МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №11 НА ТЕМУ:**

**Исследование криптографических хеш-функций**

Выполнила студентка 3 курса 4 группы

Коржова Валерия Сергеевна

Минск 2023

**Задание:** Разработать приложение, реализующее хеширование сообщений.

**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.

**Криптографическая хеш-функция** – это специальный класс хеш-функций, который имеет различные свойства, необходимые для решения задач в области криптографии. С помощью хеш-функций решаются такие задачи как: аутентификация (хранение паролей), проверка целостности данных, защита файлов и так далее.

К основным свойствам криптографических хеш-функций можно отнести следующие:

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

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

3. ***Сложность обратного вычисления***: для известного H(М) невозможно (практически) определить М. Это важнейшее свойство хеш-функции для криптографических применений – свойство односторонности преобразования. Это означает, что по хеш-коду должно быть практически невозможным восстановление входной строки М.

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

5. Коллизионная устойчивость (стойкость). Зная М, трудно найти такое М' (М ≠ М'), для которого H(М) = H(М').

Двумя наиболее распространёнными семействами алгоритмов хеширования являются MD и SHA. Базовые алгоритмы обоих рассматриваемых семейств условно можно разделить на 5 стадий:

1. Расширение исходного сообщения
2. Разбиение входного сообщения на блоки
3. Инициализация начальных констант
4. Обработка блоков
5. Выдача результата

Для разработки приложения, позволяющего хешировать сообщения с помощью алгоритма SHA256 был использован язык программирования Python со встроенных графических модулем Tkinter. Функция хеширования введённого сообщения с записью хеша в текстовое поле:

def get\_hash():

    global output

    global message\_input

    global time\_label

    message = message\_input.get('1.0', END)

    start = time()

    result = sha256(message.encode('utf-8')).hexdigest()

    time\_label.config(text=f'Time: {time() - start}')

    output.delete('1.0', END)

    output.insert(END, result)

Листинг 1 – Функция обработки ввода приложения

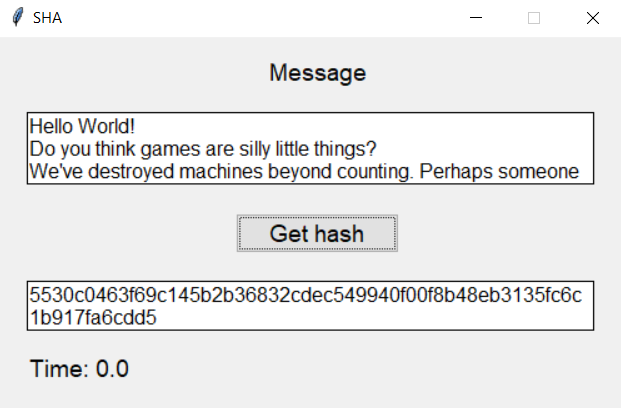


Рисунок 2 – Хеширование сообщения из 4700 символов

В нижней части окна приложение выводит время, которое было затрачено при хешировании сообщения. Как видно на рисунках 1 и 2, сообщения как малой, так и большой длины алгоритм SHA256 хеширует моментально. Также видно, что хеш имеет постоянную длину в 256 символов.

**Контрольные вопросы:**

**1. Дать определение хеш-функции.**

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*.

**2. Что такое «однонаправленность» хеш-функций и какова роль этого свойства хеш-функций в криптографии?**

Однонаправленность (или необратимость) хеш-функций – это свойство, которое означает, что вычисление хеш-значения сообщения является легкой задачей, но восстановление исходного сообщения из хеш-значения является вычислительно невозможной задачей. Самый базовый пример использования данного свойства хеш-функций – это хранение паролей в виде их хешей. Даже если взломщик получит доступ к данным паролей, он не сможет восстановить их изначальный вид.

**3. Что такое «коллизия»? Типы коллизий хеш-функций.**

Коллизией хеш-функции Н называют ситуацию, при которой различным входам (в общем случае – *х* и *у* или *М* ≠ *М'*) соответствует одинаковый хеш-код: H(*x*) = H(*y*) или 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.

