Перевод FIPS 180-4 (Secure hash standard)

Версия 1

Контроль версий

Версия	Дата	Изменения
1	сентябрь 2022	Первая версия.

Будьте осторожны, это любительский перевод ! Связь по почте tiuta сбк mail тчк ru

Стандарт криптографически стойкого хэша

Категория: компьютерная безопасность
Подкатегория: криптография

Лаборатория информационных технологий национального института стандартов и технологий.
Адрес: штат Мэриленд, г. Гейтерсберг, 20899-8900.

Данный документ доступен бесплатно по ссылке: http://dx.doi.org/10.6028/NIST.FIPS.180-4

Август 2015

Министерство торговли США
Penny Pritzker, министр

Национальный институт стандартов и технологий
Willie E. May, заместитель министра торговли по стандартам и технологиям, а также директор национального института стандартов и технологий

Вводная часть

Федеральные стандарты обработки информации, выпускаемые национальным институтом стандартов и технологий (NIST), являются официальными документами, принимаемыми и распространяемыми на основании федерального закона об информационной безопасности (FISMA) 2002 года. Критические замечания, относящиеся к федеральным стандартам обработки информации, приветствуются и принимаются директором лаборатории информационных технологий национального института стандартов и технологий по адресу 100 Bureau Drive, Stop 8900, Gaithersburg, MD 20899-8900.

Charles H. Romine, директор лаборатории
информационных технологий

Аннотация

Данный стандарт описывает алгоритмы, используемые для вычисления отпечатков сообщений. Отпечатки позволяют определить, было ли сообщение изменено с момента вычисления отпечатка.
Ключевые слова: компьютерная безопасность, криптография, отпечаток сообщения, хэш-функция, хэш-алгоритм, федеральные стандарты обработки информации, стандарт криптографически стойкого хэша.

Федеральные стандарты обработки информации
Публикация 180-4
Август 2015
Характеристика стандарта

Федеральные стандарты обработки информации публикуются национальным институтом стандартов и технологий после утверждения министром торговли. Выпуск стандартов осуществляется в соответствии с частью 5131 закона о реформе в области использования информационных технологий 1996 года (общественное право 104-106) и законом о компьютерной безопасности 1987 года (общественное право 100-235).

1. Название стандарта. Стандарт криптографически стойкого хэша (FIPS 180-4).

2. Категория стандарта. Стандарт компьютерной безопасности, криптография.

3. Описание. В данном стандарте описаны алгоритмы SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256, предназначенные для преобразования сообщения (т. е. некоторых данных) в сжатую форму, называемую отпечатком сообщения. В зависимости от алгоритма длина отпечатка составляет от 160 до 512 бит.

Каждый алгоритм имеет ограничение на максимальную длину сообщения. Алгоритмы SHA-1, SHA-224 и SHA-256 могут быть использованы только при условии, что длина сообщения меньше 2^{64} бит. Алгоритмы SHA-384, SHA-512, SHA-512/224 и SHA-512/256 могут быть использованы только при условии, что длина сообщения меньше 2^{128} бит.

Описанные в данном стандарте алгоритмы называют криптографически стойкими, потому что вычислительно невозможно:

1) найти сообщение, соответствующее заранее заданному отпечатку,

2) найти два разных сообщения, отпечатки которых совпадают.

Отпечатки сообщений обычно используются совместно с другими криптографическими алгоритмами, например, алгоритмами цифровой подписи, алгоритмами проверки подлинности сообщений с помощью хэш-функции и ключа, а также в генераторах случайных чисел (бит). Любое изменение сообщения с очень большой вероятностью повлечёт за собой изменение его отпечатка. Это позволяет алгоритмам проверки подлинности сообщений и алгоритмам проверки цифровой подписи выявить изменение сообщения.

Данный стандарт принят взамен FIPS 180-3.

4. Утверждающий орган. Министр торговли.

5. Поддерживающая организация. Министерство торговли США, национальный институт стандартов и технологий, лаборатория информационных технологий.

6. Применение. Данный стандарт разрешается применять во всех федеральных департаментах и агентствах для защиты важных данных, не относящихся к государственной тайне, исключая данные, на которые распространяется действие статьи 2315 раздела 10 Кодекса США, а также данные, относящиеся к национальной системе безопасности на основании статьи 11103(a)(1) раздела 40 Кодекса США. При необходимости использования криптографически стойкого хэша в задачах федерального уровня, в том числе совместно с другими криптографическими алгоритмами и протоколами, должен быть использован либо данный стандарт, либо FIPS 202. Также разрешается внедрение и использование данного стандарта в государственных органах нефедерального уровня.

7. Обозначения. Федеральный стандарт обработки информации 180-4 (FIPS 180-4), стандарт криптографически стойкого хэша.

8. Реализации. Алгоритмы, описанные в данном стандарте, могут быть реализованы в виде программ, прошивок, аппаратных средств или любых их комбинаций. Реализации алгоритмов будут считаться соответствующими данному стандарту, только если это подтверждено NIST. Информация о процедуре подтверждения доступна по ссылке http://csrc.nist.gov/groups/STM/index.html.

9. Внедрение. Процедура подтверждения соответствия стандарту FIPS 180-4, а также взаимосвязь стандартов FIPS 180-4 и FIPS 140-2, описаны в подразделе 1.10 руководства по внедрению стандарта FIPS 140-2, а также в руководстве по проверке криптографических модулей, доступному по ссылке http://csrc.nist.gov/groups/STM/cmvp/index.html.

10. Патенты. Реализации алгоритмов, описанных в данном стандарте, могут быть защищены патентами США или иностранными патентами.

11. Экспортный контроль. Некоторые криптографические устройства и относящиеся к ним технические данные являются объектом федерального экспортного контроля. Экспорт криптографических модулей, реализующих данный стандарт, и относящихся к ним технических данных должен производиться согласно действующему федеральному законодательству при наличии лицензии от бюро экспортного контроля министерства торговли США. Информация о правилах экспорта доступна по ссылке http://www.bis.doc.gov/index.htm.
12. Пересмотр. Целью данного стандарта является установление основных требований при формировании отпечатков сообщений. Однако соответствие данному стандарту не означает, что конкретная реализация является безопасной. В каждом департаменте или агентстве ответственный орган должен гарантировать, что принятые меры обеспечивают приемлемый уровень безопасности. Пересмотр данного стандарта для оценки его соответствия требованиям безопасности будет производиться каждые пять лет.
13. Процедура отказа. В соответствии с распоряжением министра торговли выполнение федеральных стандартов обработки информации является обязательным. Федеральный закон об информационной безопасности не предусматривает возможности отказа от выполнения федеральных стандартов обработки информации.
14. Получение копии данного стандарта. Данная публикация доступна в электронном виде по ссылке http://csrc.nist.gov/publications/. На этом же веб-сайте доступны другие публикации, посвящённые компьютерной безопасности.

Федеральные стандарты обработки информации
Публикация 180-4
Стандарт криптографически стойкого хэша
Оглавление

1. Введение

В данном стандарте описаны алгоритмы SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256. Каждый алгоритм реализует итеративную[1] однонаправленную хэш-функцию. Данные алгоритмы используются для преобразования сообщения в сжатую форму, называемую отпечатком сообщения или хэшем.

