

## Лабораторная работа № 9

### Исследование криптографических хеш-функций

**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.

### 9.1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

#### 9.1.1. Определение, свойства, описание и типы хеш-функций

Определение 1. *Хеш-функция* – математическая или иная функция,  $h = H(M)$ , которая принимает на входе строку символов  $M$ , 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).

Пример 1. Простейшим примером хеш-преобразования может быть вычисление на основе операции *суммирования по модулю 2*: символы (байты) входной строки  $M$  складываются побитно по модулю 2 и

байт-результат есть значение хеш-функции  $h$ . Длина  $l$  значения хеш-функции составит в этом случае 8 бит независимо от размера входного сообщения.

Если, в продолжение примера, представить входное сообщение в виде 16-ричных чисел:  $M = 1f\ 01\ 9a$  ( $n = 24$  бита) и далее выполнить операцию суммирования в соответствии вышеуказанным принципом:

$$\begin{array}{r} 00011111 \\ 00000001 \\ \underline{10011010} \\ 10000100. \end{array}$$

Таким образом, мы получили  $h = 10000100$  или  $h = 84$  (в 16-ричной системе счисления).

Все существующие функции хеширования можно разделить на два больших класса:

- бесключевые хеш-функции, зависящие только от сообщения,
- хеш-функции с секретным ключом, зависящие как от сообщения, так и от секретного ключа.

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.

На рис. 9.1 перечисленные некоторые известные хеш-функции, классифицированные в соответствии с используемым внутренним преобразованием.

К основным свойствам хеш-функций можно отнести следующие.

Свойство 1. *Детерминированность*: независимо от того, сколько раз вычисляется  $H(M)$ ,  $M - \text{const}$ , 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  $H(M)$  невозможно (практически) определить  $M$ . Это важнейшее свойство хеш-функции для криптографических применений – *одно-*

сторонности преобразования.

Это означает, что по хеш-коду должно быть практически невозможным восстановление входной строки  $M$ .

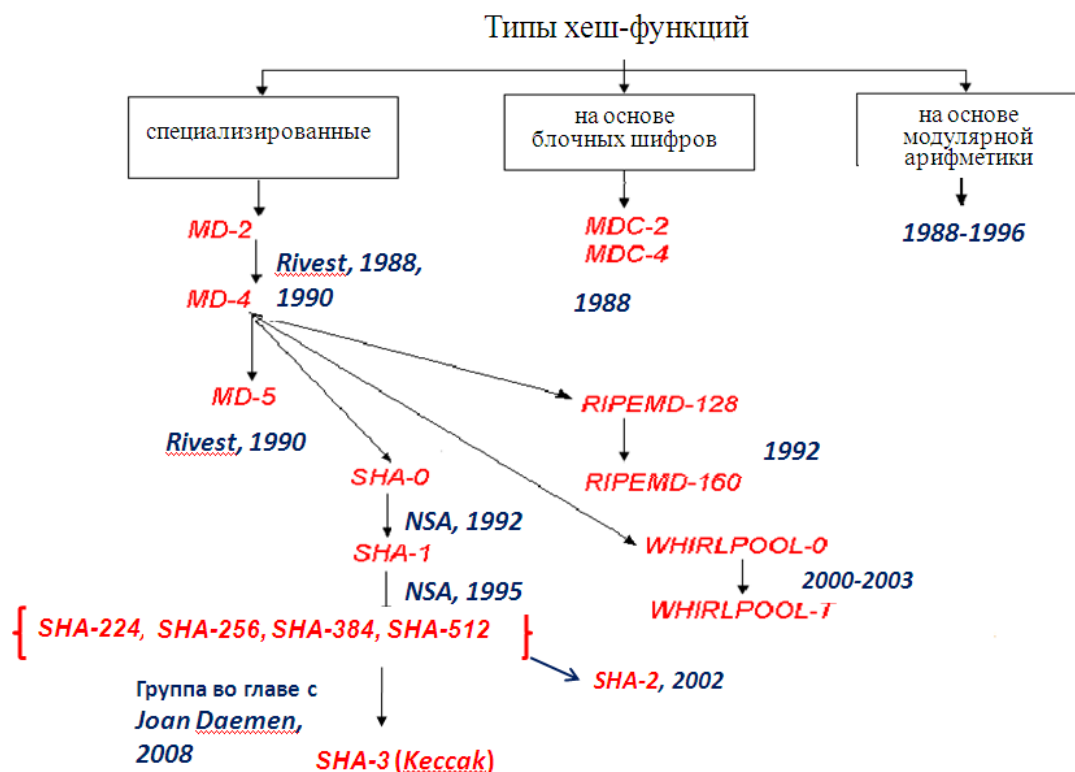


Рисунок 9.1 Основные классы хеш-функций в соответствии с используемым внутренним преобразованием

