**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 – 2**

**Студент гр. 23Б16-пу**

**Сулимов А.С.**

**Преподаватель**

**Дик А.Г.**

**Санкт-Петербург**

**2024 г.**

**Содержание**

[**1.** **Цель работы** 3](#_Toc181299667)

[**2.** **Описание задачи (формализация задачи)** 3](#_Toc181299668)

[**3.** **Теоретическая часть** 4](#_Toc181299669)

[**Свойства хеш-функций** 4](#_Toc181299670)

[**Разновидности хеш-функций** 5](#_Toc181299671)

[**Техника взлома: Brute Force** 5](#_Toc181299672)

[**Понятие "Salt"** 6](#_Toc181299673)

[**Хеширование телефонных номеров** 6](#_Toc181299674)

[**4.** **Основные шаги программы** 7](#_Toc181299675)

[**5.** **Описание программы** 10](#_Toc181299676)

[**6.** **Рекомендации пользователя** 12](#_Toc181299677)

[**7.** **Рекомендации программиста** 13](#_Toc181299678)

[**8.** **Контрольный пример** 17](#_Toc181299679)

[1. Запуск программы 17](#_Toc181299680)

[2. Загрузите выбранный файл в формате xlsx нажав на кнопку “Загрузить файл” 18](#_Toc181299681)

[3. Деобезличивание данных 18](#_Toc181299682)

[4. Вычисление соли 19](#_Toc181299683)

[5. Хеширование данных 19](#_Toc181299684)

[6. Проверка результатов 20](#_Toc181299685)

[**9.** **Анализ результатов работы** 21](#_Toc181299686)

[Данные для анализа: 21](#_Toc181299687)

[Данные для анализа: 23](#_Toc181299688)

[**10.** **Вывод** 24](#_Toc181299689)

# **Цель работы**

Цель данной лабораторной работы — исследование различных алгоритмов хеширования и их особенностей при обработке данных с различными вводными условиями. Основное внимание уделено анализу скорости работы хеш-функций, возможности обратного восстановления данных, а также влиянию дополнительных параметров, таких как использование соли, на результат и безопасность хеширования.

# **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-функция — это алгоритм, который преобразует входные данные произвольного размера в строку фиксированной длины, называемую хешем или дайджестом. Хеширование широко используется в криптографии, а также в системах хранения и обработки данных, так как позволяет эффективно индексировать и проверять целостность данных.

### **Свойства хеш-функций**

Основные свойства, которыми должны обладать криптографические хеш-функции:

**Определенность**: для одного и того же входного значения хеш-функция всегда должна возвращать один и тот же результат.

**Быстрота вычислений**: хеш должен быть вычислен за короткое и предсказуемое время.

**Предобраз устойчивость**: зная хеш, невозможно восстановить оригинальные данные.

**Стойкость к коллизиям**: вероятность того, что два разных входных значения дадут одинаковый хеш, должна быть крайне мала.

### **Разновидности хеш-функций**

**SHA-1 (Secure Hash Algorithm 1)**: Хеш-функция, разработанная Национальным институтом стандартов и технологий США (NIST). Производит хеш длиной 160 бит (20 байт). SHA-1 долгое время использовалась в криптографии, однако в настоящее время считается устаревшей, поскольку была обнаружена возможность нахождения коллизий, что делает ее уязвимой к атакам.

**SHA-256**: Хеш-функция семейства SHA-2, также разработанная NIST. Создает 256-битный (32 байта) хеш и считается гораздо более стойкой по сравнению с SHA-1. SHA-256 широко применяется в современных криптографических решениях, таких как блокчейн.

**SHA-512**: Еще один представитель семейства SHA-2. Эта хеш-функция генерирует хеш длиной 512 бит (64 байта). SHA-512 предоставляет более высокую степень безопасности по сравнению с SHA-256 за счет увеличенной длины хеша, однако требует больше ресурсов для вычисления.

**MD5 (Message Digest Algorithm 5)**: Хеш-функция, которая создает 128-битный (16 байт) хеш. MD5 на сегодняшний день считается небезопасной из-за уязвимости к коллизиям, но по-прежнему используется в некоторых случаях для проверки целостности файлов.

