Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование криптографических**

**хеш-функций**

Студент: Гвоздовский К.В.

ФИТ 3 курс 6 группа

Преподаватель:

Нистюк Ольга Александровна

1. **Цель работы**

Изучение алгоритмов хеширования и приобретение практических навыков их реализации и использования в криптографии.

1. **Задание**

1. Разработать оконное приложение, реализующее один из алгоритмов хеширования из указанного преподавателем семейства (MD или SHA; или иного). При этом можно воспользоваться доступными готовыми библиотеками. Язык программирования – на свой выбор. Приложение должно обрабатывать входные сообщения, длина которых определяется спецификацией на реализуемый алгоритм.

2. Оценить быстродействие выбранного алгоритма хеширования.

1. **Ход работы**

**Теоретическая часть**

**Математические основы асимметричных шифров**

Хеш-функция – математическая или иная функция h = H(М), которая принимает на входе строку символов М, называемую также прообразом, переменной длины n и преобразует ее в выходную строку фиксированной (обычно – меньшей) длины l.

Хеширование (или хэширование, англ. hashing) – это преобразование входного массива данных определенного типа и произвольной длины (практически) в выходную битовую строку фиксированной длины.

Хеширование – криптографический процесс, в котором большой фрагмент данных преобразуется в гораздо меньшую строку цифр и букв.

Преобразования называются хеш-функциями, или функциями свертки, а их результаты называют хешем, хеш-кодом, хеш-таблицей или дайджестом сообщения (англ. message digest).

Все существующие функции хеширования можно разделить на два больших класса:

• бесключевые хеш-функции, зависящие только от сообщения;

• хеш-функции с секретным ключом, зависящие как от сообщения, так и от секретного ключа.

Криптографическая хеш-функция – это специальный класс хеш-функций, который имеет различные свойства, необходимые для решения задач в области криптографии.

Основные задачи, решаемые с помощью хеш-функций:

• аутентификация (хранение паролей);

• проверка целостности данных;

• защита файлов;

• обнаружение зловредного ПО;

• криптовалютные технологии.

К основным свойствам хеш-функций можно отнести следующие.

Свойство 1. Детерминированность: независимо от того, сколько раз вычисляется H(M), M – const, при использовании одинакового алгоритма код хеш-преобразования h всегда должен быть одинаковым.

Свойство 2. Скорость вычисления хеша h: если процесс вычисления h недостаточно быстрый, система просто не будет эффективной.

Свойство 3. Сложность обратного вычисления: для известного H(М) невозможно (практически) определить М. Это важнейшее свойство хеш-функции для криптографических применений – свойство односторонности преобразования.

Свойство 4. Даже минимальные изменения в хешируемых данных (М ≠ М') должны изменять хеш: Н(M) ≠ Н(М').

Свойство 5. Коллизионная устойчивость (стойкость).

Зная М, трудно найти такое М' (М ≠ М'), для которого H(М) = H(М').

Если последнее равенство выполняется, то говорят о коллизии 1-го рода.

Если случайным образом выбраны два сообщения (М и М'), для которых H(М) = H(М'), говорят о коллизии 2-го рода.

Это означает, что по хеш-коду должно быть практически невозможным восстановление входной строки М.

Базовые алгоритмы обоих рассматриваемых семейств (MD и SHA) условно можно разделить на 5 стадий:

• расширение входного сообщения;

• разбивка расширенного сообщения на блоки;

• инициализация начальных констант;

• обработка сообщения поблочно (основная процедура алгоритма хеширования);

• вывод результата.

MD5 (Message-Digest Algorithm 5) – это алгоритм хеширования, разработанный Рональдом Ривестом в 1991 году. Он используется для преобразования произвольного сообщения произвольной длины в фиксированный 128-битный хеш-код.

Алгоритм MD5 состоит из четырех основных этапов:

1. Инициализация: задаются начальные значения для регистров, которые используются в процессе хеширования.
2. Предварительная обработка: сообщение дополняется битами таким образом, чтобы его длина была кратна 512 битам. Затем сообщение разбивается на блоки по 512 бит.
3. Основной цикл: каждый блок обрабатывается в цикле, состоящем из 64 итераций. В каждой итерации применяются различные преобразования к регистрам.
4. Финальная обработка: регистры объединяются в 128-битный хеш-код, который и является конечным результатом хеширования.

MD5 также считается устаревшим и небезопасным алгоритмом хеширования, так как имеет ряд уязвимостей, позволяющих произвольно изменять данные, которые приводят к одинаковым хеш-кодам. Рекомендуется использовать более современные алгоритмы хеширования, такие как SHA-256 или SHA-3.