Большинству даже простых пользователей известно, что наиболее распространенная область применения хеширования – хранение паролей. К примеру, если пользователь забыл пароль и пытается воспользоваться доступными функциями-сервисами восстановления пароля, то в этом случае, такой сервис может выдать пароль, как правило, в двух случаях:

- информационная система, в которую входит пользователь с помощью пароля, на самом деле, не хеширует этот пароль,
- используемая система восстановления пароля использует некоторую базу данных, содержащую простые, наиболее часто используемые пароли (например, 123456 или qwerty); примером такой системы восстановления пароля является *Online Reverse Hash Lookup*.

Пример 2. При длине хеш-кода  $l = 128$  бит в среднем нужно проверить – по методу «грубой силы» –  $2^{128}/2 = 2^{127}$  вариантов входного сообщения для получения фиксированного  $H(M)$ .

В плане односторонности хешей на основе блочных шифров отметим одно обстоятельство. Блочный шифр необратим по ключу шифрования, и, если в качестве такого ключа на текущем шаге преобразования используется выход предыдущего шага, а в качестве шифруемого сообщения – очередной блок сообщения (или наоборот), то можно получить хеш-функцию с хорошими криптографическими характеристиками с точки зрения односторонности.

Такой подход использовался, например, в российском стандарте хеширования – ГОСТ Р 34.11-94.

Основным недостатком хеш-функций на основе блочных шифров является сравнительно невысокая производительность.

Свойство 4. Даже минимальные изменения в хешируемых данных ( $M \neq M'$ ) должны изменять хеш:  $H(M) \neq H(M')$ .

Пример 3. Используются алгоритмы хеширования MD5 и SHA1. Положим,  $M = \text{«Hash+login2020»}$  и  $M' = \text{«hash+login2020»}$ .

Для MD5 получились следующие хеши (в 16-ричном виде):

$H(M) = \text{c71b846d449901adb3b8308421ef203d}$ ,

$H(M') = \text{d228d48d152d929c8ff667dc1a2d663a}$ ,

соответственно – для SHA1:

$H(M) = \text{a39ec24cf6a4ab6d15f51c39791211af5743963f}$ ,

$H(M') = \text{a31ec6b7710c0e7d4e0c23b261eab9a0ac37177c}$ .

Определение 4. Коллизией хеш-функции  $H$  называют ситуацию, при которой различным входам (в общем случае –  $x$  и  $y$  или  $M \neq M'$ ) соответствует одинаковый хеш-код:  $H(x) = H(y)$  или  $H(M) = H(M')$ .

Свойство 5. Коллизионная устойчивость (стойкость).

Зная  $M$ , трудно найти такое  $M'$  ( $M \neq M'$ ), для которого  $H(M) = H(M')$ .

Если последнее равенство выполняется, то говорят о коллизии 1-го рода.

Если случайным образом выбраны два сообщения ( $M$  и  $M'$ ), для которых  $H(M) = H(M')$ , говорят о коллизии 2-го рода.

Мерой криптостойкости хеш-функции считается вычислительная сложность нахождения коллизии.

Для хеш-функций одним из основных средств поиска коллизий является метод, основанный на известной статистической задаче – «парадоксе дня рождения».

В более общем случае: для того, чтобы хеш-функция  $H(M)$  счита-

лась криптографически стойкой, она должна удовлетворять трем основным требованиям: необратимостью вычислений (свойство 3), устойчивостью к коллизиям первого рода и устойчивостью к коллизиям второго рода (свойство 5).

### ***9.1.2 Парадокс «дней рождений» и его использование в криптографических приложениях***

Основной постулат *парадокса «дней рождения»* гласит: в группе минимум из 23 человек с вероятностью более 0,5 день рождения у 2-х лиц одинаков. Парадоксом является высокая (как кажется на первый взгляд) вероятность наступления указанного события. При этом предполагается, что

- в этой группе нет близнецов;
- люди рождаются независимо друг от друга, т. е. дата (день) рождения любого человека не влияет на дату рождения другого;
- люди рождаются равномерно и случайно, т. е. люди с равной вероятностью могут родиться в любой день года; с формальной точки зрения это означает, что вероятность  $p_1$  рождения отдельно выбранного члена группы (как и любого человека) в любой выбранный день равна  $p_1 = 1/365$  (хотя известно, что в реальности рождение людей не совсем соответствует такому предположению).

