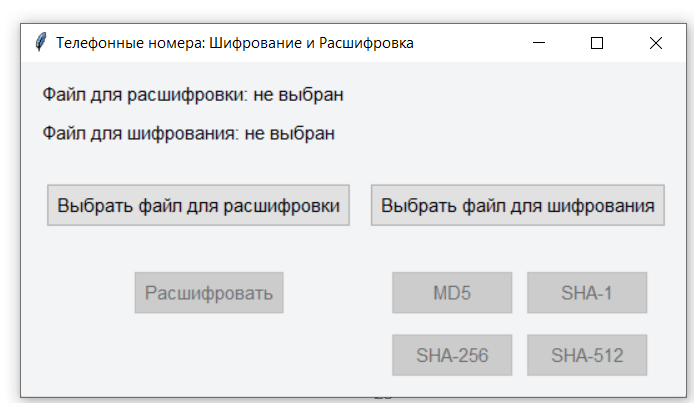


Рис 4. Графический интерфейс программы

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

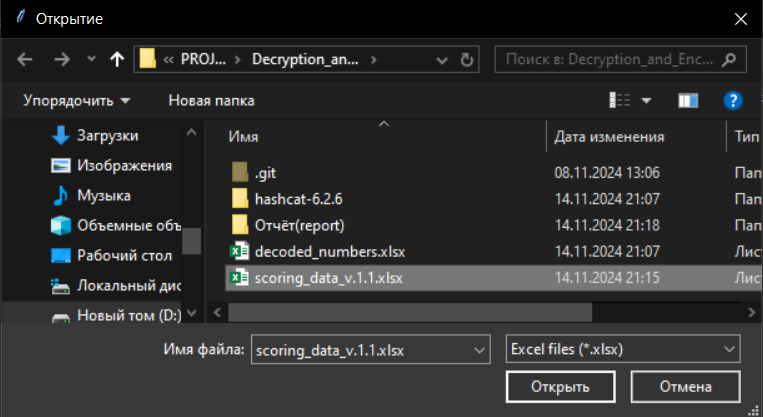


Рис 5. Выбор файла для расшифровки

1. После выбора файла для расшифровки станет доступна кнопка: «Расшифровать» (Рис.6). Нажмите и ожидайте (может занять длительное время). На выходе создаётся файл: decoded\_numbers.xlsx.

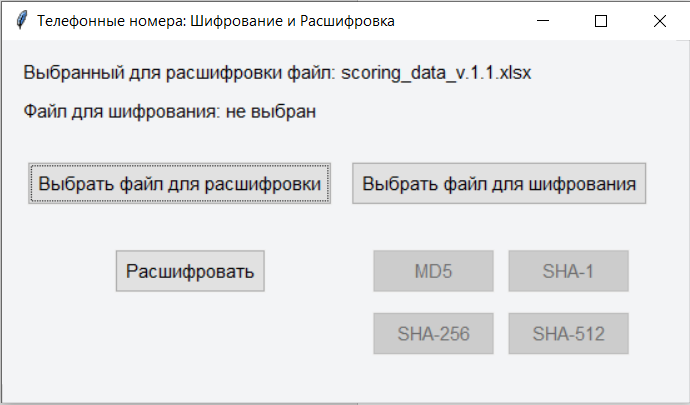


Рис 6. Разблокировка кнопки: «Расшифровать»

1. Зашифровка: сначала загрузите файл, нажав на кнопку «Выбрать файл для шифрования» и выберите decoded\_numbers.xlsx (Рис.7). Теперь, будут разблокированы кнопки: “MD5”, “SHA-1”, “SHA-256”, “SHA-512” (Рис. 8)