**4. Сформулировать в общем виде парадокс «дней рождений».**

Парадокс «дней рождений» заключается в том, что при достаточно большой группе людей вероятность того, что у двух людей в группе день рождения совпадает, оказывается выше, чем может показаться на первый взгляд.

**5. Как парадокс «дней рождений» используется в криптографии?**

Атака на хеш-функции, основанная на парадоксе "дней рождений", называется атакой дней рождений (birthday attack). Она заключается в том, чтобы найти два разных сообщения, которые дают одинаковый хеш-код. Для этого атакующий генерирует большое количество случайных сообщений и вычисляет их хеши. При достаточно большом количестве случайных сообщений вероятность того, что какие-то два из них дадут одинаковый хеш-код, становится высокой.

**6. Сколько попыток нужно сделать, чтобы с вероятностью более 0,5 (0,7; 0,8; 0,9) обнаружить коллизию при длине хеша (*l*) 64 (128; 256; 512) битов?**

Для вычисления необходимого количества попыток, которые нужно сделать, чтобы с вероятностью *P* обнаружить коллизию при длине хеша *l* битов, используется формула:

где:

N - количество попыток, которое необходимо сделать

*l* - длина хеша в битах

*p* - вероятность обнаружения коллизии

При *l*=64:

P=0,5 : N=4,3 млрд

P=0,7 : N=5,9 млрд

P=0,8 : N=6,8 млрд

P=0,9 : N=8,2 млрд

При *l*=128:

P=0,5 : N=1,8 \* 1019

P=0,7 : N=2,5 \* 1019

P=0,8 : N=2,9 \* 1019

P=0,9 : N=3,5 \* 1019

При *l*=256:

P=0,5 : N=3,4 \* 1038

P=0,7 : N=4,7 \* 1038

P=0,8 : N=5,4 \* 1038

P=0,9 : N=6,5 \* 1038

При *l*=512:

P=0,5 : N=1,2 \* 1077

P=0,7 : N=1,6 \* 1077

P=0,8 : N=1,9 \* 1077

P=0,9 : N=2,3 \* 1077

**7. Дать общую характеристику алгоритмам хеширования семейств MD и SHA. Из каких основных стадий состоит алгоритм хеширования сообщения?**

Алгоритмы MD (Message Digest) разработаны фирмой RSA Data Security и включают в себя несколько версий: MD2, MD4, MD5. Они отличаются длиной выходного хеша и количеством раундов, выполняемых для генерации хеша.

Алгоритмы SHA (Secure Hash Algorithm) разработаны Национальным институтом стандартов и технологий США (NIST) и включают в себя несколько версий: SHA-0, SHA-1, SHA-2, SHA-3. Они также отличаются длиной выходного хеша и количеством раундов.

Базовые алгоритмы обоих рассматриваемых семейств условно можно разделить на 5 стадий:

1. Расширение исходного сообщения
2. Разбиение входного сообщения на блоки
3. Инициализация начальных констант
4. Обработка блоков
5. Выдача результата

**8. Рассчитать общую длину (L') хешируемого сообщения после предварительной стадии на основе алгоритма MD, если объем (L) исходного сообщения составлял: 0; 484; 512; 1000; 2000; 16 000 битов. Какова в каждом случае будет длина хеша?**

Для вычисления длины хешируемого сообщения после предварительной стадии необходимо использовать формулу:

*L'* = (*L* + *k* + 64) mod 512,

где *L* - длина исходного сообщения в битах, *k* - минимальное необходимое количество битов заполнения для выравнивания длины сообщения до 448 битов, 64 - количество битов, занимаемых записью длины сообщения.

Для блочного алгоритма MD с блочным размером 512 бит и длиной хеша 128 бит, *k* будет равно 448 - (*L* mod 512) если 448 - (*L* mod 512) ≥ 65, иначе *k* будет равно 960 - (*L* mod 512).