Теперь посмотрим на ситуацию с иной позиции.

Пусть  $A_n$  будет соответствовать ситуации, при которой среди  $n$  членов группы все имеют различные дни рождения, т. е. дни рождения не совпадают. Вероятность такого события обозначим  $P(A_n)$ .

Если предположить, что группа состоит из 2-х человек ( $n = 2$ ), то вероятность несовпадения их дней рождения  $P(A_2) = 364/365 \approx 0,997$ . С вероятностью  $P(A_3) = (364/365) \cdot (363/365) \approx 0,991$  в группе из 3-х человек будут разные дни рождения.

Вероятность  $P(A_3)$  можно также представить следующим образом:

$$P(A_3) = P(A_2) \frac{(365-2)}{365}.$$

Следуя далее принятой логике рассуждений, можно также записать

$$P(A_4) = P(A_3) \frac{(365-3)}{365}$$

и, наконец,

$$P(A_{23}) = P(A_{22}) \frac{(365-22)}{365}. \quad (9.1)$$

Выполнив поступательно все вычисления, получим  $P(A_{23}) = 0,493$ . Тогда вероятность совпадения дня рождения для 2-х членов группы  $P_c(A_{23}) = 1 - P(A_{23}) = 1 - 0,493 = 0,507$ . Мы получили математическое подтверждение «парадоксу», сформулированному в начале подраздела.

Последний численный вывод мы получили с учетом того, что

$$P_c(A_n) = 1 - P(A_n). \quad (9.2)$$

Теперь несколько обобщим наши рассуждения, предположив, что существуют  $m$  (вместо 365) возможных дней рождения, а группа состоит из  $n$  человек. Тогда, например, (9.1) можно переписать в следующем виде:

$$P(A_n) = P(A_{n-1}) \frac{(m-(n-1))}{m}. \quad (9.3)$$

Подставляя в правую часть тождества (9.3) все необходимые сомножители, можно записать:

$$P(A_n) = \prod_{i=1}^{n-1} \left(1 - \frac{i}{m}\right). \quad (9.4)$$

Последнее соотношение можно аппроксимировать, положив  $(1 - x) \approx \exp(-x)$ :

$$\text{нужно } n^2 \quad P(A_n) = \prod_{i=1}^{n-1} \exp(-i/m) \approx \exp(-n^2/2m). \quad (9.5)$$

Теперь возвратимся к криптографической функции хеширования. Ее также можно определить следующим образом.

**Определение 5.** Хеш-функция – это функция, выполняющая отображение из множества  $M$  в число, находящееся в интервале  $[0, m-1]$ :  $h: M \rightarrow [0, m-1]$ .

Мы ранее отмечали, что стойкость хеш-преобразования к колли-

зии означает, что трудно найти такие  $M_i$  и  $M_j$  ( $M_i, M_j \in M$ ), при которых  $h(M_i) = h(M_j)$ ,  $i \neq j$ ,  $1 \leq i, j \leq n$ .

Для выполнения анализа атаки на основе парадокса «дней рождения» будем использовать те же принципы, которые мы применяли для вероятностной оценки дней рождения.

В атаке «дней рождения»  $m$  соответствует количеству календарных дней в году, а  $M$  – множеству людей, составляющих группу. Люди «хешируются» в их дни рождения, которые могут быть одним из значений  $m$ .

Допустим (переходя в информационную область), нам нужно найти коллизии с вероятностью 0,99 ( $P_c(A_n) = 0.99$ ). Мы хотим определить наименьшее  $n$ , при котором хеш двух значений из  $A_n$  будет «одним днем рождения», что в интересующей нас плоскости означает, что два входных набора данных ( $M_i, M_j \in M$ ) хешируются в одинаковое значение:  $h(M_i) = h(M_j)$ . Допустим далее, что все входные данные хешируются в  $m$  выходных хеш-кодов.

При атаке «дней рождения» злоумышленник будет случайным образом подбирать  $M_i$  и  $M_j$  и сохранять пары их хешей, пока не найдет двух значений, при которых  $h(M_i) = h(M_j)$ . Нам нужно определить, сколько раз атакующему нужно повторить эту операцию, пока не будет обнаружена коллизия.