Любое изменение сообщения с большой вероятностью приведёт к изменению его отпечатка. Это свойство позволяет выявить факт изменения сообщения. Оно широко используется при создании и проверке цифровых подписей, формировании кодов проверки подлинности сообщений, а также при вычислении случайных чисел или бит.

Каждый алгоритм может быть разделён на два этапа: подготовку исходных данных и вычисление хэша. Подготовка заключается в дополнении сообщения, разделении дополненного сообщения на блоки длиной *m* бит и присваивании начальных значений переменным, используемым при вычислении хэша. На этапе вычисления производится последовательная обработка блоков. Каждый блок с помощью функций, констант и операций над словами преобразуется в число, называемое промежуточным значением хэша. Результат обработки последнего блока является отпечатком сообщения.

Алгоритмы значительно различаются по уровню криптографической стойкости. Стойкость алгоритмов хэширования, а также общая стойкость криптосистем, в которых алгоритмы хэширования используются совместно с алгоритмами цифровой подписи и алгоритмами проверки подлинности сообщений, обсуждаются в SP 800-57 и SP 800-107.

Также алгоритмы отличаются друг от друга длиной используемых блоков данных и разрядностью слов. Основные параметры алгоритмов приведены в таблице 1.

Таблица 1 – параметры алгоритмов

алгоритм	длина сообщения, бит	длина блока, бит	разрядность слова, бит	длина отпечатка, бит
SHA-1	менее 2^{64}	512	32	160
SHA-224	менее 2^{64}	512	32	224
SHA-256	менее 2^{64}	512	32	256
SHA-384	менее 2^{128}	1024	64	384
SHA-512	менее 2^{128}	1024	64	512
SHA-512/224	менее 2^{128}	1024	64	224
SHA-512/256	менее 2^{128}	1024	64	256

2. Определения

2.1 Термины и сокращения

бит	Двоичная цифра, имеющая значение 0 или 1.
байт	Совокупность 8 бит.
FIPS	Федеральный стандарт обработки информации.
NIST	Национальный институт стандартов и технологий.
SHA	Алгоритм вычисления криптографически стойкого хэша.
SP	Специальная публикация.
слово	В зависимости от алгоритма это либо совокупность 32 бит (4 байт), либо 64 бит (8 байт).

[1] Т. е. основанную на многократном повторении определённой последовательности действий – прим. пер.

2.2 Обозначения и операции

2.2.1 Обозначения

При описании алгоритмов используются приведённые ниже обозначения.

a, b, c, …, h	Вспомогательные переменные, использующиеся при вычислении значений хэша H^i. Являются *w*-битными словами.
H^i	*i*-ое значение хэша. H^0 – начальное значение хэша. H^N – конечное значение хэша.
H_j^i	*j*-ое слово *i*-го значения хэша. H_0^i – крайнее слева слово *i*-го значения хэша.
K_t	Константа, используемая на *t*-ой итерации вычисления хэша.
k	Количество нулей, добавляемых к сообщению при его дополнении.
L	Длина сообщения *M* в битах.
m	Количество бит в блоке M^i.
M	Исходное сообщение.
M^i	*i*-й блок длиной *m* бит.
M_j^i	*j*-ое слово *i*-го блока. M_0^i – крайнее слева слово *i*-ого блока.
n	Количество бит, на которое происходит смещение или циклический сдвиг при выполнении операции над словом.
N	Количество блоков в дополненном сообщении.
T	Вспомогательное *w*-битное слово, используемое при вычислении хэша.
w	Количество бит в слове.
W_t	*t*-ое *w*-битное слово вспомогательной последовательности слов.

2.2.2 Операции

При описании алгоритмов используются приведённые ниже операции над *w*-битными словами.

∧	Побитовое И.
∨	Побитовое ИЛИ (включающее ИЛИ).
⊕	Побитовое XOR (исключающее ИЛИ).
¬	Побитовое отрицание.
+	Сложение по модулю 2^w.
$\ll$	Сдвиг влево. Для вычисления выражения $x \ll n$ необходимо от слова *x* отбросить *n* бит слева, а затем добавить *n* нулевых бит справа.
$\gg$	Сдвиг вправо. Для вычисления выражения $x \gg n$ необходимо от слова *x* отбросить *n* бит справа, а затем добавить *n* нулевых бит слева.
$ROTL^n(x)$	Циклический сдвиг влево. Вычисляется согласно выражению $ROTL^n(x) = (x \ll n) \vee (x \gg w - n)$, где *x* – это *w*-битное слово, а *n* – целое, удовлетворяющее неравенству $0 \le n < w$.
$ROTR^n(x)$	Циклический сдвиг вправо. Вычисляется согласно выражению $ROTR^n(x) = (x \gg n) \vee (x \ll w - n)$, где *x* – это *w*-битное слово, а *n* – целое, удовлетворяющее неравенству $0 \le n < w$.
$SHR^n(x)$	Сдвиг вправо. Вычисляется согласно выражению $SHR^n(x) = x \gg n$, где *x* – это *w*-битное слово, а *n* – целое, удовлетворяющее неравенству $0 \le n < w$.

3. Способы записи и соглашения

3.1 Последовательности бит и целые числа

В качестве 16-ричных цифр используются символы из набора {0, 1, …, 9, A, …, F}. Каждая 16-ричная цифра соответствует определённой последовательности из 4 бит. Например, 16-ричная цифра «7» соответствует последовательности «0111», а 16-ричная цифра «A» соответствует последовательности «1010».

Слово записывается либо как последовательность w бит, либо как последовательность 16-ричных цифр. Для преобразования двоичной записи слова в 16-ричную каждый набор из 4-х бит заменяется 16-ричной цифрой. Например, 32-битная последовательность

1010 0001 0000 0011 1111 1110 0010 0011

может быть записана как «A103FE23». А 64-битная последовательность

1010 0001 0000 0011 1111 1110 0010 0011
0011 0010 1110 1111 0011 0000 0001 1010

может быть записана как «A103FE2332EF301A». Здесь и далее по тексту при записи 32-битных и 64-битных слов используется соглашение о том, что самым старшим битом слова является крайний слева бит.

Целое число от 0 до $2^{32} - 1$ включительно может быть представлено 32-битным словом. В 16-ричной записи слова крайняя справа цифра соответствует 4-м младшим битам. Например, целое $291 = 2^8 + 2^5 + 2^1 + 2^0 = 256 + 32 + 2 + 1$ записывается в 16-ричном виде как «00000123».

Целые числа от 0 до $2^{64} - 1$ включительно могут быть представлены 64-битным словом и записываются аналогично.

Целое число может быть записано в виде пары слов. На этапе дополнения сообщения такая запись используется для длины сообщения (см. подраздел 5.1).

Если Z – целое, удовлетворяющее неравенству $0 \le Z < 2^{64}$, то оно может быть представлено в виде $Z = 2^{32}X + Y$, где $0 \le X < 2^{32}$ и $0 \le Y < 2^{32}$. Числа X и Y могут быть записаны в виде 32-битных слов x и y соответственно, поэтому целое Z может быть записано в виде составного слова xy. Это свойство используется в алгоритмах SHA-1, SHA-224 и SHA-256.

