Лабораторная работа № 5. Хеш-функции

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

5.1 Общие сведения

Хеш-функции получили широкое распространение в разнообразных алгоритмах быстрого поиска информации.

Однако с появлением криптографии у них появилась вторая, ничуть не меньшая, область применения.

Хеш-функцией (англ, hash — мелко измельчать и перемешивать) называется необратимое преобразование данных, обладающее следующими свойствами:

на вход алгоритма преобразования может поступать двоичный блок данных произвольной длины;

на выходе алгоритма получается двоичный блок данных фиксированной длины;

значения на выходе алгоритма распределяются по равномерному закону по всему диапазону возможных результатов;

при изменении хотя бы одного бита на входе алгоритма его выход значительно меняется: в идеальном случае инвертируется произвольная половина бит.

Основное, но не единственное, предназначение хеш-функций в криптографии — вычисление "неподделываемых" контрольных сумм документов.

Действительно, если для алгоритма хеширования выполняются требования невозможности подобрать иной документ с той же хеш-суммой и невозможности подобрать два документа с произвольной одинаковой хеш-суммой, то хеш-сумма становится уникальной характеристикой документа:

5.2 Алгоритм вычисления хеш-функции ГОСТ 34.11

В алгоритме ГОСТ 34.11 используются следующие преобразования:

X-преобразование. На вход функции X подаются две последовательности длиной 512 бит каждая, выходом функции является XOR этих последовательностей.

S-преобразование. Функция S является обычной функцией подстановки. Каждый байт из 512-битной входной последовательности заменяется соответствующим байтом из таблицы подстановок π.

P-преобразование. Функция перестановки. Для каждой пары байт из входной последовательности происходит замена одного байта другим.

L-преобразование. Представляет собой умножение 64-битного входного вектора на бинарную матрицу A размерами 64x64.

Функция сжатия.

Основным элементом любой хеш-функции является функция сжатия.

Опишем используемую в новом стандарте функцию сжатия g n в виде алгоритма. Пусть h, N и m — 512-битные последовательности. Для вычисления функции g(N, m, h) необходимо проделать следующие шаги:

1) Вычислить значение K = h $\oplus$ N
2) Присвоить значение K = S(K)
3) Присвоить значение K = P(K)
4) Присвоить значение K = L(K)
5) Вычислить t = E(K, m)
6) Присвоить значение t = h $\oplus$ t
7) Вычислить значение G = t $\oplus$ m
8) Вернуть G в качестве результата вычисления функции g(N, m, h)

Функция E(K, m) выполняет нижеприведенные действия:

1) Вычислить значение state = K $\oplus$ m
2) Для i=0 по 11 выполнить:

– присвоить значение state = S(state);
– присвоить значение state = P(state);
– присвоить значение state = L(state);
– вычислить K=KeySchedule(K, i);
– присвоить значение state = state $\oplus$ K.

3) Вернуть state в качестве результата.

Функция KeySchedule(K, i) отвечает за формирование временного ключа K на каждом раунде функции E(K, m) и производит следующие вычисления:

1) Присвоить значение K = K $\oplus$ C[i].
2) Присвоить значение K = S(K).
3) Присвоить значение K = P(K).
4) Присвоить значение K = L(K).
5) Вернуть K в качестве результата функции.

Вычисление хеш-функции.

Для любого входного сообщения M:

1) Присвоить начальные значения внутренних переменных:

Для хеш-функции с длиной выхода 512 бит: h=iv=0x00^{64}

Для хеш-функции с длиной выхода 256 бит: h=iv=0x01^{64}

N = 0^{512}

$\Sigma = 0^{512}$

2) Проверить следующее условие:

длина сообщения M<512. Если условие выполняется перейти к пункту 3). В противном случае выполнить последовательность вычислений:

m — последние 512 бит сообщения M

h = g(N, m, h)

N = (N + 512) mod 2^{512}

$\Sigma = (\Sigma + m) \bmod 2^{512}$

Обрезать M, убрав последние 512 бит.

Перейти к шагу 2 .

3) Произвести дополнение сообщения M до длины в 512 бит по следующему правилу:

m = $0^{511-|M|}||1||$M, где |M| — длина сообщения M в битах.

Вычислить h = g(N, m, h).

Вычислить N = (N + |M|) mod 2^{512}

Вычислить $\Sigma = (\Sigma + m) \bmod 2^{512}$

Вычислить h = g(0, h, N)

Вычислить h = g(0, h, Σ)

Для хеш-функции с длиной выхода в 512 бит возвращаем h в качестве результата. Для функции с длиной выхода 256 бит возвращаем MSB 256 (h).

5.3 Алгоритм вычисления хеш-функции SHA-1

Алгоритм SHA-1 (Secure Hash Algorithm) предложен Институтом Стандартизации США NIST как стандарт хеширования в гражданской криптографии. Этот алгоритм был призван дать еще больший запас прочности к криптоатакам.

SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения.

Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш блока Mi равен hi = f (Mi , hi−1). Хеш-значением всего сообщения является выход последнего блока.

Алгоритм получает на входе сообщение максимальной длины 2^{64} бит и создает в качестве выхода *дайджест сообщения* длиной 160 бит.

Шаг 1: добавление недостающих битов

Сообщение добавляется таким образом, чтобы его длина была кратна 448 по модулю 512 (длина ≡ 448 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, . . . , Y_{L-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 *хеш-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 H_{SHA}.