Иначе говоря, стоит задача отыскания наименьшего  $n$ , при котором хеши двух значений  $m$  будут «одним днём рождения».

Итак, наш случай ограничивается частной задачей поиска условия, при котором выполняется тождество  $P_c(A_n) = 0.99$  или, в соответствии с (9.2) –  $P(A_n) = 0.01$ . Воспользуемся (9.5):

$$P(A_n) \approx \exp(-n^2/2m) = 0.01. \quad (9.6)$$

Последнее можно записать в виде равенства

$$(n^2/2m) = \ln 100,$$

откуда находим

$$n = (2m \cdot \ln 100)^{1/2}. \quad (9.7)$$

Последнее, кстати согласуется с нашим предыдущим анализом при  $m = 365$ , потому что  $(2 \cdot 365 \cdot \ln 2)^{1/2}$  примерно равно 23 (здесь берется

логарифм числа 2, поскольку правая часть (9.6) – это число 0.5).

Если хеш имеет длину  $l$  бит, то  $m = 2^l$ . И в соответствии с (9.7) для поиска коллизии с вероятностью 0.99 нужно выполнить  $2^{l/2}$  операций хеширования различных входных сообщений.

На рис. 9.2 приведены вероятностные оценки появления коллизии для хеш-функций различной длины (в приведенной таблице параметр  $N$  соответствует принятому нами обозначению  $l$ ).

N	Количество различных выходных цепочек ( $2^N$ )	Вероятность хотя бы одной коллизии (p)									
		$10^{-18}$	$10^{-15}$	$10^{-12}$	$10^{-9}$	$10^{-6}$	0,1 %	1 %	25 %	50 %	75 %
32	$4,3 \times 10^9$	2	2	2	2.9	93	$2.9 \times 10^8$	$9.3 \times 10^8$	$5.0 \times 10^4$	$7.7 \times 10^4$	$1.1 \times 10^5$
64	$1,8 \times 10^{19}$	6.1	$1.9 \times 10^2$	$6.1 \times 10^8$	$1.9 \times 10^5$	$6.1 \times 10^6$	$1.9 \times 10^8$	$6.1 \times 10^8$	$3.3 \times 10^9$	$5.1 \times 10^9$	$7.2 \times 10^9$
128	$3,4 \times 10^{38}$	$2.6 \times 10^{10}$	$8.2 \times 10^{11}$	$2.6 \times 10^{13}$	$8.2 \times 10^{14}$	$2.6 \times 10^{16}$	$8.3 \times 10^{17}$	$2.6 \times 10^{18}$	$1.4 \times 10^{19}$	$2.2 \times 10^{19}$	$3.1 \times 10^{19}$
256	$1,2 \times 10^{77}$	$4.8 \times 10^{29}$	$1.5 \times 10^{31}$	$4.8 \times 10^{32}$	$1.5 \times 10^{34}$	$4.8 \times 10^{35}$	$1.5 \times 10^{37}$	$4.8 \times 10^{37}$	$2.6 \times 10^{38}$	$4.0 \times 10^{38}$	$5.7 \times 10^{38}$
384	$3,9 \times 10^{115}$	$8.9 \times 10^{48}$	$2.8 \times 10^{50}$	$8.9 \times 10^{51}$	$2.8 \times 10^{53}$	$8.9 \times 10^{54}$	$2.8 \times 10^{56}$	$8.9 \times 10^{56}$	$4.8 \times 10^{57}$	$7.4 \times 10^{57}$	$1.0 \times 10^{58}$
512	$1,3 \times 10^{154}$	$1.6 \times 10^{68}$	$5.2 \times 10^{69}$	$1.6 \times 10^{71}$	$5.2 \times 10^{72}$	$1.6 \times 10^{74}$	$5.2 \times 10^{75}$	$1.6 \times 10^{76}$	$8.8 \times 10^{76}$	$1.4 \times 10^{77}$	$1.9 \times 10^{77}$

Источник: [http://cryptowiki.net/index.php?title=Атака\\_«дней\\_рождения»](http://cryptowiki.net/index.php?title=Атака_«дней_рождения»)

**Рисунок 9.2** Вероятностные оценки появления коллизии для хеш-кодов различной длины  $l$

Устойчивость хеш-функции к коллизиям имеет первостепенное значение в технологиях электронной цифровой подписи и криптовалютных технологиях. В настоящее время длина хеша  $l = 64$  не относится к числу криптостойких.