Если Z – целое, удовлетворяющее неравенству $0 \le Z < 2^{128}$, то оно может быть представлено в виде $Z = 2^{64}X + Y$, где $0 \le X < 2^{64}$ и $0 \le Y < 2^{64}$. Числа X и Y могут быть записаны в виде 64-битных слов x и y соответственно, поэтому целое Z может быть записано в виде составного слова xy. Это свойство используется в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256.

В алгоритмах SHA-1, SHA-224 и SHA-256 используются блоки длиной 512 бит, при этом каждый блок рассматривается как последовательность шестнадцати 32-битных слов. В алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256 используются блоки длиной 1024 бит, при этом каждый блок рассматривается как последовательность шестнадцати 64-битных слов.

3.2 Операции

В описываемых алгоритмах используются следующие операции над w-битными словами:

1. побитовые логические операции $\wedge$, $\vee$, $\oplus$, $\neg$ (см. пункт 2.2.2),
2. сложение по модулю 2^w,
3. сдвиг вправо $SHR^n(x)$,
4. циклический сдвиг вправо $ROTR^n(x)$,
5. циклический сдвиг влево $ROTL^n(x)$.

В алгоритмах SHA-1, SHA-224 и SHA-256 используются 32-битные слова ($w = 32$). В алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256 используются 64-битные слова ($w = 64$).

Операция сложения по модулю 2^w определяется с помощью вспомогательной операции mod. Для положительных целых U и V результатом операции U mod V является остаток от деления U на V. Сложение слов x и y по модулю 2^w производится следующим образом:

- складываемые слова x и y преобразуются в целые числа X и Y, где $0 \le X < 2^w$ и $0 \le Y < 2^w$,
- вычисляется число $Z = (X + Y) \bmod 2^w$, где $0 \le Z < 2^w$,
- число Z преобразуется в слово z, которое является результатом сложения, т. е. $x + y = z$.

Операция сдвига вправо $SHR^n(x)$, где x – это w-битное слово, а n – целое, удовлетворяющее неравенству $0 \le n < w$, определяется выражением $SHR^n(x) = x \gg n$. Данная операция используется в алгоритмах SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256.

Операция циклического сдвига вправо $ROTR^n(x)$, где x – это w-битное слово, а n – целое, удовлетворяющее неравенству $0 \le n < w$, определяется выражением $ROTR^n(x) = (x \gg n) \vee (x \ll w - n)$. Данная операция осуществляет циклический сдвиг бит числа x на n позиций вправо. Она используется в алгоритмах SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256.

Операция циклического сдвига влево $ROTL^n(x)$, где x – это w-битное слово, а n – целое, удовлетворяющее неравенству $0 \le n < w$, определяется выражением $ROTL^n(x) = (x \ll n) \vee (x \gg w - n)$. Данная операция осуществляет циклический сдвиг бит числа x на n позиций влево. Она используется только в алгоритме SHA-1.

Для любого фиксированного w выполняются следующие соотношения:

$$ROTL^n(x) = ROTR^{w-n}(x)$$
$$ROTR^n(x) = ROTL^{w-n}(x)$$

4. Функции и константы

4.1 Функции

В данном подразделе описаны используемые в алгоритмах функции. В алгоритмах SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256 используются схожие функции. Но из-за разной разрядности входных (и выходных) слов их описания разделены на два пункта. Функции, используемые в алгоритмах SHA-224 и SHA-256, описаны в пункте 4.1.2. Функции, используемые в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256, описаны в пункте 4.1.3.

В каждом алгоритме используются функции $Ch(x, y, z)$ и $Maj(x, y, z)$. При замене в этих функциях операции исключающего ИЛИ ($\oplus$) на операцию побитового ИЛИ ($\vee$) будет получен идентичный результат.

4.1.1 Функции в алгоритме SHA-1

В алгоритме SHA-1 используются 80 функций f_0, f_1, …, f_{79}. Каждая функция преобразует три 32-битных слова x, y и z в новое 32-битное слово. Функции определяются выражением

$$f_t(x,y,z) = \begin{cases} Ch(x,y,z) = (x \wedge y) \oplus (\neg x \wedge z) & 0 \le t \le 19 \\ Parity(x,y,z) = x \oplus y \oplus z & 20 \le t \le 39 \\ Maj(x,y,z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) & 40 \le t \le 59 \\ Parity(x,y,z) = x \oplus y \oplus z & 60 \le t \le 79 \end{cases} \quad (4.1)$$

4.1.2 Функции в алгоритмах SHA-224 и SHA-256

В алгоритмах SHA-224 и SHA-256 используются шесть функций. Аргументами функций являются 32-битные слова x, y и z. Результатом вычисления функции является новое 32-битное слово.

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \quad (4.2)$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \quad (4.3)$$

$$\sum_0^{256}(x) = ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) \quad (4.4)$$

$$\sum_1^{256}(x) = ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) \quad (4.5)$$

$$\sigma_0^{256}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) \quad (4.6)$$

$$\sigma_1^{256}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) \quad (4.7)$$

4.1.3 Функции в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256

В алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256 используются шесть функций. Аргументами функций являются 64-битные слова *x*, *y* и *z*. Результатом вычисления функции является новое 64-битное слово.

$$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \tag{4.8}$$

$$Maj(x, y, z) = (x \wedge y) \oplus (x \wedge z) \oplus (y \wedge z) \tag{4.9}$$

$$\sum_0^{512}(x) = ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) \tag{4.10}$$

$$\sum_1^{512}(x) = ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) \tag{4.11}$$

$$\sigma_0^{512}(x) = ROTR^{1}(x) \oplus ROTR^{8}(x) \oplus SHR^{7}(x) \tag{4.12}$$

$$\sigma_1^{512}(x) = ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^{6}(x) \tag{4.13}$$

4.2 Константы

4.2.1 Константы в алгоритме SHA-1

В алгоритме SHA-1 используются 80 констант K_0, K_1, …, K_{79}. Константы[2] являются 32-битными словами и приведены ниже.

$$K_t = \begin{cases} 5A827999 & 0 \le t \le 19 \\ 6ED9EBA1 & 20 \le t \le 39 \\ 8F1BBCDC & 40 \le t \le 59 \\ CA62C1D6 & 60 \le t \le 79 \end{cases} \tag{4.14}$$

4.2.2 Константы в алгоритмах SHA-224 и SHA-256

В алгоритмах SHA-224 и SHA-256 используются 64 константы $K_0^{256}, K_1^{256}, \dots, K_{63}^{256}$. Константы являются 32-битными словами. Каждая константа формируется путём преобразования простого числа (используются первые 64 простых числа). Для этого из простого числа извлекается кубический корень, затем дробная часть результата преобразуется в двоичный вид. Первые 32 бита после запятой используются в качестве константы. Значения констант (в 16-ричном виде) приведены ниже, в порядке слева направо.

```
428A2F98  71374491  B5C0FBCF  E9B5DBA5  3956C25B  59F111F1  923F82A4
AB1C5ED5  D807AA98  12835B01  243185BE  550C7DC3  72BE5D74  80DEB1FE
9BDC06A7  C19BF174  E49B69C1  EFBE4786  0FC19DC6  240CA1CC  2DE92C6F
4A7484AA  5CB0A9DC  76F988DA  983E5152  A831C66D  B00327C8  BF597FC7
C6E00BF3  D5A79147  06CA6351  14292967  27B70A85  2E1B2138  4D2C6DFC
53380D13  650A7354  766A0ABB  81C2C92E  92722C85  A2BFE8A1  A81A664B
C24B8B70  C76C51A3  D192E819  D6990624  F40E3585  106AA070  19A4C116
1E376C08  2748774C  34B0BCB5  391C0CB3  4ED8AA4A  5B9CCA4F  682E6FF3
748F82EE  78A5636F  84C87814  8CC70208  90BEFFFA  A4506CEB  BEF9A3F7
C67178F2
```

4.2.3 Константы в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256

В алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256 используются 80 констант $K_0^{512}, K_1^{512}, \dots, K_{79}^{512}$. Константы являются 64-битными словами. Каждая константа формируется путём преобразования простого числа (используются первые 80 простых чисел). Для этого из простого числа извлекается кубический корень, затем дробная часть результата преобразуется в двоичный вид. Первые 64 бита после запятой используются в качестве константы. Значения констант (в 16-ричном виде) приведены ниже, в порядке слева направо.