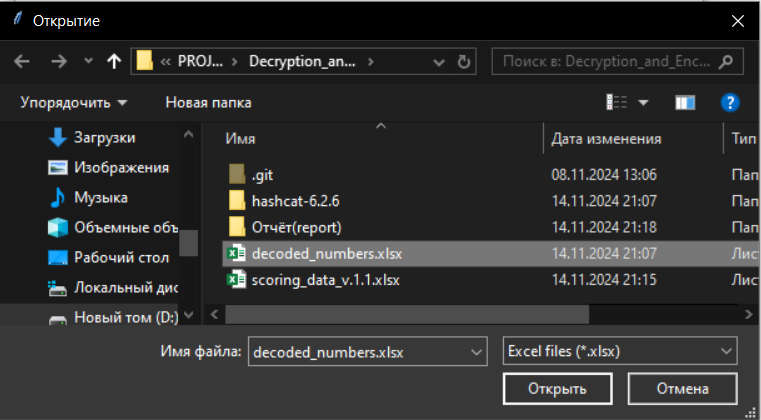


Рис 7. Выбор файла для зашифровки

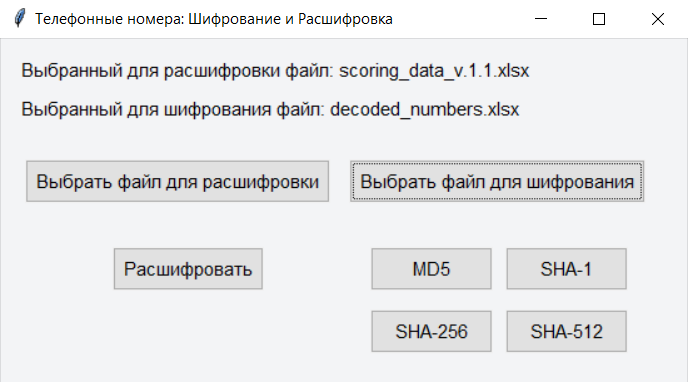


Рис 8. Разблокировка кнопок

Осталось выбрать нужную зашифровку и ожидать завершения программы. (Рис. 9)

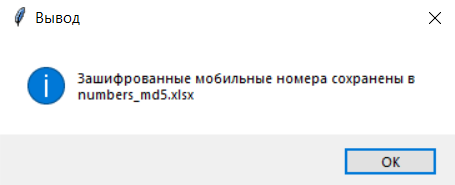


Рис 9. Завершение программы

1. 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, найти его можно в файле: numbers\_###.xlsx):  
   

Выведется статус работы, где можно будет посмотреть сколько времени займёт расшифровка. (Рис. 10)

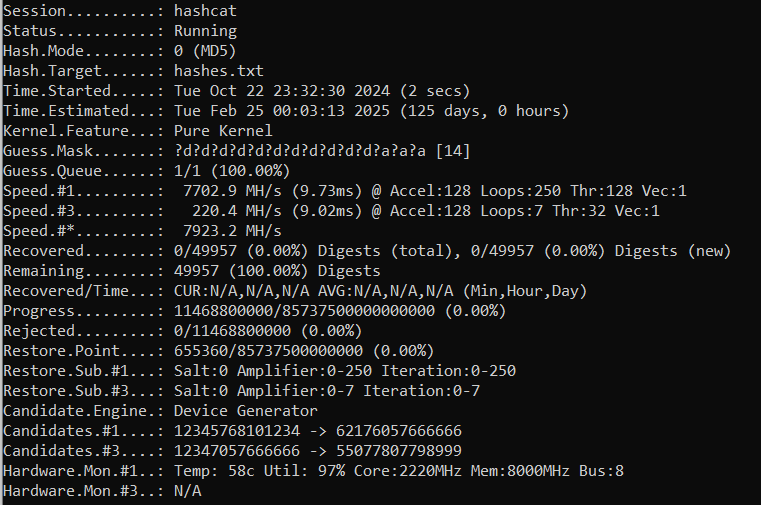


Рис 10. Статус работы hashcat

# От чего зависит время расшифровки? Анализ.

Скорость расшифровки данных, особенно при попытке взлома хешей, определяется рядом факторов. Ключевые из них включают тип соли, её длину и тип используемой хеш-функции. Каждый из этих параметров существенно влияет на время и вычислительные ресурсы, необходимые для успешного взлома данных. Рассмотрим, как именно они воздействуют на процесс и какие нюансы существуют в каждом случае.