### **9.1.2 Структурные и функциональные особенности некоторых хеш-функций**

Рисунок 9.1 дает начальные представления об основных хеш-преобразованиях. В [40-47] можно ознакомиться с подробным описанием всех упоминаемых нами алгоритмов (от их создателей). Много полезной информации, а также программные коды некоторых алгоритмов хеширования на С можно найти в [4].

Алгоритмы семейства MD-х (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, SHA-256, SHA-384 и SHA-512. SHA-224 и SHA-384 являются, по сути, аналогами SHA-256 и SHA-512 соответственно.

Известен также алгоритм хеширования, долгое время использовавшийся в качестве национального стандарта (ГОСТ 34.11-94) России.

С 5 августа 2015 г. утвержден и опубликован в качестве действующего стандарта (FIPS 202) алгоритм SHA-3, одной из отличительных особенностей которого является использование конструкции «криптографической губки». В этой конструкции реализован итеративный подход для создания функции с произвольной длиной на входе и произвольной длиной на выходе на основе определенного преобразования [48]. На основе технологии «губки» построен ныне действующий в РФ стандарт хеширования [49].

Алгоритмы семейства MD входные сообщения максимальной длины  $2^{64-1}$  бит (в общем случае –  $L$  бит) преобразуют в хеш длиной  $l = 128$  бит. Исключением является последняя – 6 – из версий алгоритма, где длина результирующего хеша может изменяться от 1 до 512 бит.

Максимальный объем хешируемых сообщений для алгоритмов SHA-1, SHA-256, SHA-224 такой же, как и для алгоритмов MD. Однако длина хешей разная: в SHA-1 – 160 бит; во алгоритмах, относящихся к семейству SHA-2 – соответствует числу, дополняющему через дефис название алгоритма. Максимальная же длина входных сообщений в алгоритмах SHA-512, SHA-384, SHA-512/256, SHA-512/224 составляет  $2^{128-1}$  бит.

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.

Процедура расширения хешируемого сообщения достаточно по-

дробно описана в [1] на примере алгоритма MD-4 (см. п. 10.3.1).

Входное сообщение «дополняется» (расширяется) так, чтобы его длина (в битах) была конгруэнтной к 448 по модулю 512. Это значит, что сообщение начальной длиной  $L$  бит расширяется так, что остаются незаполненными всего лишь 64 бита, чтобы итоговая длина  $L'$  была кратной 512. В указанные 64 бита записывается двоичная длина  $L$ .

Расширение происходит всегда, даже если длина сообщения уже соответствует 448, по модулю 512. Эта операция выполняется следующим образом: один бит «1» добавляется к сообщению, а затем добавляются биты «0», так что длина в битах дополненного сообщения стала конгруэнтной 448 по модулю 512. Добавляется не менее одного бита, но не более 448 бит. Схематично состав и размеры дополненного входного сообщения показаны на рис. 9.3.

Входное сообщение длиной $L$ бит	Расширение: от 1 до 448 бит	64 бита
----------------------------------	-----------------------------	---------

$$L' (L' \bmod 512 \equiv 0)$$

**Рисунок 9.3 Состав и размеры дополненного хешируемого сообщения**

Основой рассматриваемых базовых алгоритмов является модуль, состоящий из циклических преобразований каждого 512-битного блока, который делится на подблоки длиной 32 либо 64 (в алгоритмах SHA-512, SHA-384, SHA-512/256, SHA-512/224) бита. При длине подблока в 16 бит каждый 512-битный блок должен состоять из 32 подблоков.

Рассмотрим пример.

