Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Защита информации и надёжность информационных систем**

**Лабораторная работа №11**

ИССЛЕДОВАНИЕ КРИПТОГРАФИЧЕСКИХ ХЕШ-ФУНКЦИЙ

Студент: Вайсера Р.Л.

ФИТ 3 курс 4 группа

Преподаватель: Сазонова

Минск 2023

**Цель**: изучение алгоритмов хеширования и приобретение практических навыков их реализации и использования в криптографии.

**Задачи**:

1. Закрепить теоретические знания по алгебраическому описанию и алгоритмам реализации операций вычисления однонаправленных хеш-функций.

2. Освоить методику оценки криптостойкости хеш-преобразований на основе «парадокса дня рождения».

3. Разработать приложение для реализации заданного алгоритма хеширования (из семейств MD и SHA).

4. Оценить скорость вычисления кодов хеш-функций.

5. Результаты выполнения лабораторной работы оформить в виде описания разработанного приложения, методики выполнения экспериментов с использованием приложения и результатов эксперимента.

**Практическое задание**

Было разработано приложение, реализующее хеш-метод SHA-1.

Алгоритм состоит из следующих шагов:

Шаг 1: добавление недостающих битов

Сообщение добавляется таким образом, чтобы его длина была кратна 448 по модулю 512 (длина https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-TjQgHh.png448 mod 512). Добавление осуществляется всегда, даже если сообщение уже имеет нужную длину. Таким образом, число добавляемых битов находится в диапазоне от 1 до 512. Добавление состоит из единицы, за которой следует необходимое количество нулей.

Шаг 2: добавление длины

К сообщению добавляется блок из 64 битов. Этот блок трактуется как беззнаковое 64-битное целое и содержит длину исходного сообщения до добавления.

Результатом первых двух шагов является сообщение, длина которого кратна 512 битам. Расширенное сообщение может быть представлено как последовательность 512-битных блоков *Y*0, *Y*1, ..., *YL-*1, так что общая длина расширенного сообщения есть *L* × 512 бит. Таким образом, результат кратен шестнадцати 32-битным словам.

Шаг 3: инициализация SHA-1 буфера

Используется 160-битный буфер для хранения промежуточных и окончательных результатов хэш-функции. Буфер может быть представлен как пять 32-битных регистров *A*, *B*, *C*, *D* и *E*. Эти регистры инициализируются следующими шестнадцатеричными числами:

*A*=67452301; *B*=EFCDAB89; *C*=98BADCFE; *D*=10325476; *E*=C3D2E1F0

Шаг 4: обработка сообщения в 512-битных (16-словных) блоках

Основой алгоритма является модуль, состоящий из 80 циклических обработок, обозначенный как HSHA. Все 80 циклических обработок имеют одинаковую структуру.

Шаг 5: выход

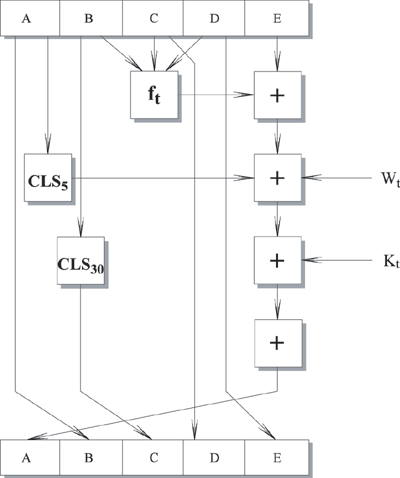
После обработки всех 512-битных блоков выходом *L*-ой стадии является 160-битный дайджест сообщения.