1. **Тип соли**

Соль представляет собой случайно сгенерированную строку, добавляемую к исходным данным перед их хешированием. Главная функция соли — предотвратить использование радужных таблиц (предвычисленных таблиц хешей), которые значительно упрощают процесс взлома за счёт заранее подготовленных пар значений и их хешей. Тип соли определяет, насколько сложным окажется перебор вариантов, необходимый для взлома. Различные типы соли включают:

* **Цифровая соль (только цифры)**  
  Цифровая соль состоит из цифр (от 0 до 9). Такой тип соли относительно прост для перебора, так как на каждую позицию приходится всего 10 возможных значений. Следовательно, общее число комбинаций будет значительно меньше по сравнению с другими типами соли, что ускоряет процесс расшифровки при атаке.
* **Соль из букв**  
  В этом случае соль составляется из букв латинского алфавита, включая как строчные, так и прописные буквы. На каждую позицию приходится 52 возможных символа. Это увеличивает общее число комбинаций, которое злоумышленнику нужно проверить, замедляя процесс перебора по сравнению с цифровой солью.
* **Алфавитно-цифровая соль (цифры + буквы)**  
  Алфавитно-цифровая соль сочетает цифры и буквы. В результате, на каждую позицию может приходиться 62 символа (10 цифр + 52 буквы), что еще больше усложняет подбор. Этот тип соли заметно увеличивает время, необходимое для перебора всех возможных комбинаций.
* **Соль из всего набора символов (буквы, цифры, спецсимволы)**  
  Если для создания соли используются цифры, буквы и специальные символы, то на каждую позицию приходится до 94 возможных вариантов. Такой тип соли даёт максимальную защиту, поскольку общее число возможных комбинаций увеличивается в разы. Атака на хеш, содержащий такую соль, требует гораздо больше времени и ресурсов.

1. **Длина соли**

Длина соли также играет решающую роль в защите данных. Чем длиннее соль, тем больше уникальных комбинаций потребуется проверить при атаке, что значительно увеличивает вычислительную сложность процесса взлома.

Например:

* **Короткие соли** (1-2 символа) имеют ограниченное количество возможных комбинаций, что делает их относительно простыми для перебора.
* **Средние соли** (3-5 символов) значительно увеличивают количество возможных вариантов и делают взлом более трудоемким.
* **Длинные соли** (6 символов и более) практически исключают возможность эффективного перебора всех комбинаций в разумные сроки, даже при использовании быстрых хеш-функций.

Таким образом, увеличение длины соли прямо пропорционально увеличивает защиту данных, так как каждый символ добавляет дополнительный набор возможных комбинаций.

1. **Тип хеш-функции**

Немаловажным фактором является выбор самой хеш-функции. Разные хеш-функции имеют различную вычислительную сложность и требуемые ресурсы для их выполнения. В общем, хеш-функции можно разделить на быстрые и медленные:

* **Быстрые хеш-функции (например, MD5 и SHA-1)**  
  Эти функции разрабатывались для быстрого хеширования, и хотя они эффективно выполняют эту задачу, они не обеспечивают высокого уровня защиты от атак перебором. Злоумышленник может сгенерировать и проверить большое количество хешей за короткий промежуток времени, что делает такие функции уязвимыми к взлому, особенно если соли отсутствуют или они просты.
* **Медленные хеш-функции (например, bcrypt, scrypt или Argon2)**  
  Эти функции разработаны для создания дополнительных препятствий при попытках взлома. Они требуют больших ресурсов (времени и памяти) на каждый цикл, что делает их более защищёнными от атак перебором. Такие функции особенно полезны, когда нужно защитить данные от атак даже в случаях, если соль короткая или простая.

Медленные хеш-функции часто применяют дополнительные механизмы, например, многократное повторение вычислений хеша, что значительно увеличивает время на взлом и делает такую атаку практически бесполезной для злоумышленника, особенно при наличии сложной и длинной соли.

1. **Взаимодействие факторов**

Эти факторы (тип соли, длина соли и тип хеш-функции) взаимосвязаны и усиливают друг друга. Например, даже при использовании простой хеш-функции (например, MD5), длинная и сложная соль способна существенно усложнить взлом. Однако наилучшие результаты достигаются, когда все три фактора работают совместно: используется сложная и длинная соль вместе с медленной хеш-функцией.