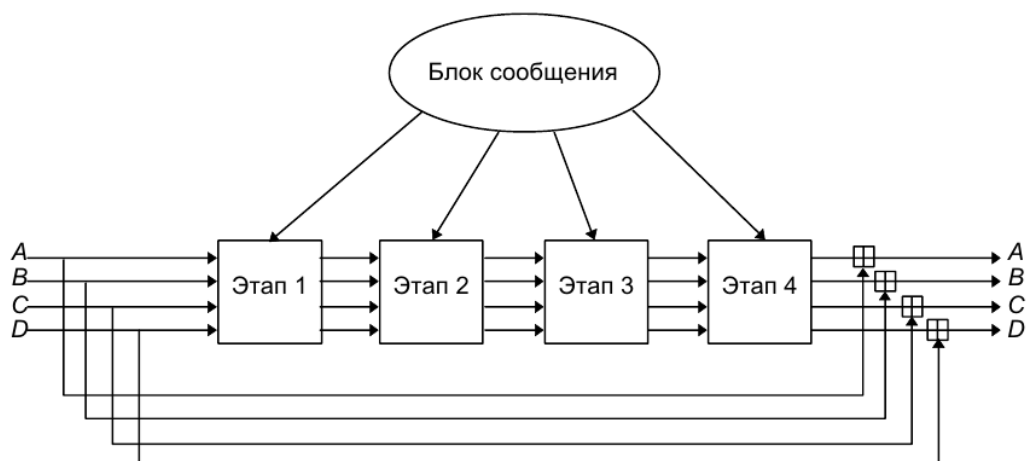
Пример 3. Пусть исходное сообщение или первообразная будет словом «CRYPTO» ( $M = \text{«CRYPTO»}$ ) или в кодах ASCII – это будут десятичные (67-82-89-80-84-79) и соответствующие двоичные (01000011-01010010-01011001-01010000-01010100-01001111; здесь для отделения чисел используются дефисы) числа. Как видим, длина  $L$  хешируемого сообщения равна 48 битам. Эту длину мы должны расширить до 448 бит, добавив одну «1» и 399 «0». В последнюю часть из 64 бит полученного 512-битного модуля ( $L' = 512$ ) мы запишем справа двоичное представление числа  $L = 48$ : 110000. В остальные 58 разрядов (из 64) мы впишем «0». После этого полученный

расширенный блок делим на 16-ти 32-разрядных подблоков:

[illegible]

Как было отмечено выше, основная операция заключается в циклической (пораундовой или поэтапной) обработке 512-битных блоков. Таких циклов может быть 3 (как в MD-4), или 4 (как в MD-5), или более. В каждом цикле используется своя нелинейная функция (обычно обозначаемая по порядку  $F, G, H, \dots$ ), зависящая от текущего состояния 4 (в MD), 5 (в SHA-1), 8 (SHA-256) и т. д. переменных, начальные состояния которых известны, а текущие — зависят от выполненных операций над хешируемым сообщением [1, 4].

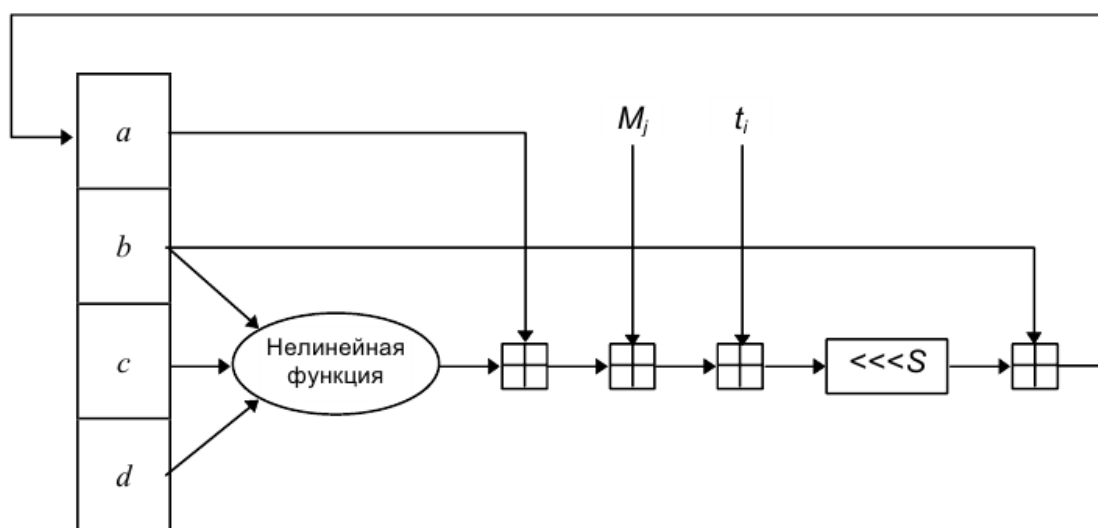
На рис. 9.4 приведена укрупненная структурная схема алгоритма MD-5, на рис. 9.5 – структурная схема одной операции (над одним 32-разрядным подблоком).