Рассмотрим более детально логику в каждом из 80 циклов обработки одного 512-битного блока. Каждый цикл можно представить в виде:

*A*, *B*, *C*, *D*, *E* (*CLS5* (*A*) + *ft* (*B*, *C*, *D*) + *E* + *Wt* + *Kt*), *A*, *CLS30* (*B*), *C*, *D*

Где

|  |
| --- |
| *A*, *B*, *C*, *D*, *E* - пять слов из буфера. |
| *t* - номер цикла, 0 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-mIxj6a.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-nUdWIe.png79. |
| *ft* - элементарная логическая функция. |
| *CLSs* - циклический левый сдвиг 32-битного аргумента на s битов. |
| *Wt* - 32-битное слово, полученное из текущего входного 512-битного блока. |
| *Kt* - дополнительная константа. |
| + - сложение по модулю 232. |



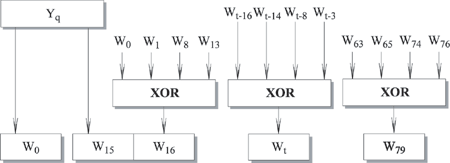
Логика выполнения отдельного цикла

Каждая элементарная функция получает на входе три 32-битных слова и создает на выходе одно 32-битное слово. Элементарная функция выполняет набор побитных логических операций, т.е. *n*-ый бит выхода является функцией от *n*-ых битов трех входов. Функции следующие:

|  |  |
| --- | --- |
| Номер цикла | *ft (B, C, D)* |
| (0 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-jkE3xC.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-83Si4f.png19) | *(B https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-bAMH3O.pngC)https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-xf3RUm.png(¬ Bhttps://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-r_S0hJ.pngD)* |
| (20 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-TkuMyj.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-UixLDL.png39) | *B https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-HYzBh_.pngChttps://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-hrPrRQ.pngD* |
| (40 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-_vC1Pq.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-R_1dyC.png59) | *(B https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-bKVZUY.pngC)https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-caUN2A.png(Bhttps://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-HzChZ4.pngD)https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-6lKMuc.png(Chttps://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-YTWSCl.pngD)* |
| (60 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-FOLcBg.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-ibhNDt.png79) | *B https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-Tg4LMs.pngChttps://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-6nnVMv.pngD* |

На самом деле используются только три различные функции. Для 0 https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-AHmTYH.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-r4LEgE.png19 функция является условной: if *B* then *C* else *D*. Для 20https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-l0M57M.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-XkpPVe.png39 и 60https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-2dvxs3.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-i2N6Wx.png79 функция создает бит четности. Для 40https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-i1Q94g.png*t*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-_4wD0T.png59 функция является истинной, если два или три аргумента истинны.

32-битные слова *Wt* получаются из очередного 512-битного блока сообщения следующим образом.



Получение входных значений каждого цикла из очередного блока

Первые 16 значений Wt берутся непосредственно из 16 слов текущего блока. Оставшиеся значения определяются следующим образом:

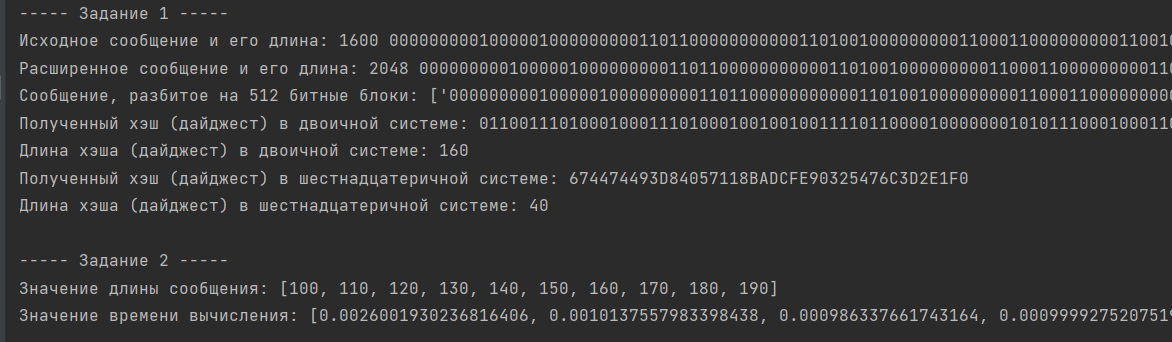
*Wt = Wt-*16*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-gsvbxp.pngWt-*14*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-nR0Xd0.pngWt-*8*https://studfile.net/html/2706/35/html_RyvasxLb5F.gZhq/img-jHJtEt.pngWt-*3

В первых 16 циклах вход состоит из 32-битного слова данного блока. Для оставшихся 64 циклов вход состоит из XOR нескольких слов из блока сообщения.