Таким образом, для обеспечения максимальной защиты данных, рекомендуется использовать длинные и разнообразные соли в сочетании с медленными хеш-функциями. Эти меры значительно усложняют процесс подбора данных и делают возможные атаки дорогостоящими и маловероятными.

# Практический анализ

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 | 9 мин. |
| **MD5** | **Спецсимволы** | 1 | 15 мин. |
| **MD5** | Спецсимволы, цифры, буквы | 1 | 1ч. |
| **MD5** | Буквенная | 2 | 5 ч. |
| **MD5** | **Спецсимволы** | 2 | 7 ч. |
| **MD5** | Спецсимволы, цифры, буквы | 2 | 3 д. |
| **MD5** | Буквенная | 3 | 5 д. |
| **MD5** | **Спецсимволы** | 3 | 12 д. |
| **MD5** | Спецсимволы, цифры, буквы | 3 | 354 д. |
| **MD5** | Буквенная | 4 | 3 г. |
| **MD5** | **Спецсимволы** | 4 | 10 л. |
| **MD5** | Спецсимволы, цифры, буквы | 4 | - |
| **MD5** | Буквенная | 5 | 9 г. |
| **MD5** | **Спецсимволы** | 5 | 32 г. |
| **MD5** | Спецсимволы, цифры, буквы | 5 | - |
| **MD5** | Буквенная | 6 | - |
| **MD5** | **Спецсимволы** | 6 | - |
| **MD5** | Спецсимволы, цифры, буквы | 6 | - |
| **SHA-1** | Буквенная | 1 | 30 мин. |
| **SHA-1** | **Спецсимволы** | 1 | 54 мин. |
| **SHA-1** | Спецсимволы, цифры, буквы | 1 | 2 ч. |
| **SHA-1** | Буквенная | 2 | 12 ч. |
| **SHA-1** | **Спецсимволы** | 2 | 20 ч. |
| **SHA-1** | Спецсимволы, цифры, буквы | 2 | 5 д. |
| **SHA-1** | Буквенная | 3 | 12 д. |
| **SHA-1** | **Спецсимволы** | 3 | 25 д. |
| **SHA-1** | Спецсимволы, цифры, буквы | 3 | 1 г. |
| **SHA-1** | Буквенная | 4 | 4 г. |
| **SHA-1** | **Спецсимволы** | 4 | 13 л. |
| **SHA-1** | Спецсимволы, цифры, буквы | 4 | - |
| **SHA-1** | Буквенная | 5 | 20 лет |
| **SHA-1** | **Спецсимволы** | 5 | 45 лет |
| **SHA-1** | Спецсимволы, цифры, буквы | 5 | - |
| **SHA-1** | Буквенная | 6 | - |
| **SHA-1** | **Спецсимволы** | 6 | - |
| **SHA-1** | Спецсимволы, цифры, буквы | 6 | - |
| **SHA-256** | Буквенная | 1 | 1 ч. |
| **SHA-256** | **Спецсимволы** | 1 | 3 ч. |
| **SHA-256** | Спецсимволы, цифры, буквы | 1 | 6 ч. |
| **SHA-256** | Буквенная | 2 | 2 д. |
| **SHA-256** | **Спецсимволы** | 2 | 4 д. |
| **SHA-256** | Спецсимволы, цифры, буквы | 2 | 11д. |
| **SHA-256** | Буквенная | 3 | 47 д. |
| **SHA-256** | **Спецсимволы** | 3 | 112 д. |
| **SHA-256** | Спецсимволы, цифры, буквы | 3 | 4 г. |
| **SHA-256** | Буквенная | 4 | 10 л. |
| **SHA-256** | **Спецсимволы** | 4 | 22 г. |
| **SHA-256** | Спецсимволы, цифры, буквы | 4 | - |
| **SHA-256** | Буквенная | 5 | 74 г. |
| **SHA-256** | **Спецсимволы** | 5 | 144 л. |
| **SHA-256** | Спецсимволы, цифры, буквы | 5 | - |
| **SHA-256** | Буквенная | 6 | - |
| **SHA-256** | **Спецсимволы** | 6 | - |
| **SHA-256** | Спецсимволы, цифры, буквы | 6 | - |
| **SHA-512** | Буквенная | 1 | 4 ч. |
| **SHA-512** | **Спецсимволы** | 1 | 5 ч. |
| **SHA-512** | Спецсимволы, цифры, буквы | 1 | 8 ч. |
| **SHA-512** | Буквенная | 2 | 3 д. |
| **SHA-512** | **Спецсимволы** | 2 | 7 д. |
| **SHA-512** | Спецсимволы, цифры, буквы | 2 | 27 д. |
| **SHA-512** | Буквенная | 3 | 94 д. |
| **SHA-512** | **Спецсимволы** | 3 | 257 д. |
| **SHA-512** | Спецсимволы, цифры, буквы | 3 | 7 л. |
| **SHA-512** | Буквенная | 4 | 17 л. |
| **SHA-512** | **Спецсимволы** | 4 | 32 г. |
| **SHA-512** | Спецсимволы, цифры, буквы | 4 | - |
| **SHA-512** | Буквенная | 5 | 152 г. |
| **SHA-512** | **Спецсимволы** | 5 | - |
| **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.**