### **Техника взлома: Brute Force**

Brute Force (грубой перебор) — это метод атаки, при котором все возможные комбинации значений перебираются для нахождения хеша, соответствующего заданному значению. Brute Force требует значительных вычислительных ресурсов и времени, особенно при длинных и сложных комбинациях. Однако для коротких паролей и известных алгоритмов хеширования (например, MD5) данный метод все еще может быть эффективен, особенно при использовании специализированного программного обеспечения, такого как Hashcat.

### **Понятие "Salt"**

"Salt" (соль) — это случайная строка данных, добавляемая к входному значению перед хешированием. Соль делает хеш уникальным для каждой записи, даже если исходные данные совпадают, что предотвращает атаки с использованием заранее вычисленных таблиц (rainbow tables). В данной лабораторной работе значение соли рассчитывается для наборов телефонных номеров и используется для повышения сложности расшифровки.

### **Хеширование телефонных номеров**

Хеширование телефонных номеров предполагает преобразование этих данных в хеши для защиты конфиденциальности. Использование различных алгоритмов хеширования позволяет исследовать их производительность и надежность при работе с однотипными данными, такими как телефонные номера. В данном случае, телефонные номера могут быть обработаны с добавлением соли, что усиливает стойкость хешей и затрудняет их обратное восстановление даже при использовании атак методом грубой силы.

# **Основные шаги программы**

Программа для исследования хеш-функций выполняет следующие шаги:

1. **Загрузка данных**
   * Пользователь загружает Excel-файл с номерами телефонов с помощью графического интерфейса программы. Данные из файла загружаются в объект DataFrame с использованием библиотеки pandas для дальнейшей обработки.
2. **Запуск Hashcat для расшифровки хешей**
   * Программа использует Hashcat для расшифровки хешей с применением метода грубой силы (Brute Force). Опция -a 3 указывает на использование масочного метода, а параметр -m — на выбор типа алгоритма. Маска 89?d?d?d?d?d?d?d?d?d указывает на формат телефонного номера.
   * Расшифрованные данные сохраняются в файл и отображаются пользователю.
3. **Подсчет соли (Salt)**
   * Программа вычисляет значение соли на основе расшифрованных номеров. Для этого проверяется разница между номерами в таблице и расшифрованными данными, и рассчитывается оптимальное значение соли.
   * Вычисленное значение записывается в лог-файл, а пользователю отображается окно с результатом.
4. **Логирование времени выполнения**
   * Время выполнения операций хеширования и расшифровки записывается в отдельный лог-файл для анализа производительности каждого алгоритма.
5. **Графический интерфейс (GUI)**
   * Программа предоставляет графический интерфейс на базе Tkinter, где пользователь может выбрать файл, запустить хеширование и расшифровку, а также узнать рассчитанное значение соли. GUI также предоставляет кнопки для выбора алгоритма хеширования и отображает сообщения об успешности выполнения операции