SHA-256 (Secure Hash Algorithm 256) – это один из наиболее распространенных алгоритмов хеширования, который используется для преобразования произвольного сообщения произвольной длины в фиксированный 256-битный хеш-код.

Алгоритм SHA-256 состоит из шести основных этапов:

1. Инициализация: задаются начальные значения для регистров, которые используются в процессе хеширования.
2. Предварительная обработка: сообщение дополняется битами таким образом, чтобы его длина была кратна 512 битам. Затем сообщение разбивается на блоки по 512 бит.
3. Инициализация переменных: переменные инициализируются значениями, используя начальные значения регистров и специальные константы.
4. Основной цикл: каждый блок обрабатывается в цикле, состоящем из 64 итераций. В каждой итерации применяются различные преобразования к переменным.
5. Формирование хеш-кода: после обработки всех блоков, значения переменных объединяются в 256-битный хеш-код.
6. Завершение: переменные и регистры очищаются.

SHA-256 является криптографически стойким алгоритмом, который обеспечивает высокий уровень безопасности при хешировании данных. Он широко используется в различных областях, таких как криптография, блокчейн, безопасность информации и т.д.

SHA-256 лежит в основе биткоина. Он может взять файл любого размера, смешать его и дать данные фиксированного размера длинной 256 бит.

Когда мы пропускаем любой файл/любые данные через алгоритм SHA-256  
или другую хеш-функцию, мы никогда не узнаем, что получим на выходе. Результат будет случайный. Но несмотря на это, результат детерминистический.

То есть, если мы получаем один и тот же файл через алгоритм, то на выходе мы получаем один и тот же результат. Это означает, что имея хэш какого-либо файла, мы сможем проверить этот файл на его подлинность.

Если мы пропустим файл через SHA-256 и увидим известный нам хэш, значит это именно тот файл.

Алгоритм хеширования SHA-256 включает в себя несколько раундов, каждый из которых состоит из нескольких этапов. Один раунд SHA-256 состоит из следующих шагов:

1. Инициализация переменных. На этом этапе инициализируются переменные, которые будут использоваться в ходе раунда.
2. Подготовка расширенного сообщения. Сообщение, которое необходимо захешировать, дополняется до определенной длины и разбивается на блоки.
3. Обновление хеш-значения. На этом этапе происходит обновление текущего хеш-значения. Изначально оно инициализируется определенным значением, а затем изменяется в процессе выполнения алгоритма.
4. Обработка блоков сообщения. Каждый блок сообщения обрабатывается отдельно. Вначале происходит подготовка блока к обработке, затем блок разбивается на 16 32-битных слов, которые затем преобразуются в новые значения с помощью нескольких логических операций.
5. Обновление переменных. На этом этапе происходит обновление переменных, которые будут использоваться на следующем раунде. Они получают значения, которые были вычислены в текущем раунде.
6. Повторение. Если блоки сообщения еще остались, то происходит повторение шагов 3-5.
7. Возвращение хеш-значения. После обработки всех блоков сообщения, текущее значение хеша возвращается в качестве результата работы алгоритма.

Каждый раунд SHA-256 состоит из 64 таких шагов, которые производятся последовательно. Каждый шаг включает в себя несколько операций, таких как вращение, смещение, логические операции и сложение по модулю 2^32. Результаты этих операций объединяются в определенном порядке, что и приводит к формированию нового значения хеш-функции. Таким образом, каждый раунд алгоритма SHA-256 улучшает стойкость хеш-функции и делает ее более надежной для защиты данных.

**Практическая часть**

В качестве используемого алгоритма был выбран алгоритм MD5.

Алгоритм MD5 (Message Digest Algorithm 5) — это криптографическая хеш-функция, которая принимает на вход произвольное сообщение фиксированной длины и вычисляет 128-битный хеш-код (MD5-хеш) для этого сообщения. Алгоритм MD5 был разработан Рональдом Ривестом в 1991 году. Вот шаги, описывающие работу алгоритма MD5:

1. Исходное сообщение делится на блоки фиксированного размера (512 бит).
2. Каждый блок обрабатывается независимо. Для начала обработки блока выполняется инициализация четырех 32-битных переменных, известных как A, B, C и D.
3. Блок разбивается на 16 слов по 32 бита каждое. Каждое из этих слов обрабатывается внутри цикла.
4. Инициализируются четыре вспомогательные переменные, известные как F, G, H и I.
5. Выполняются четыре раунда, в каждом из которых слово блока преобразуется с использованием функций F, G, H и I, а также текущих значений переменных A, B, C и D.
6. После завершения всех раундов, значения переменных A, B, C и D объединяются в одно 128-битное значение, которое и представляет собой MD5-хеш исходного сообщения.
7. В конечном итоге, MD5-хеш является уникальным идентификатором исходного сообщения. Любое изменение в сообщении, даже незначительное, приводит к значительным изменениям в хеш-коде.

