**Задание**

Цель работы — реализовать алгоритм хэш-функции SHA-3, который принимает на вход файл и сохраняет хэш-значение этого файла в новом файле. Программа должна выполнять следующие действия:

1. Принимать на вход файл для хэширования.

2. Хэшировать этот файл и сохранять результат в новый файл.

**Краткие теоретические сведения**

Общие сведения о криптографических хэш-функциях

Криптографические хэш-функции используются для преобразования данных произвольной длины в строку фиксированной длины, называемую хэш-значением. Эти функции важны для обеспечения целостности и подлинности данных. Основные требования к таким функциям:

**1. Необратимость:** Невозможно восстановить исходные данные из хэш-значения.

**2. Устойчивость к коллизиям первого рода:** Невозможно найти два разных сообщения с одинаковым хэш-значением.

**3. Устойчивость к коллизиям второго рода:** Невозможно найти любые две разные строки, имеющие одинаковый хэш.

**Принципы работы криптографических хэш-функций**

Одна из ранних конструкций для создания хэш-функций — конструкция Меркла–Дамгора. Она преобразует данные произвольной длины в фиксированную длину, используя функцию сжатия, применяемую к частям данных итерационно. Современные алгоритмы, такие как SHA-3, используют конструкцию губки. Губка работает в два этапа:

**1. Впитывание (absorbing):** Входные данные разбиваются на блоки и обрабатываются функцией сжатия.

**2. Выжимание (squeezing):** Состояние после впитывания используется для генерации хэш-значения.

**Описание программной реализации**

Программа написана на языке Python с использованием стандартной библиотеки **hashlib**, которая поддерживает алгоритм SHA-3. Я использовал текстовый редактор Visual Studio Code и Python версии 3.10 для разработки программы.

**Подробное описание работы программы**

1. Функция **hash\_file**:

- Создаёт объект хэш-функции SHA3-256 с помощью **hashlib.sha3\_256()**.

- Открывает файл в бинарном режиме для чтения. Это позволяет работать с любыми типами файлов.

- Читает файл порциями по 8192 байт. Это сделано для того, чтобы программа могла работать с большими файлами, не загружая их полностью в память.

- Обновляет хэш-функцию с каждой порцией данных с помощью **hasher.update(buff)**.

- Возвращает итоговое хэш-значение в шестнадцатеричном формате с помощью **hasher.hexdigest()**.

2. Функция **save\_ file**:

- Принимает на вход хэш-значение и путь к файлу, в который нужно сохранить это значение.

- Открывает указанный файл для записи и сохраняет хэш-значение.

3. Основной блок программы:

- Определяет путь к входному файлу (который будет хэшироваться) и путь к выходному файлу (в который будет сохранено хэш-значение).

- Вызывает функцию **hash\_file** для хэширования содержимого файла input.txt.

- Сохраняет результат хэширования в файл output\_hash.txt с помощью функции **save\_ file**.

- Выводит сообщение в консоль о том, что хэш сохранён, вместе с самим хэш-значением.

4. Результаты работы программы

Программа успешно хэширует входной файл и сохраняет хэш-значение в выходной файл. Пример работы программы:

Пример:

Входной файл: **input.txt**

Выходной файл: **output\_hash.txt**

Содержимое output\_hash.txt:

**dbee0609d4ef0b0584f5e7e207c6ba0677b3665a918cd7eb1c06b9f7fc042a34**

5. Выводы

В результате выполнения работы были приобретены навыки программной реализации алгоритма криптографической хэш-функции SHA-3. Программа позволяет надёжно хэшировать файлы и сохранять результаты в новом файле, что демонстрирует практическое применение теоретических знаний в области криптографии.