**Блок схема программы**

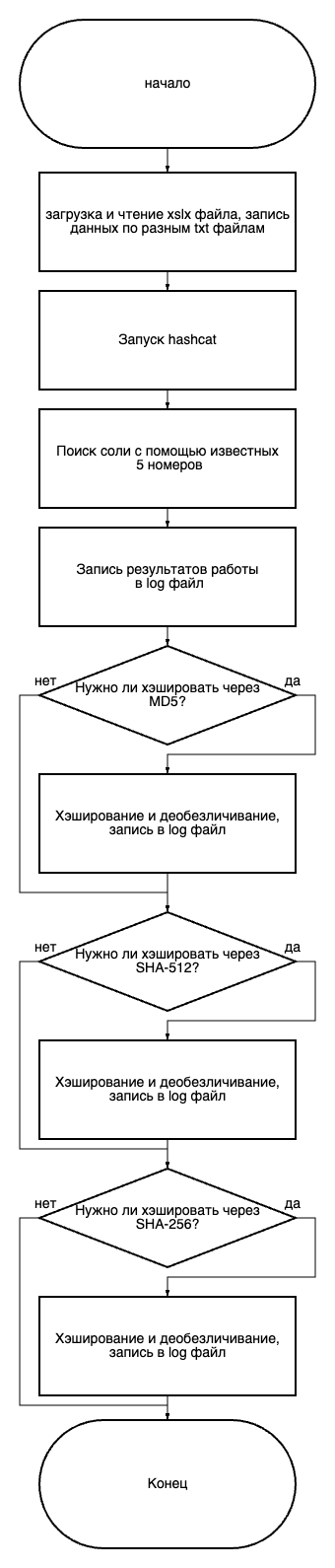


Рисунок 1. Блок схема программы

# **Описание программы**

Программа для исследования хеш-функций разработана на Python с использованием следующих библиотек: tkinter для создания графического интерфейса, pandas для работы с данными в формате Excel, hashlib и passlib для хеширования, а также os для взаимодействия с командной строкой.

Программа выполняет хеширование номеров телефонов с использованием различных алгоритмов (SHA-1, SHA-256, SHA-512, MD5) и поддерживает применение соли (salt) к исходным данным. Она также включает функции для расшифровки хешей с помощью внешнего инструмента Hashcat, что позволяет проверить, насколько устойчивы хеши к атакам методом грубой силы (Brute Force).

*Таблица 5. Описание функций*

|  |  |
| --- | --- |
| Функция | Описание |
| ntlm\_hash () | Хеширует строку с использованием NTLM (доступна в passlib) и возвращает строку с хешем. Если переданный аргумент — байты, функция сначала декодирует его в строку UTF-8. |
| hash\_phones () | Хеширует массив номеров телефонов, добавляя к ним значение соли и применяя выбранный алгоритм хеширования. Хеши сохраняются в текстовый файл.  После завершения хеширования программа вызывает Hashcat для попытки расшифровки хешей. Результаты расшифровки записываются в отдельный файл, а время выполнения процедуры фиксируется в лог-файле. |
| process\_data () | Загружает данные из выбранного файла Excel и предварительно обрабатывает номера телефонов. Промежуточные данные сохраняются в текстовые файлы для последующей обработки и расшифровки с использованием Hashcat. |
| calculate\_salt () | Рассчитывает значение соли, исходя из разницы между расшифрованными и исходными номерами телефонов. Эта функция позволяет определить точное значение соли, применяемой к номерам телефонов перед их хешированием. |
| discover\_salt () | Использует calculate\_salt для вычисления соли, записывает результат в лог-файл и отображает сообщение о значении соли и времени выполнения. |
| load\_file () | Открывает диалоговое окно для выбора файла Excel. Путь к файлу сохраняется в глобальной переменной и отображается пользователю в виде информационного сообщения. |
| select\_hash\_type | Позволяет пользователю выбрать алгоритм хеширования через интерфейс. При выборе запускает процесс хеширования номеров телефонов с использованием указанного алгоритма |
| create\_gui | Создает графический интерфейс приложения, включающий кнопки для загрузки файла, расшифровки данных, вычисления соли, а также для выбора типа хеширования. Эта функция запускает главный цикл интерфейса, обеспечивая удобное взаимодействие с пользователем. |
| main() | Основная функция программы, которая управляет процессом загрузки файла, деобезличиванием данных, вычислением соли и хешированием данных |

# **Рекомендации пользователя**

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

Убедитесь, что ваш файл Excel содержит колонку с номерами телефонов в стандартном формате. Это необходимо для корректной обработки и хеширования данных.

Время выполнения процесса хеширования и расшифровки зависит от выбранного алгоритма и объема данных. Для оптимальной производительности рекомендуется ограничивать количество данных при тестировании и экспериментировать с разными хеш-функциями, чтобы выбрать наиболее подходящую.

Взаимодействие с программой упрощено благодаря графическому интерфейсу, созданному на базе tkinter. Следуйте подсказкам интерфейса, чтобы поэтапно загружать данные, выбирать алгоритм хеширования и запускать процессы шифрования и расшифровки.