Алгоритм SHA-1 можно суммировать следующим образом:

*SHA0* = *IV*

*SHAq+*1 = Σ32 (*SHAq*, *ABCDEq* )

*SHA = SHAL-1,*

|  |
| --- |
| где *IV* - начальное значение буфера *ABCDE*. |
| *ABCDEq* - результат обработки q-того блока сообщения. |
| *L* - число блоков в сообщении, включая поля добавления и длины. |
| Σ32 - сумма по модулю 232, выполняемая отдельно для каждого слова буфера. |
| *SHA* - значение дайджеста сообщения. |

Рис. 1 – вывод

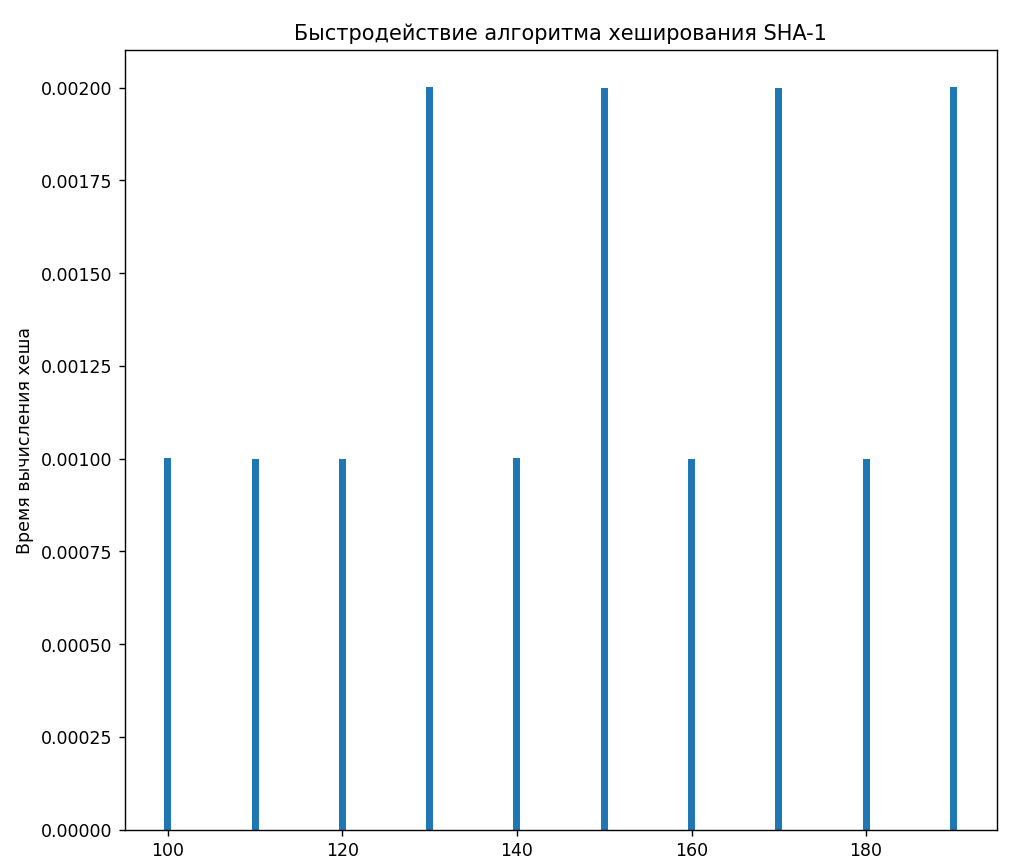
Хеш функция работает очень быстро и даже очень большие сообщения хеширует менее, чем за 0,001 секунду.

Рис. 2 – график зависимости время выполнения хеширования от длины сообщения

**Вывод**: в ходе лабораторной работы были изучены методы криптографических хеш-функций, так же было разработано приложение, использующий алгоритм из семейства SHA-1.