[2] Первое число – это корень из 2 делённый на 4 (первые 32 бита после запятой), второе число – это корень из 3 делённый на 4 (первые 32 бита после запятой), 3-е число – корень из 5 делённый на 4 (первые 32 бита после запятой), 4-е число – корень из 10 делённый на 4 (первые 32 бита после запятой) – прим. пер.

```
428A2F98D728AE22  7137449123EF65CD  B5C0FBCFEC4D3B2F  E9B5DBA58189DBBC
3956C25BF348B538  59F111F1B605D019  923F82A4AF194F9B  AB1C5ED5DA6D8118
D807AA98A3030242  12835B0145706FBE  243185BE4EE4B28C  550C7DC3D5FFB4E2
72BE5D74F27B896F  80DEB1FE3B1696B1  9BDC06A725C71235  C19BF174CF692694
E49B69C19EF14AD2  EFBE4786384F25E3  0FC19DC68B8CD5B5  240CA1CC77AC9C65
2DE92C6F592B0275  4A7484AA6EA6E483  5CB0A9DCBD41FBD4  76F988DA831153B5
983E5152EE66DFAB  A831C66D2DB43210  B00327C898FB213F  BF597FC7BEEF0EE4
C6E00BF33DA88FC2  D5A79147930AA725  06CA6351E003826F  142929670A0E6E70
27B70A8546D22FFC  2E1B21385C26C926  4D2C6DFC5AC42AED  53380D139D95B3DF
650A73548BAF63DE  766A0ABB3C77B2A8  81C2C92E47EDAEE6  92722C851482353B
A2BFE8A14CF10364  A81A664BBC423001  C24B8B70D0F89791  C76C51A30654BE30
D192E819D6EF5218  D69906245565A910  F40E35855771202A  106AA07032BBD1B8
19A4C116B8D2D0C8  1E376C085141AB53  2748774CDF8EEB99  34B0BCB5E19B48A8
391C0CB3C5C95A63  4ED8AA4AE3418ACB  5B9CCA4F7763E373  682E6FF3D6B2B8A3
748F82EE5DEFB2FC  78A5636F43172F60  84C87814A1F0AB72  8CC702081A6439EC
90BEFFFA23631E28  A4506CEBDE82BDE9  BEF9A3F7B2C67915  C67178F2E372532B
CA273ECEEA26619C  D186B8C721C0C207  EADA7DD6CDE0EB1E  F57D4F7FEE6ED178
06F067AA72176FBA  0A637DC5A2C898A6  113F9804BEF90DAE  1B710B35131C471B
28DB77F523047D84  32CAAB7B40C72493  3C9EBE0A15C9BEBC  431D67C49C100D4C
4CC5D4BECB3E42B6  597F299CFC657E2A  5FCB6FAB3AD6FAEC  6C44198C4A475817
```

5. Подготовка

Подготовка заключается в дополнении сообщения (подраздел 5.1), разделении сообщения на блоки (подраздел 5.2) и задании начального значения хэша (подраздел 5.3).

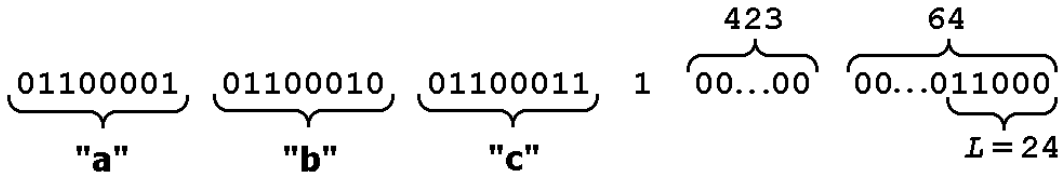
5.1 Дополнение сообщения

Данная операция заключается в увеличении длины сообщения до значения, кратного 512 или 1024 битам (в зависимости от алгоритма). Дополнение может быть выполнено до начала вычисления хэша или в любой другой момент, предшествующий обработке дополненного блока (или блоков).

5.1.1 Дополнение в алгоритмах SHA-1, SHA-224 и SHA-256

Длина исходного сообщения составляет L бит. Сначала к сообщению добавляется двоичная единица. Затем добавляются k нулевых бит, где k – наименьшее неотрицательное решение уравнения $L + 1 + k = 448 \bmod 512$. Затем добавляются 64 бита, являющиеся двоичной записью числа L. Длина дополненного сообщения должна быть кратна 512 битам.

На рисунке ниже приведён пример дополнения сообщения «abc». В 8-битном коде ASCII его длина составляет 8×3 = 24 бита. Сначала к сообщению добавляется двоичная единица, затем 448 - (24 + 1) = 423 нулевых бита, затем 64 бита, представляющие длину сообщения. Длина дополненного сообщения составляет 512 бит[3].



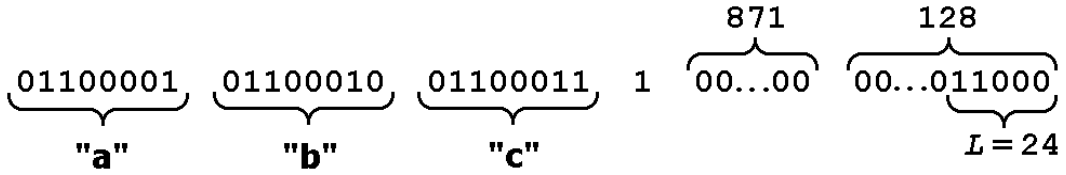
5.1.2 Дополнение в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256

Длина исходного сообщения составляет L бит. Сначала к сообщению добавляется двоичная единица. Затем добавляются k нулевых бит, где k – наименьшее неотрицательное решение уравнения $L + 1 + k = 896 \bmod 1024$. Затем добавляются 128 бит, являющиеся двоичной записью числа L. Длина дополненного сообщения должна быть кратна 1024 битам.

На рисунке ниже приведён пример дополнения сообщения «abc». В 8-битном коде ASCII его длина составляет 8×3 = 24 бита. Сначала к сообщению добавляется двоичная единица, затем

[3] Дополнение выполняется всегда, даже если длина сообщения уже кратна 512 битам. В том числе, если M – пустое сообщение ($L = 0$) – прим. пер.