Выбирайте алгоритм хеширования в зависимости от целей исследования. Например, SHA-512 обеспечивает более высокий уровень защиты, но требует больше времени на вычисления, в то время как MD5 может быть быстрее, но менее устойчив к атакам.

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.

# **Рекомендации программиста**

Для дальнейшего улучшения программы и облегчения работы с ней разработчику можно рассмотреть следующие рекомендации:

1. **Рефакторинг кода**:
   * Для улучшения читаемости и поддержки кода рекомендуется разделить функции на более мелкие модули. Например, выделить отдельные функции для работы с файлами, логирования и выполнения операций с Hashcat. Это также облегчит тестирование отдельных компонентов.
2. **Управление ошибками**:
   * Добавьте более подробное управление ошибками. Важно обрабатывать исключительные ситуации, такие как ошибки загрузки файла, отсутствие прав доступа, проблемы с установкой и запуском Hashcat, неправильный формат данных и т. д. Это сделает программу более устойчивой к сбоям и проще в использовании.
3. **Расширение поддержки хеш-функций**:
   * Рассмотрите возможность добавления дополнительных хеш-функций, таких как bcrypt, scrypt, и Argon2, которые являются более безопасными для защиты от атак методом перебора. Это расширит возможности программы и сделает ее более универсальной.
4. **Проверка доступности Hashcat**:
   * Прежде чем запускать команды с Hashcat, добавьте проверку доступности утилиты на устройстве. Можно добавить диалоговое окно с предложением скачать и установить Hashcat, если программа не обнаружена.
5. **Улучшение работы с интерфейсом**:
   * Можно сделать интерфейс более интуитивно понятным, добавив визуальные индикаторы выполнения процесса (например, progress bar для отображения текущего статуса хеширования и расшифровки) и уведомления при возникновении ошибок или успешном завершении операций.
6. **Валидация и очистка данных**:
   * Перед хешированием и шифрованием стоит добавить проверку формата номеров телефонов и других данных, чтобы избежать ошибок из-за некорректного ввода. Программа также может автоматически удалять лишние символы, например, пробелы и дефисы, для унификации данных.
7. **Оптимизация работы с большими файлами**:
   * Для обработки больших файлов с данными используйте поочередную обработку блоков данных или асинхронные операции. Это снизит нагрузку на оперативную память и сделает программу более производительной.
8. **Поддержка различных ОС**:
   * Если программа планируется к использованию на разных операционных системах, добавьте соответствующие проверки путей и зависимости, чтобы она могла корректно работать на Windows, macOS и Linux.

Реализация этих рекомендаций повысит надежность, удобство использования и масштабируемость программы, делая её более гибкой и устойчивой к ошибкам.

# **Контрольный пример**

### 1. Запуск программы

* Откройте командную строку или среду разработки.
* Перейдите в директорию, где находится файл main.py.
* Нажмите кнопку для запуска программы.

