Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный технологический университет»

Факультет информационных технологий

Кафедра информационных систем и технологий

**Отчет к лабораторной работе**:

Исследование криптографических хеш-функций

Выполнил:

студент 3 курса 4 группы

специальности ПОИТ

Дубалеко В.В.

Минск 2020

1. **Теоретические сведения**

Определение 1. 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.

Определение 2. Хеширование (или хэширование, англ. hashing ) – это преобразование входного массива данных определенного типа и произвольной длины (практически) в выходную битовую строку фиксированной длины.

Преобразования называются хеш-функциями или функциями свертки, а их результаты называют хешем, хеш-кодом, хеш-таблицей или дайджестом сообщения (анг. message digest).Теорема 1. Основная теорема арифметики. Всякое натуральное число N, кроме 1, можно представить как произведение простых множителей:

Определение 3. 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 следющие.

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 не достаточно 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 – односторонности преобразования.

Определение 5. Хеш-функция – это функция, выполняющая отображение из множества М в число, находящееся в интервале

[0, m–1]: h: M → [0, m–1]. (x mod pi) = ai ,

Алгоритмы семейства MD-x (2/4/5/6) являются творениями Р. Ривеста; MD – Message Digest. Алгоритм MD6, в отличие от предыдущих версий алгоритма этого семейства, не стандартизован.

Алгоритмы семейства SHA (SHA – Secure Hash Algorithm) являются в настоящее время широко распространенными. По существу во многих случаях завершился переход от SHA-1 к стандартам версии SHA-2. SHA-2 – собирательное название алгоритмов SHA-224, SHA256, SHA-384 и SHA-512. SHA-224 и SHA-384 являются, по сути, аналогами SHA-256 и SHA-512 соответственно.

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.

1. **Практическая часть**

В данной лабораторной работе необходимо разработать оконное приложение, реализующее один из алгоритмов хеширования из указанного преподавателем семейства (MD или SHA; или иного).

В связи с поставленными требованиями было разработано приложение реализующее алгоритм MD5 при помощи встроенного пакета MessageDigest в языке Java. Ниже представлен листинг приложения.

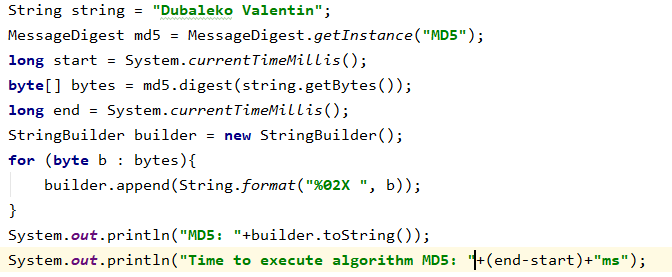


Рисунок 2.1 – Листинг использования алгоритма MD5

На вход алгоритма поступает входной поток данных, хеш которого необходимо найти. Длина сообщения измеряется в битах и может быть любой (в том числе нулевой). Запишем длину сообщения в L. Это число целое и неотрицательное. Кратность каким-либо числам необязательна. После поступления данных идёт процесс подготовки потока к вычислениям.

Сначала к концу потока дописывают единичный бит.Затем добавляют некоторое число нулевых бит такое, чтобы новая длина потока {\displaystyle L'}L стала сравнима с 448 по модулю 512, ({\displaystyle L'=512\times N+448}L’=512\*n+448). Выравнивание происходит в любом случае, даже если длина исходного потока уже сравнима с 448.

В конец сообщения дописывают 64-битное представление длины данных (количество бит в сообщении) до выравнивания. Сначала записывают младшие 4 байта, затем старшие. Если длина превосходит {\displaystyle 2^{64}-1}2^64-1, то дописывают только младшие биты (эквивалентно взятию по модулю {\displaystyle 2^{64}}2^64). После этого длина потока станет кратной 512. Вычисления будут основываться на представлении этого потока данных в виде массива слов по 512 бит (пример рисунок 2.2).

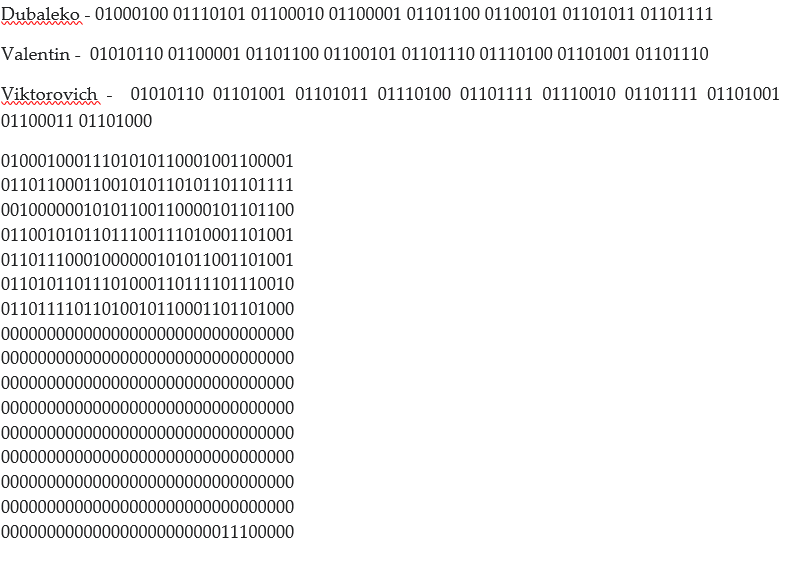


Рисунок 2.2 – Выравнивание потока и добавления длины сообщения

Инициализация буфера:

Для вычислений инициализируются 4 переменных размером по 32 бита и задаются начальные значения шестнадцатеричными числами (порядок байтов [little-endian](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%80%D1%8F%D0%B4%D0%BE%D0%BA_%D0%B1%D0%B0%D0%B9%D1%82%D0%BE%D0%B2" \o "Порядок байтов), сначала младший байт):

А = 01 23 45 67; // 67452301h

В = 89 AB CD EF; // EFCDAB89h

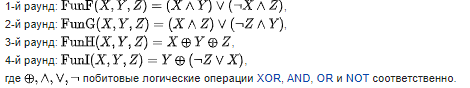
С = FE DC BA 98; // 98BADCFEh

D = 76 54 32 10. // 10325476h

В этих переменных будут храниться результаты промежуточных вычислений. Начальное состояние ABCD называется инициализирующим вектором.

Вычисления в цикле:

Для каждого раунда потребуется своя функция. Введём функции от трёх параметров — слов, результатом также будет слово:



Заносим в блок данных элемент *n* из массива 512-битных блоков. Сохраняются значения A, B, C и D, оставшиеся после операций над предыдущими блоками (или их начальные значения, если блок первый).

AA = A

BB = B

CC = C

DD = D

**Этап 1**

/\* [abcd k s i] a = b + ((a + F(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 0 7 1][DABC 1 12 2][CDAB 2 17 3][BCDA 3 22 4]

[ABCD 4 7 5][DABC 5 12 6][CDAB 6 17 7][BCDA 7 22 8]

[ABCD 8 7 9][DABC 9 12 10][CDAB 10 17 11][BCDA 11 22 12]

[ABCD 12 7 13][DABC 13 12 14][CDAB 14 17 15][BCDA 15 22 16]

Этап 2

/\* [abcd k s i] a = b + ((a + G(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 1 5 17][DABC 6 9 18][CDAB 11 14 19][BCDA 0 20 20]

[ABCD 5 5 21][DABC 10 9 22][CDAB 15 14 23][BCDA 4 20 24]

[ABCD 9 5 25][DABC 14 9 26][CDAB 3 14 27][BCDA 8 20 28]

[ABCD 13 5 29][DABC 2 9 30][CDAB 7 14 31][BCDA 12 20 32]

Этап 3

/\* [abcd k s i] a = b + ((a + H(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 5 4 33][DABC 8 11 34][CDAB 11 16 35][BCDA 14 23 36]

[ABCD 1 4 37][DABC 4 11 38][CDAB 7 16 39][BCDA 10 23 40]

[ABCD 13 4 41][DABC 0 11 42][CDAB 3 16 43][BCDA 6 23 44]

[ABCD 9 4 45][DABC 12 11 46][CDAB 15 16 47][BCDA 2 23 48]

Этап 4

/\* [abcd k s i] a = b + ((a + I(b,c,d) + X[k] + T[i]) <<< s). \*/

[ABCD 0 6 49][DABC 7 10 50][CDAB 14 15 51][BCDA 5 21 52]

[ABCD 12 6 53][DABC 3 10 54][CDAB 10 15 55][BCDA 1 21 56]

[ABCD 8 6 57][DABC 15 10 58][CDAB 6 15 59][BCDA 13 21 60]

[ABCD 4 6 61][DABC 11 10 62][CDAB 2 15 63][BCDA 9 21 64]

Суммируем с результатом предыдущего цикла:

A = AA + A

B = BB + B

C = CC + C

D = DD + D

После окончания цикла необходимо проверить, есть ли ещё блоки для вычислений. Если да, то переходим к следующему элементу массива (n + 1) и повторяем цикл.

Результат вычислений находится в буфере ABCD, это и есть хеш. Если выводить побайтово, начиная с младшего байта A и закончив старшим байтом D, то мы получим MD5-хеш. 1, 0, 15, 34, 17, 18…

Существует мнение что взломать хэш MD5 невозможно, однако это неправда, существует множество программ, подбирающих исходное слово на основе хэша. Абсолютное большинство из них осуществляет перебор по словарю, однако существуют такие методы как RainbowCrack, он основан на генерировании множества хэшей из набора символов, чтобы по получившейся базе производить поиск хэша.

Также у MD5, как у любой хэш-функции, существует такое понятие как коллизии — это получение одинаковых хэшей для разных исходных строк.

**Вывод**

В данной лабораторной работе я закрепил теоретические знания по хеш-функциям. А также, разработал приложения по алгоритму MD5.