896 - (24 + 1) = 871 нулевых бит, затем 128 бит, представляющих длину сообщения. Длина дополненного сообщения составляет 1024 бит.



5.2 Разделение на блоки

Дополненное сообщение разделяется на блоки длиной *m* бит. Общее количество блоков обозначается буквой *N*.

5.2.1 Разделение в алгоритмах SHA-1, SHA-224 и SHA-256

В алгоритмах SHA-1, SHA-224 и SHA-256 дополненное сообщение разделяется на блоки $M^1, M^2, \ldots, M^N$ длиной 512 бит. Каждый блок разделяется на шестнадцать 32-битных слов. Крайние слева 32 бита *i*-го блока обозначаются как M_0^i, следующие 32 бита – как M_1^i, и так далее до M_{15}^i.

5.2.2 Разделение в алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256

В алгоритмах SHA-384, SHA-512, SHA-512/224 и SHA-512/256 дополненное сообщение разделяется на блоки $M^1, M^2, \ldots, M^N$ длиной 1024 бит. Каждый блок разделяется на шестнадцать 64-битных слов. Крайние слева 64 бита *i*-го блока обозначаются как M_0^i, следующие 64 бита – как M_1^i, и так далее до M_{15}^i.

5.3 Начальное значение хэша

В каждом алгоритме используется начальное значение хэша, которое является набором заранее известных слов. Количество и разрядность слов зависят от длины хэша.

5.3.1 Начальное значение хэша алгоритма SHA-1

Начальное значение хэша алгоритма SHA-1 приведено ниже в виде пяти 32-битных слов (используется 16-ричная запись).

$$H_0^0 = 67452301$$
$$H_1^0 = EFCDAB89$$
$$H_2^0 = 98BADCFE$$
$$H_3^0 = 10325476$$
$$H_4^0 = C3D2E1F0$$

5.3.2 Начальное значение хэша алгоритма SHA-224

Начальное значение хэша алгоритма SHA-224 приведено ниже в виде восьми 32-битных слов[4] (используется 16-ричная запись).

$$H_0^0 = C1059ED8$$
$$H_1^0 = 367CD507$$
$$H_2^0 = 3070DD17$$
$$H_3^0 = F70E5939$$

[4] Данные слова являются младшими половинами 64-битных слов, образующих начальное значение хэша алгоритма SHA-384 (пункт 5.3.4). Обратите внимание, что длина начального значения хэша составляет не 224, а 256 бит – прим. пер.

$$H_4^0 = \texttt{FFC00B31}$$
$$H_5^0 = \texttt{68581511}$$
$$H_6^0 = \texttt{64F98FA7}$$
$$H_7^0 = \texttt{BEFA4FA4}$$

5.3.3 Начальное значение хэша алгоритма SHA-256

Начальное значение хэша алгоритма SHA-256 приведено ниже в виде восьми 32-битных слов (используется 16-ричная запись).

$$H_0^0 = \texttt{6A09E667}$$
$$H_1^0 = \texttt{BB67AE85}$$
$$H_2^0 = \texttt{3C6EF372}$$
$$H_3^0 = \texttt{A54FF53A}$$
$$H_4^0 = \texttt{510E527F}$$
$$H_5^0 = \texttt{9B05688C}$$
$$H_6^0 = \texttt{1F83D9AB}$$
$$H_7^0 = \texttt{5BE0CD19}$$

Каждое слово формируется путём преобразования простого числа (используются первые восемь простых чисел). Для этого из простого числа извлекается квадратный корень, затем дробная часть результата преобразуется в двоичный вид. Слово формируется из первых 32 бит после запятой.

5.3.4 Начальное значение хэша алгоритма SHA-384

Начальное значение хэша алгоритма SHA-384 приведено ниже в виде восьми 64-битных слов (используется 16-ричная запись).

$$H_0^0 = \texttt{CBBB9D5DC1059ED8}$$
$$H_1^0 = \texttt{629A292A367CD507}$$
$$H_2^0 = \texttt{9159015A3070DD17}$$
$$H_3^0 = \texttt{152FECD8F70E5939}$$
$$H_4^0 = \texttt{67332667FFC00B31}$$
$$H_5^0 = \texttt{8EB44A8768581511}$$
$$H_6^0 = \texttt{DB0C2E0D64F98FA7}$$
$$H_7^0 = \texttt{47B5481DBEFA4FA4}$$

Каждое слово формируется путём преобразования простого числа (используются простые числа, начиная с девятого и заканчивая шестнадцатым). Для этого из простого числа извлекается квадратный корень, затем дробная часть результата преобразуется в двоичный вид. Слово формируется из первых 64 бит после запятой.

5.3.5 Начальное значение хэша алгоритма SHA-512

Начальное значение хэша алгоритма SHA-512 приведено ниже в виде восьми 64-битных слов (используется 16-ричная запись).

$$H_0^0 = \texttt{6A09E667F3BCC908}$$
$$H_1^0 = \texttt{BB67AE8584CAA73B}$$

$$H_2^0 = \texttt{3C6EF372FE94F82B}$$
$$H_3^0 = \texttt{A54FF53A5F1D36F1}$$
$$H_4^0 = \texttt{510E527FADE682D1}$$
$$H_5^0 = \texttt{9B05688C2B3E6C1F}$$
$$H_6^0 = \texttt{1F83D9ABFB41BD6B}$$
$$H_7^0 = \texttt{5BE0CD19137E2179}$$

Каждое слово формируется путём преобразования простого числа (используются первые восемь простых чисел). Для этого из простого числа извлекается квадратный корень, затем дробная часть результата преобразуется в двоичный вид. Слово формируется из первых 64 бит после запятой.

5.3.6 Начальное значение хэша алгоритмов SHA-512/*t*

На основе алгоритма SHA-512 сформирован ряд алгоритмов под общим названием SHA-512/*t*. Конкретному алгоритму соответствует определённое значение целочисленной переменной *t*. Число *t* должно быть положительным целым меньшим 512 и не равным 384. Формирование алгоритма, соответствующего конкретному значению *t*, производится путём изменения в алгоритме SHA-512 начального значения хэша, а также сокращения длины конечного значения хэша до *t* бит.

В данном пункте описана процедура формирования начального значения хэша для любого заданного значения *t*. В описании будут использованы следующие обозначения:

H^{512} начальное значение хэша алгоритма SHA-512 (см. пункт 5.3.5 выше),

H^{512+} вспомогательное начальное значение хэша,

$H^{512/t}$ начальное значение хэша алгоритма SHA-512/*t*.

Начальное значение хэша, соответствующее конкретному значению *t*, формируется за 4 шага.

Шаг 1. Записать название алгоритма в виде последовательности символов (строки). При этом запись числа *t* не должна начинаться с нуля. Например, *t* равное 256 должно быть представлено строкой «256», и не может быть представлено как «0256». При *t* = 256 название алгоритма SHA-512/*t* является 11-символьной строкой «SHA-512/256».

Шаг 2. С помощью ASCII-таблицы преобразовать символьную запись названия алгоритма в последовательность бит. Например, строка «SHA-512/256» преобразуется в последовательность бит `5348412D3531322F323536` (запись 16-ричная).