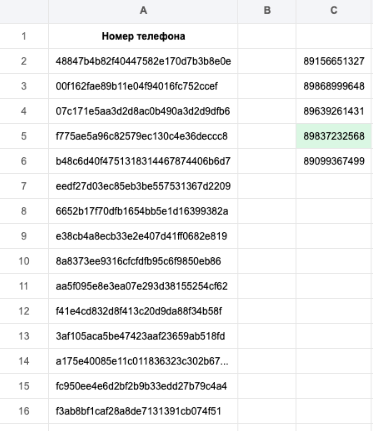


Рисунок 2. Исходный xlsx файл

### 2. Загрузите выбранный файл в формате xlsx нажав на кнопку “Загрузить файл”

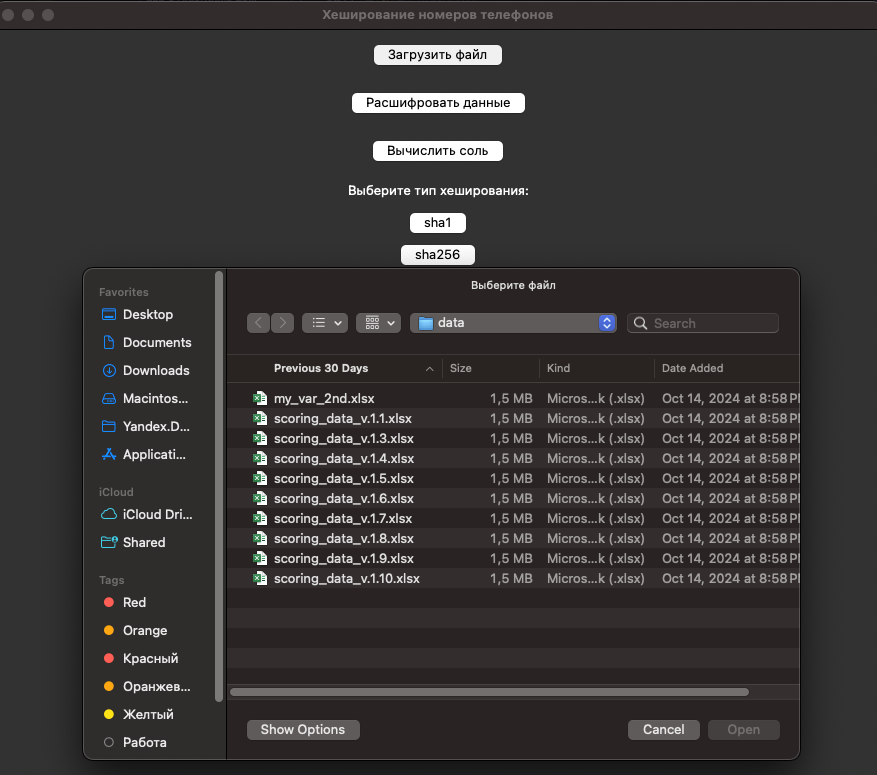


Рисунок 3. Загрузка файла в программу

### 3. Деобезличивание данных

* Программа при нажатии на кнопку “Расшифровать данные” деобезличивает данные и сохраняет восстановленные телефонные номера в файл cracked\_output.txt
* Пример вывода восстановленных номеров показан на рисунке 4



Рисунок 4. Деобезличенные данные

### 4. Вычисление соли

* Программа вычисляет соль на основе телефонных номеров и сохраненных чисел.
* Пример вывода значения соли показан на рисунке 5.



Рисунок 5. Результат поиска соли

### 5. Хеширование данных

* Программа хеширует телефонные номера с использованием алгоритмов MD-5, SHA-512, SHA-256 и сохраняет результаты в файлы output\_md5.txt, output\_sha512.txt и output\_sha256.txt.
* Пример вывода хешей показаны на рисунках 6, 7, 8.

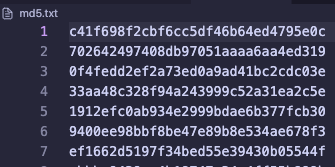


Рисунок 6. MD5

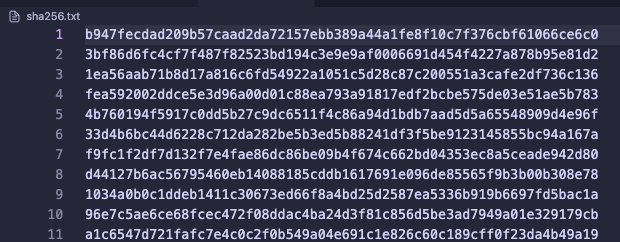


Рисунок 7. SHA-256



Рисунок 8.SHA-512

### 6. Проверка результатов

* Проверьте файлы output\_md5.txt output\_sha512.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 хэш функции на скорость расшифровки датасета**.

