Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

**Исследование криптографических хеш-функций**

Студент: Решетилова Антонина

ФИТ 3 курс 4 группа

Минск 2023

1. **Описание приложения**

Приложение написано на языке программирования Python и позволяет:

* хешировать входное сообщение;
* оценивать скорость выполнения алгоритма.

1. **Методика выполнения поставленных задач**

SHA256 – хеш-функция из семейства алгоритмов SHA-2, которая предназначена для создания дайджестов для сообщений произвольной длины. Длина дайджеста – 256 бит. Исходное сообщение дополняется до нужной длины, а затем разбивается на блоки. Каждый блок – на 16 слов. Каждый блок сообщения пропускается через 64 итерации. На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сума – значение хеш-функции. Т.к. инициализация внутреннего состояния производится результатом обработки предыдущего блока, то нет возможности обрабатывать блоки параллельно.

Результат обработки строки с помощью алгоритма SHA256 представлен на рисунке 2.1.

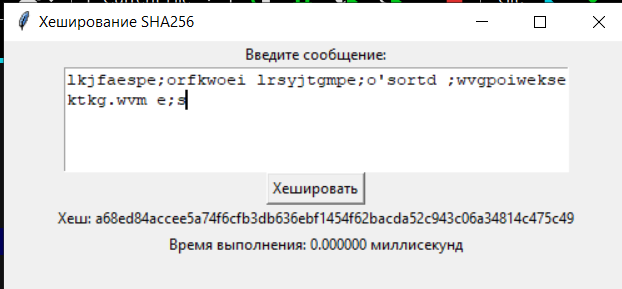


Рисунок 2.1 – Результат хеширования SHA256

Функция для хеширования строки с помощью алгоритма SHA256 представлена на рисунке 2.2.



Рисунок 2.2 – Функция для хеширования строки

В алгоритмах хеширования часто используется такое понятие как соль. Соль – строка данных, которая передается хеш-функции вместе с входными данными для вычисления хеша. Используется для усложнения определения прообраза хеш-функции методом перебора по словарю возможных значений, включая радужные атаки. Позволяет скрыть факт использования одинаковых прообразов при использовании разной соли.

Таким образом, были реализованы все поставленные задачи. Были исследованы асимметричные шифры.

**Вывод**

В ходе лабораторной работы были приобретены навыки хеширования используя алгоритм SHA256. Были изучены основные принципы работы хеширования.

Также было разработано приложение, на языке программирования Python, для реализации задач, связанных с хешированием данных.

**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).

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

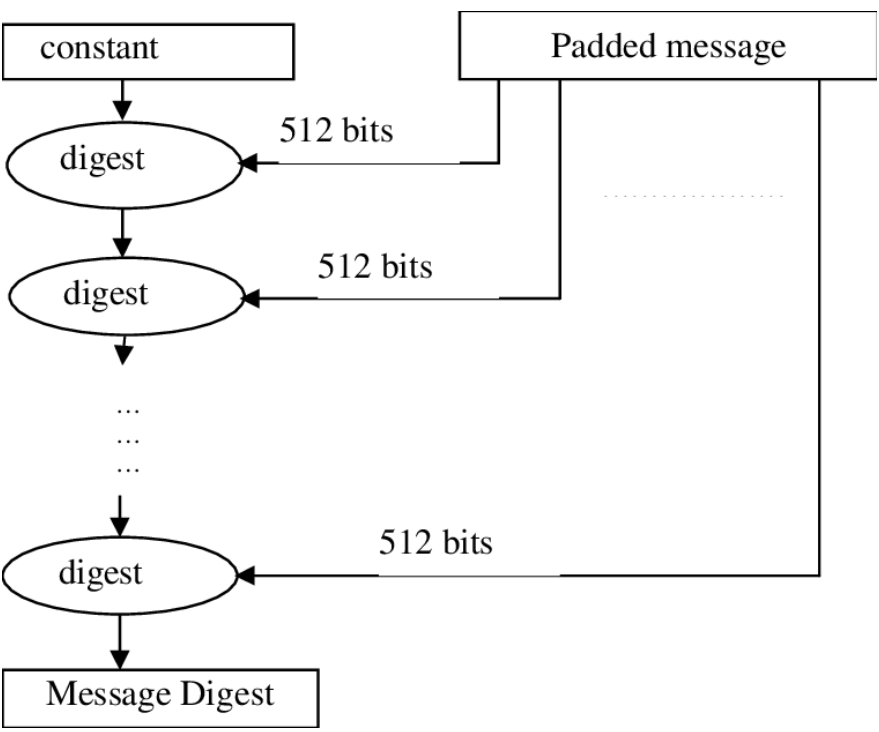


Рисунок 3 – Базовая схема алгоритмов MD-4 и MD-5

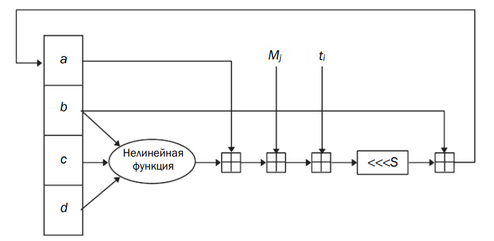


Рисунок 4 – Структурная схема одной операции алгоритма MD-5

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.

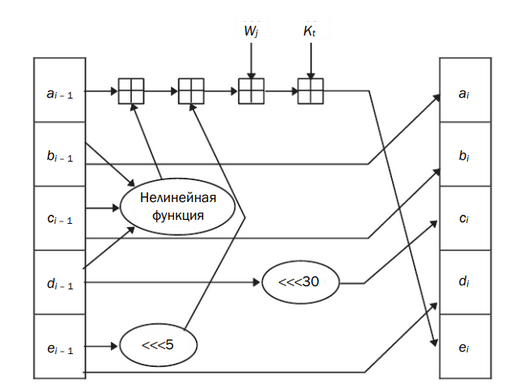


Рисунок 5 – Схема выполнения одной операции в алгоритме SHA-1

**10. 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.

**11. 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.

**12.** **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.