Шаг 3. Вычислить H^{512+} путём выполнения операции xor между каждым словом H^{512} и 16-ричным числом `A5A5A5A5A5A5A5A5`, т. е. для *i* от 0 до 7 вычислить

$H_i^{512+} = H_i^{512} \oplus A5A5A5A5A5A5A5A5$

Шаг 4. Используя H^{512+} в качестве начального значения хэша алгоритма SHA-512, вычислить

$H^{512/t}$ = SHA-512(<последовательность бит, сформированная на шаге 2>)

Алгоритмы SHA-512/224 (*t* = 224) и SHA-512/256 (*t* = 256) являются официально утверждёнными алгоритмами. Начальное значение хэша для каждого из них приведено ниже. Алгоритмы, соответствующие другим значениям *t*, возможно, будут описаны в SP 800-107 позднее (по мере необходимости).

5.3.6.1 Начальное значение хэша алгоритма SHA-512/224

Начальное значение хэша алгоритма SHA-512/224 приведено ниже в виде восьми 64-битных слов (используется 16-ричная запись).

$$H_0^0 = \texttt{8C3D37C819544DA2}$$
$$H_1^0 = \texttt{73E1996689DCD4D6}$$
$$H_2^0 = \texttt{1DFAB7AE32FF9C82}$$

$$H_3^0 = 679DD514582F9FCF$$
$$H_4^0 = 0F6D2B697BD44DA8$$
$$H_5^0 = 77E36F7304C48942$$
$$H_6^0 = 3F9D85A86A1D36C8$$
$$H_7^0 = 1112E6AD91D692A1$$

Способ получения этих слов описан выше.

5.3.6.2 Начальное значение хэша алгоритма SHA-512/256

Начальное значение хэша алгоритма SHA-512/256 приведено ниже в виде восьми 64-битных слов (используется 16-ричная запись).

$$H_0^0 = 22312194FC2BF72C$$
$$H_1^0 = 9F555FA3C84C64C2$$
$$H_2^0 = 2393B86B6F53B151$$
$$H_3^0 = 963877195940EABD$$
$$H_4^0 = 96283EE2A88EFFE3$$
$$H_5^0 = BE5E1E2553863992$$
$$H_6^0 = 2B0199FC2C85B8AA$$
$$H_7^0 = 0EB72DDC81C52CA2$$

Способ получения этих слов описан выше.

6. Алгоритмы

Алгоритмы описаны не в порядке увеличения длины хэша. Это связано с тем, что некоторые алгоритмы очень похожи.

Алгоритм SHA-224 сформирован на основе алгоритма SHA-256 путём изменения в нём начального значения хэша и сокращения конечного значения хэша до 224 бит. Поэтому алгоритм SHA-256 описан до алгоритма SHA-224.

По той же причине алгоритм SHA-512 описан до алгоритмов SHA-384, SHA-512/224 и SHA-512/256. Алгоритмы SHA-384, SHA-512/224 и SHA-512/256 сформированы путём изменения в алгоритме SHA-512 начального значения хэша и сокращения длины конечного значения хэша. В алгоритме SHA-512/224 конечное значение хэша сокращается до 224 бит, в алгоритме SHA-512/256 – до 256 бит, и в алгоритме SHA-384 – до 384 бит.

В каждый алгоритм разрешается вносить изменения, не влияющие на итоговое значение хэша. В качестве примера в пункте 6.2 приведён модифицированный алгоритм SHA-1. Реализации таких модифицированных алгоритмов будут считаться соответствующими данному стандарту.

6.1 Алгоритм SHA-1

Алгоритм SHA-1 преобразует сообщение M в хэш длиной 160 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{64}$, где L – длина сообщения в битах.

В алгоритме используются функции и константы, описанные в пунктах 4.1.1 и 4.2.1 соответственно. Сложение (+) выполняется по модулю 2^{32}.

Вычисление хэша начинается с дополнения сообщения и разделения его на блоки $M^1, M^2, \dots, M^N$ в соответствии с разделом 5. Затем производится последовательная обработка блоков. Обработка первого блока заключается в преобразовании начального значения хэша H^0, приведённого в пункте 5.3.1, и блока M^1 в новое значение хэша H^1. Остальные блоки обрабатываются аналогично. Обработка очередного i-го блока заключается в преобразовании значения хэша H^{i-1} и блока M^i в новое значение хэша H^i. Значение хэша H^{i-1} является результатом обработки предыдущего блока M^{i-1}. Значение хэша H^i используется при обработке

следующего блока M^{i+1}. Итоговым результатом работы алгоритма является значение хэша H^N, сформированное после обработки блока M^N.

В описании алгоритма используются следующие вспомогательные переменные:
- восемьдесят 32-битных слов W_0, W_1, …, W_{79},
- шесть 32-битных слов *a*, *b*, *c*, *d*, *e* и *T,*
- пять 32-битных слов $H_0^i, H_1^i, \ldots, H_4^i$, используемых для хранения промежуточных значений хэша (верхний индекс обозначает номер промежуточного значения).

Алгоритм SHA-1
Для *i*, начиная с 1 до *N* включительно, выполнить
{ 1. Вычислить вспомогательную последовательность слов W_t

$$W_t = \begin{cases} M_t^i & 0 \le t \le 15 \\ ROTL^1(W_{t-3} \oplus W_{t-8} \oplus W_{t-14} \oplus W_{t-16}) & 16 \le t \le 79 \end{cases}$$

2. Задать значения переменных *a*, *b*, *c*, *d*, *e*, используя (*i* – 1)-ое значение хэша

$$a = H_0^{i-1}$$
$$b = H_1^{i-1}$$
$$c = H_2^{i-1}$$
$$d = H_3^{i-1}$$
$$e = H_4^{i-1}$$

3. Для *t*, начиная с 0 до 79 включительно, выполнить
{ $T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_t$
$e = d$
$d = c$
$c = ROTL^{30}(b)$
$b = a$
$a = T$
}
4. Вычислить *i*-ое значение хэша

$$H_0^i = a + H_0^{i-1}$$
$$H_1^i = b + H_1^{i-1}$$
$$H_2^i = c + H_2^{i-1}$$
$$H_3^i = d + H_3^{i-1}$$
$$H_4^i = e + H_4^{i-1}$$

}

После того как приведённые выше действия будут выполнены все *N* раз (т. е. после обработки блока M^N), формируется[5] итоговое 160-битное значение хэша

$$H_0^N || H_1^N || H_2^N || H_3^N || H_4^N$$

6.2 Альтернативный способ вычисления хэша SHA-1

В описании алгоритма SHA-1 (подраздел 6.1) подразумевается, что вспомогательная последовательность слов W_0, W_1, …, W_{79} является массивом из восьмидесяти 32-битных слов. Использование массива такой длины позволяет уменьшить время вычисления последовательности, потому что индексы слов $W_{t-3}, \ldots, W_{t-16}$ определяются легко.

[5] Знаком «||» обозначается операция объединения двух битовых последовательностей в единую последовательность – прим. пер.

Если объём памяти ограничен, то может быть использован альтернативный способ вычисления хэша, в котором последовательность слов W_t вычисляется с помощью кольцевой очереди. Данная очередь может быть реализована в виде массива из шестнадцати 32-битных слов W_0, W_1, …, W_{15}. Это позволяет уменьшить необходимый объём памяти на шестьдесят четыре 32-битных числа. Однако время вычисления последовательности, скорее всего, увеличится из-за сложности определения индексов слов W_t на шаге 3 алгоритма.