Для исходных сообщений длиной в 0, 484, 512, 1000, 2000 и 16000 битов длина хешируемого сообщения после предварительной стадии будет составлять соответственно: 512, 960, 1024, 2048, 4096 и 32768 битов (при условии, что используется алгоритм MD с блочным размером 512 бит и длиной хеша 128 бит).

**10. Представить и охарактеризовать структурную схему одного раунда алгоритмов хеширования на основе MD4; MD5; SHA-1.**

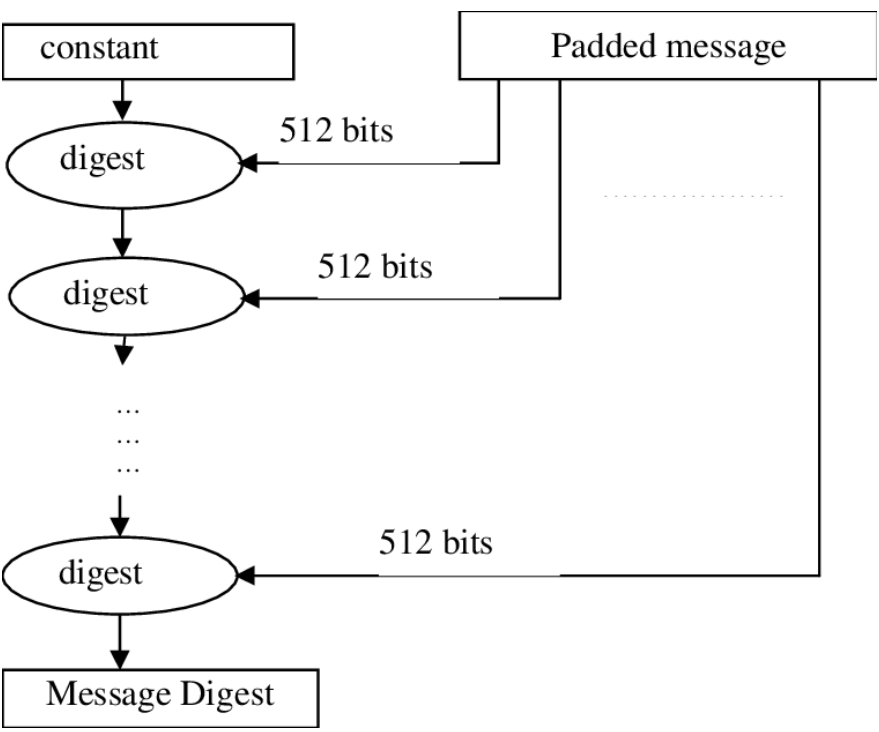


Рисунок 3 – Базовая схема алгоритмов MD-4 и MD-5

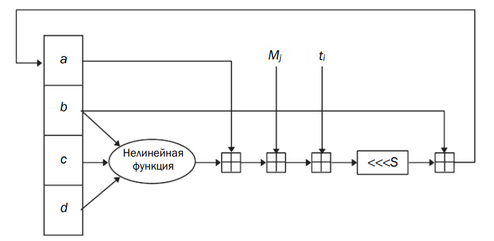


Рисунок 4 – Структурная схема одной операции алгоритма MD-5

Каждая операция MD представляет собой нелинейную функцию над тремя из *a*, *b*, *c* и *d*. Затем она добавляет этот результат к четвертой переменной, подблоку текста *Мj* и константе *ti*. Далее результат циклически сдвигается вправо на переменное число *s* битов и добавляет результат к одной из переменных *a*, *b*, *c* и *d*. Наконец, результат заменяет одну из этих переменных.

Результатом хеширования *h* является конкатенация последних значений указанных переменных.

На рисунке 5 показана схема выполнения одной операции в алгоритме SHA-1. Цикл состоит из четырех этапов по 20 операций в каждом (в MD5 – 4 этапа по 16 операций в каждом). Каждая операция представляет собой нелинейную функцию над тремя из пяти: *a*, *b*, *c*, *d*, *e*. Сдвиг и сложение – аналогично MD5.