**Рисунок 9.4 Укрупненная структурная схема алгоритма MD-5**

Здесь знак «+» в квадратной фигуре соответствует операции сложения по модулю  $2^{32}$ . Вспомним (см. ЛР №5):

$$A + B \pmod{2^{32}} = \left\{ \begin{array}{l} A + B, \text{ если } A + B < 2^{32} \\ A + B - 2^{32}, \text{ если } A + B \geq 2^{32} \end{array} \right\}.$$



**Рисунок 9.5 Структурная схема одной операции алгоритма MD-5**

Главный модуль (рис. 9.4) состоит из четырех похожих этапов (у MD-4 было только три этапа). На каждом этапе 16 раз используются различные операции. Каждая операция представляет собой нелинейную функцию над тремя из  $a$ ,  $b$ ,  $c$  и  $d$ . Затем она добавляет этот результат к четвертой переменной, подблоку текста  $M_j$  и константе  $t_i$ . Далее результат циклически сдвигается вправо на переменное число  $s$  бит и добавляет результат к одной из переменных  $a$ ,  $b$ ,  $c$  и  $d$ . Наконец результат заменяет одну из этих переменных (рис. 9.5).

Результатом хеширования,  $h$ , является конкатенация последних значений указанных переменных, т. е.  $32 \cdot 4 = 128$  бит.

На рис. 9.6 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 5:  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ . Сдвиг и сложение – аналогично MD5.

В алгоритме используются следующие четыре константы:

$K_t = 0x5a827999$ , при  $t = 0, \dots, 19$ ,

$K_t = 0x6ed9eba1$ , при  $t = 20, \dots, 39$ ,

$K_t = 0x8f1bbcd6$ , при  $t = 40, \dots, 59$ ,

$K_t = 0xca62c1d6$ , при  $t = 60, \dots, 79$

Блок сообщения трансформируется из 16-ти 32-битных слов (от  $M_0$  по  $M_{15}$ ) в 80-т 32-битных слов ( $W_0, \dots, W_{79}$ ) с помощью следующего алгоритма:

$W_t = M_t$ , при  $t = 0, \dots, 15$ ,

$W_t = (W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) \lll 1$ , при  $t = 16, \dots, 79$ .

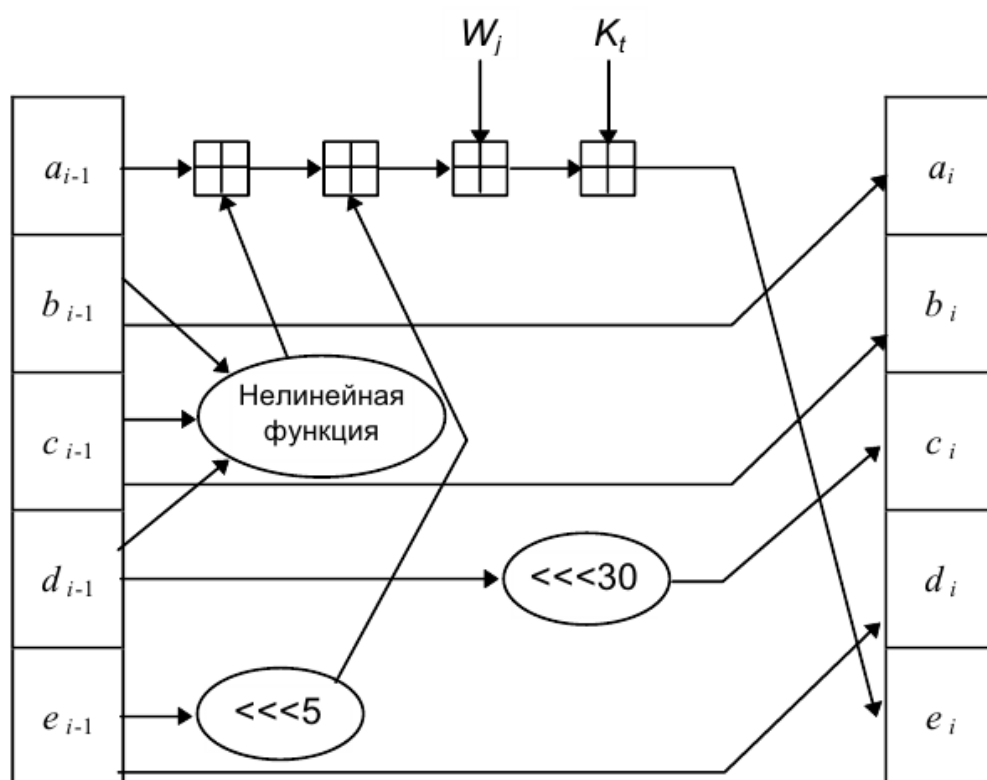


Рисунок 9.6 Схема выполнения одной операции в алгоритме SHA-1 [4]

После обработки всех 512-битных блоков выходом является 160-битный дайджест сообщения в виде конкатенации последних значений переменных  $a$ ,  $b$ ,  $c$ ,  $d$ ,  $e$ .