Альтернативный способ приводит к точно такому же результату, что и алгоритм, описанный в подразделе 6.1. Используемое в описании число *MASK* является константой со значением 0000000F (в 16-ричном виде). Как и в подразделе 6.1, сложение выполняется по модулю 2^{32}.

При условии, что дополнение сообщения и разделение его на блоки уже выполнено, вычисление хэша выполняется согласно приведённым ниже действиям.

Альтернативный способ вычисления хэша SHA-1.
Для *i*, начиная с 1 до *N* включительно, выполнить
{ 1. Для *t*, начиная с 0 до 15 включительно, выполнить { $W_t = M_t^i$ }
2. Задать значения переменных *a*, *b*, *c*, *d* и *e*, используя (*i* – 1)-ое значение хэша

$$a = H_0^{i-1}$$
$$b = H_1^{i-1}$$
$$c = H_2^{i-1}$$
$$d = H_3^{i-1}$$
$$e = H_4^{i-1}$$

3. Для *t*, начиная с 0 до 79 включительно, выполнить
{ $s = t \wedge MASK$
Если $t \geq 16$, то { $W_s = ROTL^1(W_{(s+13)\wedge MASK} \oplus W_{(s+8)\wedge MASK} \oplus W_{(s+2)\wedge MASK} \oplus W_s)$ }
$T = ROTL^5(a) + f_t(b,c,d) + e + K_t + W_s$
$e = d$
$d = c$
$c = ROTL^{30}(b)$
$b = a$
$a = T$
}
4. Вычислить *i*-ое значение хэша

$$H_0^i = a + H_0^{i-1}$$
$$H_1^i = b + H_1^{i-1}$$
$$H_2^i = c + H_2^{i-1}$$
$$H_3^i = d + H_3^{i-1}$$
$$H_4^i = e + H_4^{i-1}$$

}

После того как приведённые выше действия будут выполнены все *N* раз (т. е. после обработки блока M^N), формируется итоговое 160-битное значение хэша

$$H_0^N || H_1^N || H_2^N || H_3^N || H_4^N$$

6.3 Алгоритм SHA-256

Алгоритм SHA-256 преобразует сообщение *M* в хэш длиной 256 бит. Длина сообщения должна удовлетворять условию $0 \leq L < 2^{64}$, где *L* – длина сообщения в битах.

В алгоритме используются функции и константы, описанные в пунктах 4.1.2 и 4.2.2 соответственно. Сложение (+) выполняется по модулю 2^{32}.

Вычисление хэша начинается с дополнения сообщения и разделения его на блоки $M^1, M^2, \dots, M^N$ в соответствии с разделом 5. Затем производится последовательная обработка блоков. Обработка первого блока заключается в преобразовании начального значения хэша H^0, приведённого в пункте 5.3.3, и блока M^1 в новое значение хэша H^1. Остальные блоки обрабатываются аналогично. Обработка очередного *i*-го блока заключается в преобразовании значения хэша H^{i-1} и блока M^i в новое значение хэша H^i. Значение хэша H^{i-1} является результатом обработки предыдущего блока M^{i-1}. Значение хэша H^i используется при обработке следующего блока M^{i+1}. Итоговым результатом работы алгоритма является значение хэша H^N, сформированное после обработки блока M^N.

В описании алгоритма используются следующие вспомогательные переменные:

- шестьдесят четыре 32-битных слова W_0, W_1, …, W_{63},
- десять 32-битных слов a, b, c, d, e, f, g, h, T_1, T_2,
- восемь 32-битных слов $H_0^i, H_1^i, \dots, H_7^i$, используемых для хранения промежуточных значений хэша (верхний индекс обозначает номер промежуточного значения).

Алгоритм SHA-256

Для *i*, начиная с 1 до *N* включительно, выполнить

{ 1. Вычислить вспомогательную последовательность слов W_t

$$W_t = \begin{cases} M_t^i & 0 \le t \le 15 \\ \sigma_1^{256}(W_{t-2}) + W_{t-7} + \sigma_0^{256}(W_{t-15}) + W_{t-16} & 16 \le t \le 63 \end{cases}$$

2. Задать значения переменных *a*, *b*, *c*, *d*, *e*, *f*, *g* и *h*, используя (*i* – 1)-ое значение хэша

$$a = H_0^{i-1}$$
$$b = H_1^{i-1}$$
$$c = H_2^{i-1}$$
$$d = H_3^{i-1}$$
$$e = H_4^{i-1}$$
$$f = H_5^{i-1}$$
$$g = H_6^{i-1}$$
$$h = H_7^{i-1}$$

3. Для *t*, начиная с 0 до 63 включительно, выполнить

{ $T_1 = h + \sum_1^{256}(e) + Ch(e,f,g) + K_t^{256} + W_t$

$T_2 = \sum_0^{256}(a) + Maj(a,b,c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

4. Вычислить *i*-ое значение хэша

$$H_0^i = a + H_0^{i-1}$$
$$H_1^i = b + H_1^{i-1}$$

$$H_2^i = c + H_2^{i-1}$$
$$H_3^i = d + H_3^{i-1}$$
$$H_4^i = e + H_4^{i-1}$$
$$H_5^i = f + H_5^{i-1}$$
$$H_6^i = g + H_6^{i-1}$$
$$H_7^i = h + H_7^{i-1}$$

}

После того как приведённые выше действия будут выполнены все N раз (т. е. после обработки блока M^N), формируется итоговое 256-битное значение хэша

$$H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N||H_7^N$$

6.4 Алгоритм SHA-224

Алгоритм SHA-224 преобразует сообщение M в хэш длиной 224 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{64}$, где L – длина сообщения в битах.

Алгоритм SHA-224 сформирован на основе алгоритма SHA-256 (подраздел 6.3) путём внесения в него следующих двух изменений:

1. В качестве начального значения хэша H^0 используется значение, приведённое в пункте 5.3.2
2. После преобразования значения H^N в последовательность бит $H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N||H_7^N$, производится сокращение длины последовательности до 224 бит (отбрасываются биты справа). Итоговым значением хэша является последовательность $H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N$

6.5 Алгоритм SHA-512

Алгоритм SHA-512 преобразует сообщение M в хэш длиной 512 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{128}$, где L – длина сообщения в битах.

В алгоритме используются функции и константы, описанные в пунктах 4.1.3 и 4.2.3 соответственно. Сложение (+) выполняется по модулю 2^{64}.

Вычисление хэша начинается с дополнения сообщения и разделения его на блоки $M^1, M^2, \dots, M^N$ в соответствии с разделом 5. Затем производится последовательная обработка блоков. Обработка первого блока заключается в преобразовании начального значения хэша H^0, приведённого в пункте 5.3.5, и блока M^1 в новое значение хэша H^1. Остальные блоки обрабатываются аналогично. Обработка очередного i-го блока заключается в преобразовании значения хэша H^{i-1} и блока M^i в новое значение хэша H^i. Значение хэша H^{i-1} является результатом обработки предыдущего блока M^{i-1}. Значение хэша H^i используется при обработке следующего блока M^{i+1}. Итоговым результатом работы алгоритма является значение хэша H^N, сформированное после обработки блока M^N.