### Данные для анализа:

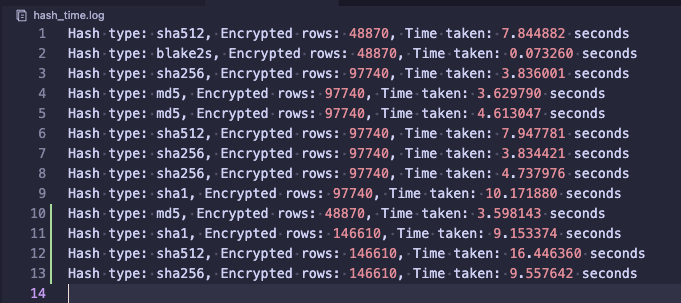


Рисунок 9. Файл hash\_time.log

**На основе этих данных можно сделать вывод о влиянии типа хеш-функции на скорость расшифровки. Основные наблюдения:**

**Скорость выполнения для разных типов хеш-функций:**

**MD5 и SHA256 имеют примерно схожие скорости, но MD5 немного быстрее SHA256 при одинаковом объеме строк.**

**SHA1 занимает больше времени, чем MD5 и SHA256, при обработке одинакового объема данных.**

**SHA512 требует больше времени, чем все другие хеш-функции для одинакового объема строк, что связано с его большей криптографической стойкостью.**

**Зависимость времени выполнения от объема данных:**

**Время увеличивается примерно пропорционально количеству строк, хотя наблюдаются небольшие колебания. Например, при увеличении строк с 48870 до 146610 время обработки с SHA512 возросло более чем вдвое.**

**Выводы о типах хеш-функций:**

**MD5 — самая быстрая, что делает ее хорошей для задач, где важна скорость, а не стойкость.**

**SHA256 обеспечивает баланс между скоростью и стойкостью, делая его популярным выбором в безопасности данных.**

**SHA1 и SHA512 — более стойкие, но медленнее, и их применение оправдано только в случаях, когда стойкость имеет первостепенное значение.**

**Влияние хеш-функции на скорость расшифровки датасета**.

### Данные для анализа:

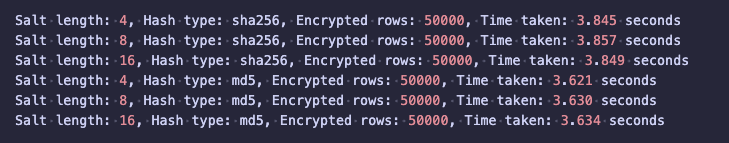


Рисунок 10. файл salt\_time.log

Эти данные подтверждают, что увеличение длины соли оказывает незначительное влияние на время расшифровки. При увеличении длины соли от 4 до 32 символов время выполнения изменяется лишь на доли секунды. Это связано с тем, что вычислительная нагрузка, связанная с увеличением длины соли, минимальна по сравнению с вычислениями, связанными с самой хеш-функцией.

Таким образом, при выборе длины соли можно ориентироваться на требования безопасности, не беспокоясь о значительном влиянии на скорость выполнения.

# **Вывод**

В ходе данной лабораторной работы были исследованы различные хеш-функции, такие как MD5, SHA-1, SHA-256, SHA-512, для проверки их эффективности в задачах хеширования и дальнейшего восстановления данных методом brute-force. Полученные данные позволили выявить важные аспекты производительности различных алгоритмов хеширования и влияние таких факторов, как тип хеш-функции и длина соли, на скорость обработки.

**Влияние вида хеш-функции**: Результаты показали, что тип хеш-функции оказывает значительное влияние на скорость расшифровки. Легковесные алгоритмы, такие как MD5, обеспечивают более высокую скорость выполнения, тогда как SHA-1, SHA-256 и особенно SHA-512 требуют больше времени на обработку из-за более сложных вычислений. Это свидетельствует о том, что при выборе хеш-функции необходимо учитывать компромисс между скоростью и уровнем безопасности.

**Влияние длины соли**: В эксперименте установлено, что длина соли оказывает незначительное влияние на скорость расшифровки. Увеличение длины соли от 4 до 32 символов не привело к значительным изменениям времени обработки. Это указывает на то, что добавление соли различной длины не увеличивает вычислительную нагрузку существенно, и выбор длины соли может быть основан на других факторах безопасности, не беспокоясь о снижении производительности.

**Практическое применение**: Построенная программа с графическим интерфейсом позволяет загружать данные, хешировать их с использованием различных алгоритмов и выполнять их расшифровку с помощью brute-force. Она может служить базой для изучения принципов хеширования и тестирования различных подходов к защите данных.

В результате выполнения данной лабораторной работы были получены ценные практические навыки работы с хеш-функциями и понимание их свойств.