Все 80 циклических обработок имеют одинаковую структуру.

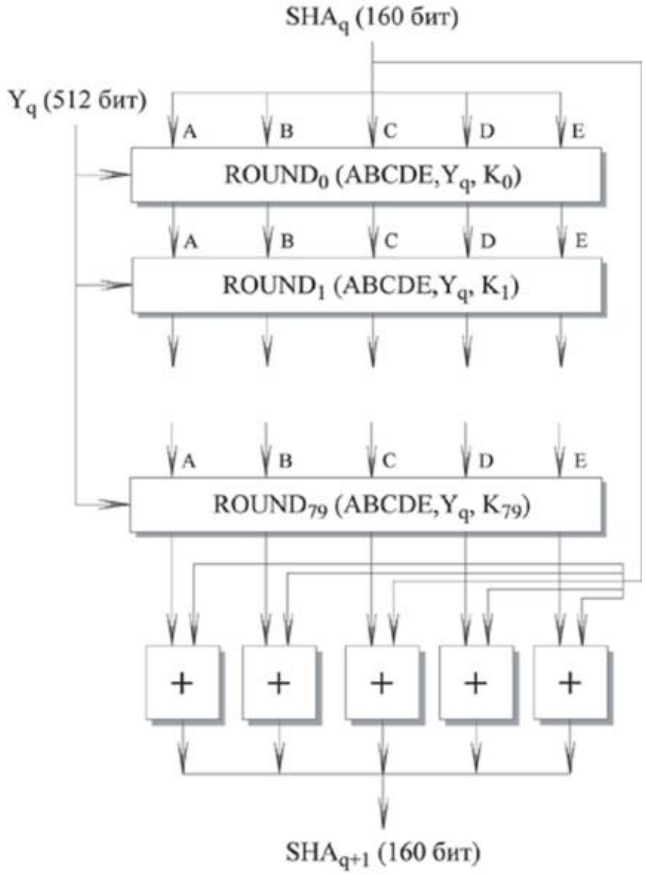

Рисунок 5,1 Обработка очередного 512-битного блока

Каждый цикл получает на входе текущий 512-битный обрабатываемый блок Y_q и 160-битное значение буфера *ABCDE*, и изменяет содержимое этого буфера.

В каждом цикле используется дополнительная константа K_t, которая принимает только четыре различных значения:

$0<=t<=19$ $\mathbf{K_t=5A827999}$ (целая часть числа $[2^{30} \times 2^{1/2}]$)

$20<=t<=39$ $\mathbf{K_t=6ED9EBA1}$ (целая часть числа $[2^{30} \times 3^{1/2}]$)

$40<=t<=59$ $\mathbf{K_t=8F1BBCDC}$ (целая часть числа $[2^{30} \times 5^{1/2}]$)

$60<=t<=79$ $\mathbf{K_t=CA62C1D6}$ (целая часть числа $[2^{30} \times 10^{1/2}]$)

Для получения SHA_{q+1} выход 80-го цикла складывается со значением SHA_q.

Сложение по модулю 2^{32} выполняется независимо для каждого из пяти слов в буфере с каждым из соответствующих слов в SHA_q.

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:

$$A,B,C,D,E(CLS_5(A)+f_t(B, C, D)+E+W_t+K_t),A,CLS_{30}(B),C,D$$

где

A, B, C, D, E - пять слов из буфера.

t - номер цикла, 0 <= t <= 79.

f_t - элементарная *логическая функция*.

CLS_s - циклический левый сдвиг 32-битного аргумента на s битов.

W_t - 32-битное слово, полученное из текущего входного 512-битного блока.

K_t - дополнительная константа.

\+ - сложение по модулю 2^{32}.

Логика выполнения отдельного цикла SHA-1

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 <= t <= 19)	$(B \wedge C) \vee (\neg B \wedge D)$
(20 <= t <= 39)	$B \oplus C \oplus D$
(40 <= t <= 59)	$(B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
(60 <= t <= 79)	$B \oplus C \oplus D$

На самом деле используются только три различные функции.

Для 0 <= t <= 19 функция является условной:

if B then C else D.

Для 20 <= t <= 39 и 60 <= t <= 79 функция создает бит *четности*.

Для 40 <= t <= 59 функция является истинной, если два или три аргумента истинны.

Получение входных значений каждого цикла из очередного блока/

Двухбитные слова W_t получаются из очередного 512-битного блока сообщения следующим образом.

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

$$W_t = W_{t-16} \oplus W_{t-14} \oplus W_{t-8} \oplus W_{t-3}$$

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:

$$\mathbf{SHA_0 = IV}$$

$$\mathbf{SHA_{q+1} = \Sigma 32(SHA_q, ABCDE_q),}$$

Где:

IV - начальное значение буфера *ABCDE*;

ABCDE $_{\mathbf{q}}$ - результат обработки q-того блока сообщения;

L - число блоков в сообщении, включая поля добавления и длины;

Σ32 - сумма по модулю 2^{32}, выполняемая отдельно для каждого слова буфера;

SHA - значение *дайджеста сообщения*.

ЗАДАНИЕ:

1. Изучить теоретические сведения
2. Реализовать самостоятельно (без использования готовых библиотек и функций) программное средство контроля целостности сообщений с помощью вычисления хеш-функции и алгоритма ГОСТ 34.11 и SHA 1.