В описании алгоритма используются следующие вспомогательные переменные:

- восемьдесят 64-битных слов $W_0, W_1, \dots, W_{79}$,
- десять 64-битных слов $a, b, c, d, e, f, g, h, T_1, T_2$,
- восемь 64-битных слов $H_0^i, H_1^i, \dots, H_7^i$, используемых для хранения промежуточных значений хэша (верхний индекс обозначает номер промежуточного значения).

Алгоритм SHA-512

Для *i*, начиная с 1 до *N* включительно, выполнить

{ 1. Вычислить вспомогательную последовательность слов W_t

$$W_t = \begin{cases} M_t^i & 0 \le t \le 15 \\ \sigma_1^{512}(W_{t-2}) + W_{t-7} + \sigma_0^{512}(W_{t-15}) + W_{t-16} & 16 \le t \le 79 \end{cases}$$

2. Задать значения переменных *a*, *b*, *c*, *d*, *e*, *f*, *g* и *h*, используя (*i* – 1)-ое значение хэша

$$a = H_0^{i-1}$$
$$b = H_1^{i-1}$$
$$c = H_2^{i-1}$$
$$d = H_3^{i-1}$$
$$e = H_4^{i-1}$$
$$f = H_5^{i-1}$$
$$g = H_6^{i-1}$$
$$h = H_7^{i-1}$$

3. Для *t*, начиная с 0 до 79 включительно, выполнить

{ $T_1 = h + \sum_1^{512}(e) + Ch(e,f,g) + K_t^{512} + W_t$

$T_2 = \sum_0^{512}(a) + Maj(a,b,c)$

$h = g$

$g = f$

$f = e$

$e = d + T_1$

$d = c$

$c = b$

$b = a$

$a = T_1 + T_2$

}

4. Вычислить *i*-ое значение хэша

$$H_0^i = a + H_0^{i-1}$$
$$H_1^i = b + H_1^{i-1}$$
$$H_2^i = c + H_2^{i-1}$$
$$H_3^i = d + H_3^{i-1}$$
$$H_4^i = e + H_4^{i-1}$$
$$H_5^i = f + H_5^{i-1}$$
$$H_6^i = g + H_6^{i-1}$$
$$H_7^i = h + H_7^{i-1}$$

}

После того как приведённые выше действия будут выполнены все *N* раз (т. е. после обработки блока M^N), формируется итоговое 512-битное значение хэша

$$H_0^N || H_1^N || H_2^N || H_3^N || H_4^N || H_5^N || H_6^N || H_7^N$$

6.6 Алгоритм SHA-384

Алгоритм SHA-384 преобразует сообщение M в хэш длиной 384 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{128}$, где L – длина сообщения в битах.

Алгоритм SHA-384 сформирован на основе алгоритма SHA-512 (подраздел 6.5) путём внесения в него следующих двух изменений:

1. В качестве начального значения хэша H^0 используется значение, приведённое в пункте 5.3.4
2. После преобразования значения H^N в последовательность бит $H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N||H_7^N$, производится сокращение длины последовательности до 384 бит (отбрасываются биты справа). Результат является итоговым значением хэша.

6.7 Алгоритм SHA-512/224

Алгоритм SHA-512/224 преобразует сообщение M в хэш длиной 224 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{128}$, где L – длина сообщения в битах.

Алгоритм SHA-512/224 сформирован на основе алгоритма SHA-512 (подраздел 6.5) путём внесения в него следующих двух изменений:

1. В качестве начального значения хэша H^0 используется значение, приведённое в пункте 5.3.6.1.
2. После преобразования значения H^N в последовательность бит $H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N||H_7^N$, производится сокращение длины последовательности до 224 бит (отбрасываются биты справа). Результат является итоговым значением хэша.

6.8 Алгоритм SHA-512/256

Алгоритм SHA-512/256 преобразует сообщение M в хэш длиной 256 бит. Длина сообщения должна удовлетворять условию $0 \le L < 2^{128}$, где L – длина сообщения в битах.

Алгоритм SHA-512/256 сформирован на основе алгоритма SHA-512 (подраздел 6.5) путём внесения в него следующих двух изменений:

1. В качестве начального значения хэша H^0 используется значение, приведённое в пункте 5.3.6.2.
2. После преобразования значения H^N в последовательность бит $H_0^N||H_1^N||H_2^N||H_3^N||H_4^N||H_5^N||H_6^N||H_7^N$, производится сокращение длины последовательности до 256 бит (отбрасываются биты справа). Итоговым значением хэша является последовательность $H_0^N||H_1^N||H_2^N||H_3^N$

7. Сокращение длины хэша

Для решения некоторых задач может потребоваться хэш, длина которого отличается от приведённых в данном стандарте. В таких случаях допускается формировать хэш нужной длины следующим образом. Сначала с помощью одного из описанных выше алгоритмов вычисляется более длинный, чем нужно хэш. Затем его длина сокращается до необходимой путём отбрасывания бит справа. Рекомендации по сокращению хэша, а также данные о влиянии длины хэша на криптографическую стойкость использующих его криптографических преобразований, приведены в специальной публикации SP 800-107.

Приложение А – дополнительная информация

А.1 Криптографическая стойкость алгоритмов

Криптографическая стойкость алгоритмов SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256 обсуждается в SP 800-107.

А.2 Замечания по реализации

Примеры вычисления хэшей SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256 доступны по ссылке http://csrc.nist.gov/groups/ST/toolkit/examples.html.

А.3 Идентификаторы объектов

Идентификаторы объектов (OID) для алгоритмов SHA-1, SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224 и SHA-512/256 приведены по ссылке http://csrc.nist.gov/groups/ST/crypto_apps_infra/csor/algorithms.html

Приложение Б – ссылки

FIPS 180-3 Национальный институт стандартов и технологий, федеральный стандарт обработки информации 180-3, «Стандарт криптографически стойкого хэша», октябрь 2008

SP 800-57 Специальная публикация национального института стандартов и технологий (SP) 800-57, часть 1, «Рекомендации по управлению ключами: общие положения», черновая версия, май 2011

SP 800-107 Специальная публикация национального института стандартов и технологий (SP) 800-107, «Рекомендации для приложений, использующих утверждённые хэш-алгоритмы», черновая версия, сентябрь 2011.

Приложение В – отличия от FIPS-180-3

В данном приложении приведены отличия данного стандарта от его предыдущей версии (FIPS 180-3).

1. Согласно FIPS 180-3 дополнение сообщения должно производиться до вычисления хэша. В FIPS 180-4 это ограничение устранено. Дополнение может быть выполнено до начала вычисления хэша или в любой другой момент, предшествующий обработке дополненного блока (или блоков).
2. В FIPS 180-4 описаны два новых алгоритма: SHA-512/224 и SHA-512/256. Также описана процедура формирования начального значения хэша алгоритмов SHA-512/*t* для любого значения *t*.

Список замеченных ошибок

дата	тип	изменение	номер страницы[6]
5 сентября 2014	редакционный	заменить «$t < 79$» на «$t \leq 79$»	страница 10, пункт 4.1.1, строка 1

[6] Учтено в тексте перевода. Номер страницы указывает на страницу в английской версии документа – прим. пер.