1. **Прежде всего, увеличение длины соли значительно повышает стойкость к взлому. Чем длиннее соль, тем больше уникальных комбинаций требуется перебрать злоумышленнику, что делает процесс взлома более длительным и ресурсоёмким. Например, для хеша MD5 с солью из одного буквенного символа взлом занимает всего несколько минут, но при увеличении длины соли до пяти символов потребуются уже несколько лет.**
2. **Во-вторых, выбор алгоритма хеширования играет ключевую роль в защите данных. Современные алгоритмы, такие как SHA-256 и SHA-512, обладают повышенной стойкостью к взлому по сравнению с устаревшими алгоритмами, например MD5 и SHA-1. Так, для SHA-256 — более продвинутого алгоритма из семейства SHA-2 — взлом даже с короткой солью потребует значительно больше времени по сравнению с MD5 или SHA-1. Алгоритм SHA-256, например, показывает высокую стойкость при использовании соли длиной 5 и спецсимволов: в этом случае взлом займёт около 144 года, в то время как для тех же условий с MD5 — примерно 32 лет.**
3. **Наконец, тип символов в соли также оказывает значительное влияние на сложность взлома. Если соль содержит только буквенные символы, время взлома будет меньше, чем при использовании комбинированной строки, состоящей из букв, цифр и специальных символов, так как расширенный набор символов увеличивает количество возможных комбинаций. Например, строка из спецсимволов длиной 3 для SHA-1 взламывается за 25 дней, но если строка включает буквы и цифры, то время взлома возрастает до 1 года.**

**Таким образом, сочетание увеличенной длины соли, сложного набора символов и современных хеш-функций, таких как SHA-256 или SHA-512, обеспечивает максимальную защиту данных от взлома.**

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

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

# Вывод

В рамках лабораторной работы проведено исследование хеш-функций с использованием соли для восстановления зашифрованных телефонных номеров. Вначале были изучены особенности шифрования, после чего разработана и протестирована программа для расшифровки данных. Программа проверялась как на заданных данных, так и на различных хеш-функциях, включая MD5, SHA-1, SHA-256 и SHA-512, что позволило оценить их эффективность и устойчивость к атакам.

Анализ показал, что на скорость расшифровки влияют тип соли, её длина и выбранный алгоритм хеширования. Например, более длинная соль существенно увеличивает время, необходимое для взлома. Проведённые тесты продемонстрировали, как хеширование способно защитить личные данные, но также показали, что знание нескольких телефонных номеров из набора данных может облегчить восстановление исходной информации, указывая на возможные уязвимости в зависимости от особенностей применённого алгоритма и длины соли.

# Источники

* Редактор блок-схем.

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

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

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

* 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)

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

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

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

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