Для реализации данного алгоритма в приложении было использовано пространство имён System.Security.Cryptography, которое содержит поддержку наиболее распространенных симметричных (DES, 3DES, RC2, Rijndael), асимметричных (RSA, DSA) и хеш-(MD5, SHA-1, SHA-256, SHA-384, SHA- 512) алгоритмов криптографии.

Класс, представленный на рисунке 3.1 использует функции из данного пространства имён для хэширования сообщения.

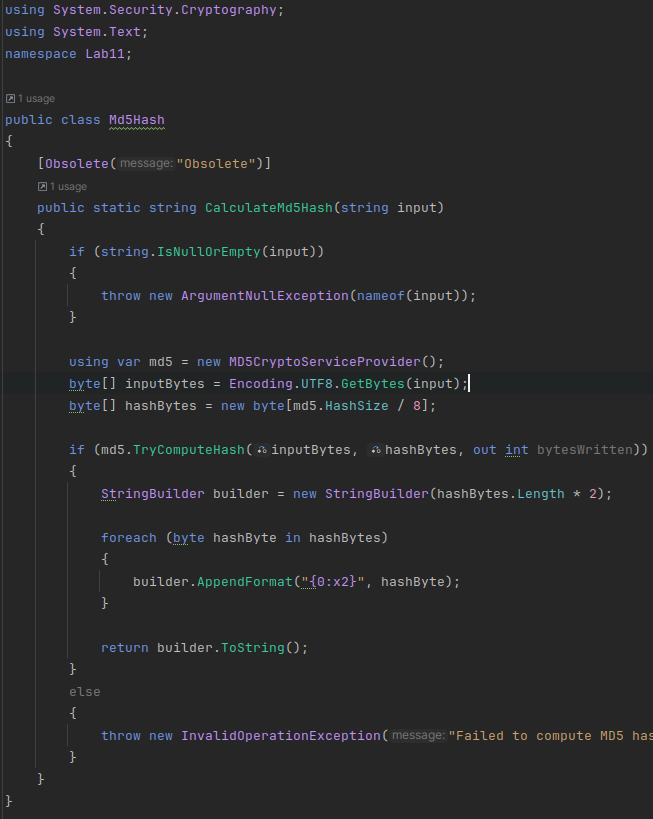


Рисунок 3.1 – Класс MD5Hash, реализующий алгоритм MD5

На следующих рисунках 3.2 – 3.3 можно увидеть пример использования класса MD5Hash и результат его работы.

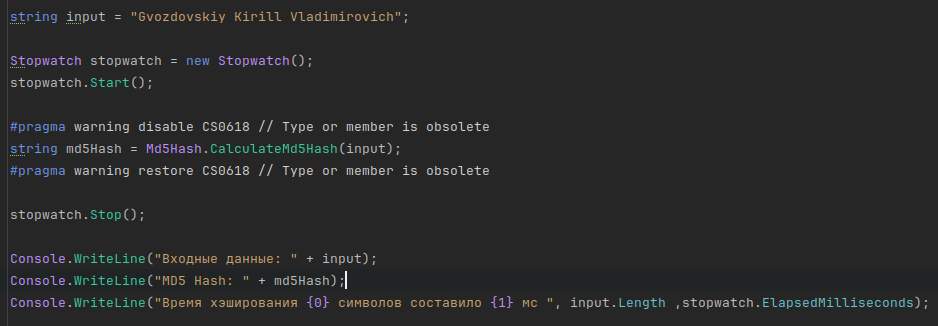


Рисунок 3.2 – Пример использования класса MD5Hash

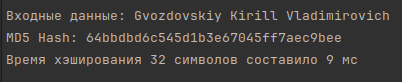


Рисунок 3.3 – Результат хэширования

Теперь увеличим длину сообщения и оценим время работы алгоритма. Результаты можно увидеть на рисунке 3.4.

Рисунок 3.4 – Скорость хэширования сообщений различной длины

**Вывод**

В ходе лабораторной работы были изучены алгоритмы хеширования и приобретены практические навыки их реализации и использования в криптографии.

Также было разработано авторское приложение в соответствии с целью лабораторной работы.