В алгоритме используются следующие четыре константы:

*Kt* = 0x5a827999, при *t* = 0, …, 19,

*Kt* = 0x6ed9eba1, при *t* = 20, …, 39,

*Kt* = 0x8flbbcdc, при *t* = 40, …, 59,

*Kt* = 0xca62c1d6, при *t* = 60, …, 79.

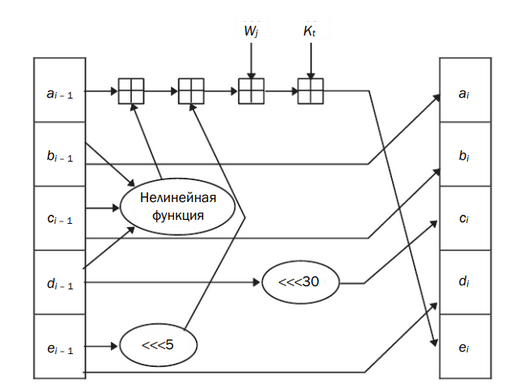


Рисунок 5 – Схема выполнения одной операции в алгоритме SHA-1

**11. На чем основан «лавинный эффект» в алгоритмах хеширования? В чем состоит цель его реализации?**

«Лавинный эффект» – это свойство алгоритмов хеширования, при котором небольшие изменения во входных данных приводят к значительным изменениям в выходном хеш-коде. Это означает, что даже незначительные изменения в сообщении приводят к значительно измененному хеш-коду.

Цель реализации «лавинного эффекта» заключается в том, чтобы обеспечить стойкость алгоритма хеширования к атакам, таким как подбор значений или вставка заранее вычисленных хеш-кодов.

**12. В чем состоят основные структурные и функциональные особенности алгоритма хеширования SHA-3?**

Основные структурные и функциональные особенности алгоритма хеширования SHA-3 (Keccak) следующие:

1. Конструкция с порталом: Keccak использует конструкцию с порталом (sponge construction), которая позволяет использовать алгоритм не только для хеширования, но и для аутентификации и шифрования данных.
2. Гибкость размера хеш-кода: Keccak позволяет создавать хеш-коды различных размеров (224, 256, 384 или 512 бит), что делает его универсальным для различных задач хеширования.
3. Использование буфера состояния: Keccak использует большой буфер состояния, который является основной работающей памятью алгоритма. Это позволяет алгоритму быстро и эффективно обрабатывать входные данные.
4. Простые и эффективные операции: Keccak использует простые и эффективные операции, такие как замены битов и перестановки. Это делает алгоритм быстрым и простым в реализации.
5. Высокий уровень безопасности: Keccak был выбран Национальным институтом стандартов и технологий (NIST) как стандарт для алгоритмов хеширования, и он считается одним из самых безопасных алгоритмов хеширования.
6. Высокая параллелизуемость: Keccak обладает высокой параллелизуемостью, что позволяет использовать его на многоядерных процессорах и графических процессорах (GPU).
7. Устойчивость к боковым каналам: Keccak разработан с учетом устойчивости к боковым каналам, что делает его подходящим для использования в криптографических устройствах, таких как умные карты и устройства аутентификации.1.

**13.** **Охарактеризуйте структурные, функциональные особенности и криптостойкость белорусского государственного стандарта хеширования (СТБ 34.101.77–2016).**

       Белорусский государственный стандарт хеширования (СТБ 34.101.77–2016), также известный как «БелТСИПС», является семейством алгоритмов хеширования, который включает в себя четыре версии: БелТСИПС 2012, БелТСИПС 2015, БелТСИПС 2016 и БелТСИПС 2020.

Основные структурные и функциональные особенности БелТСИПС:

* Размер хеша составляет 256 бит.
* Базируется на комбинации преобразований Фейстеля и замены Перестановкой-Заменой.
* Размер входного сообщения не ограничен.
* Используется несколько ключей различной длины для усиления стойкости алгоритма.
* Доступен как с общим, так и с секретным ключом.

**Вывод:** В ходе лабораторной работы было разработано оконное приложение, позволяющее хешировать входные сообщения с использованием алгоритма SHA256. Было установлено, что использованный алгоритм является крайне высокопроизводительным.