Как мы заметили, в алгоритмах MD-5 SHA-1 результат текущего действия прибавляется к результату предыдущего. Это направлено на усиление лавинного эффекта. Этой же цели служит то обстоятельство, что значения циклического сдвига влево на каждом этапе были приближенно оптимизированы: четыре сдвига, используемые на каждом

этапе, отличаются от значений, используемых на других этапах.

Как отмечает Б. Шнайер [4], SHA – это MD-4 с добавлением расширяющего преобразования, дополнительного этапа и с улучшенным лавинным эффектом. MD-5 – это MD-4 с улучшенным битовым хешированием, дополнительным этапом и улучшенным лавинным эффектом.

## **9.2 Практическое задание**

1. Разработать оконное приложение, реализующее один из алгоритмов хеширования из указанного преподавателем семейства (MD или SHA; или иного). При этом можно воспользоваться доступными готовыми библиотеками. Язык программирования – на свой выбор.

Приложение должно обрабатывать входные сообщения, длина которых определяется спецификацией на реализуемый алгоритм.

2. Оценить быстроедействие выбранного алгоритма хеширования.

3. Результаты оформить в виде отчета по установленным правилам.

## **ВОПРОСЫ ДЛЯ КОНТРОЛЯ И САМОКОНТРОЛЯ**

1. Дать определение хеш-функции.

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

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

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

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

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) бит?

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

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; 16000 бит. Какова в каждом случае будет длина хеша?

9. Входное сообщение (прообраз) состоит из

а) вашего имени,

б) ваших фамилии\_имени\_отчества (алфавит – на свой выбор).

Используя представление сообщения в кодах ASCII, представить в табличной форме (как выше в примере 5) содержание каждого 32-битного подблока расширенного входного сообщения.

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

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

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

13. Охарактеризовать структурные, функциональные особенности и криптостойкость белорусского государственного стандарта хеширования (СТБ 34.101.77-2016).

## К СПИСКУ ЛИТЕРАТУРЫ

39. Коркинъ, А. Н. Таблица первообразныхъ корней и характеровъ, къ нимъ относящихся, для простыхъ чиселъ, меньшихъ 4000// Матем. сб., 1909, т. 27, № 1. – С. 121–137 (Ресурс удаленного доступа: <http://www.mathnet.ru/links/10b5c3ec64e112fd63987586e9f47e93/sm6551.pdf>)
40. Rivest, R. RFC 1320: The MD4 Message Digest Algorithm. – MIT Laboratory for Computer Science & RSA Data Security, October 1990. – 20 p. (<https://tools.ietf.org/html/rfc1320>)
41. Rivest, R. RFC 1321: The MD5 Message-Digest Algorithm The MD5 Message-Digest Algorithm. – Internet Engineering Task Force, 1992. – 21p. (<https://tools.ietf.org/html/rfc1321>)
42. RFC 3174: US Secure Hash Algorithm 1 (SHA1). – Network Working Group, 2001. – 22 p. <https://tools.ietf.org/html/rfc3174>
43. Housley, R. RFC 3174: A 224-bit One-way Hash Function: SHA-224. – Network Working Group, 2004. – 6 p. <https://tools.ietf.org/html/rfc3874>
44. RFC 4634: US Secure Hash Algorithms (SHA and HMAC-SHA) ). – Network Working Group, 2006. – 108p. <https://tools.ietf.org/html/rfc4634>
45. FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. – National Institute of Standards and Technology , 2015. – 29 p.  
<https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf>
46. MD6 website. <http://groups.csail.mit.edu/cis/md6/>
47. ГОСТ Р 34.11-94 Информационная технология (ИТ). Криптографическая защита информации. Функция хэширования. – М.: Издательство стандартов, 1994.  
<http://docs.cntd.ru/document/1200004857>
48. Bertoni G., Daemen J., Peeters M., Van Assche G. Cryptographic sponge functions, 2011. <http://sponge.noekeon.org/CSF-0.1.pdf>
49. СТБ 34.101.77-2016. Информационные технологии и безопасность. Алгоритмы хэширования. – Минск: Госстандарт, 2016. – 13 с.  
<http://www.apmi.bsu.by/assets/files/std/bash-spec10.pdf>