ВикипедиЯ

SHA-2

Материал из Википедии — свободной энциклопедии

SHA-2 (англ. *Secure Hash Algorithm Version 2* — безопасный алгоритм хеширования, версия 2) — семейство криптографических алгоритмов — однонаправленных хеш-функций, включающее в себя алгоритмы *SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/256* и *SHA-512/224*.

Хеш-функции предназначены для создания «отпечатков» или «дайджестов» для сообщений произвольной длины. Применяются в различных приложениях или компонентах, связанных с защитой информации.

SHA-2	
Создан	2002
Опубликован	2002
Предшественник	SHA-1
Преемник	Keccak
Размер хеша	224, 256, 384 или 512 бит
Число раундов	64 или 80
Тип	семейство хеш-функций

Содержание

История

Хеш-функции *SHA-2* разработаны Агентством национальной безопасности США и опубликованы Национальным институтом стандартов и технологий в федеральном стандарте обработки информации *FIPS PUB 180-2* в августе 2002 года[1]. В этот стандарт также вошла хеш-функция *SHA-1*, разработанная в 1995 году. В феврале 2004 года в *FIPS PUB 180-2* была добавлена *SHA-224*[2]. В октябре 2008 года вышла новая редакция стандарта — *FIPS PUB 180-3*[3]. В марте 2012 года вышла последняя на данный момент редакция *FIPS PUB 180-4*, в которой были добавлены функции *SHA-512/256* и *SHA-512/224*, основанные на SHA-512 (поскольку на 64-битных архитектурах SHA-512 работает быстрее, чем SHA-256)[4].

В июле 2006 года появился стандарт RFC 4634 «Безопасные хеш-алгоритмы США (*SHA* и *HMAC-SHA*)», описывающий *SHA-1* и семейство *SHA-2*.

Агентство национальной безопасности от лица государства выпустило патент на *SHA-2*[5] под лицензией *Royalty-free*[6].

Алгоритм

Общее описание

Хеш-функции семейства *SHA-2* построены на основе структуры Меркла — Дамгарда.

Исходное сообщение после дополнения разбивается на блоки, каждый блок — на 16 слов. Алгоритм пропускает каждый блок сообщения через цикл с 64 или 80 итерациями (раундами). На каждой итерации 2 слова преобразуются, функцию преобразования задают остальные слова. Результаты обработки каждого блока складываются, сумма является значением хеш-функции. Тем не менее, инициализация внутреннего состояния производится результатом обработки предыдущего блока. Поэтому независимо обрабатывать блоки и складывать результаты нельзя. Подробнее — см. псевдокод.

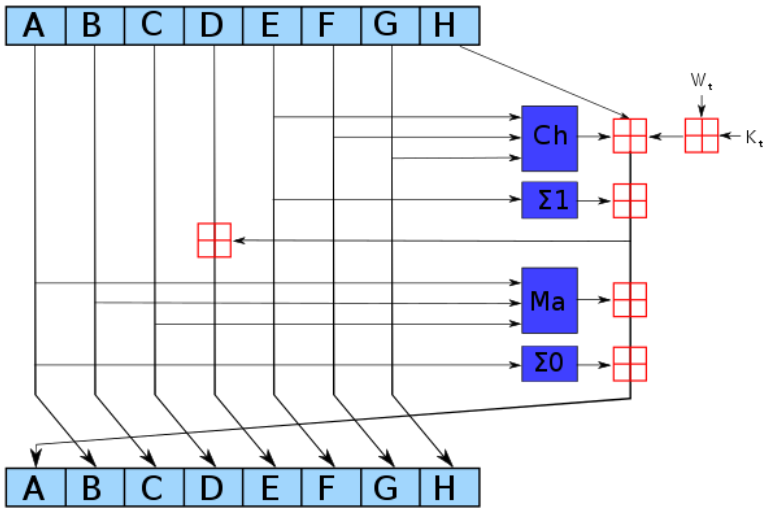

Схема одной итерации алгоритмов SHA-2

Алгоритм использует следующие битовые операции:

- ‖ — конкатенация,
- + — сложение,
- *and* — побитовое «И»,
- *xor* — исключающее «ИЛИ»,
- *shr* (shift right) — логический сдвиг вправо,
- *rotr* (rotate right) — циклический сдвиг вправо.

Сравнение SHA хеш-функций.

В следующей таблице показаны некоторые технические характеристики различных вариантов SHA-2. «Внутреннее состояние» обозначает промежуточную хеш-сумму после обработки очередного блока данных:

Хеш-функция	Длина дайджеста сообщения (бит)	Длина внутреннего состояния (бит)	Длина блока (бит)	Максимальная длина сообщения (бит)	Длина слова (бит)	Количество итераций в цикле	Скорость (MiB/s)[7]
SHA-256, *SHA-224*	256/224	256 (8 x 32)	512	$2^{64} - 1$	32	64	139
SHA-512, *SHA-384*, *SHA-512/256*, *SHA-512/224*	512/384/256/224	512 (8 x 64)	1024	$2^{128} - 1$	64	80	154

Псевдокод

SHA-256

```
Пояснения:
 Все переменные беззнаковые, имеют размер 32 бита и при вычислениях суммируются по модулю 2^32
 message — исходное двоичное сообщение
 m — преобразованное сообщение

 Инициализация переменных
 (первые 32 бита дробных частей квадратных корней первых восьми простых чисел [от 2 до 19]):
h0 := 0x6A09E667
h1 := 0xBB67AE85
h2 := 0x3C6EF372
h3 := 0xA54FF53A
h4 := 0x510E527F
h5 := 0x9B05688C
h6 := 0x1F83D9AB
h7 := 0x5BE0CD19

Таблица констант
(первые 32 бита дробных частей кубических корней первых 64 простых чисел [от 2 до 311]):
k[0..63] :=
   0x428A2F98, 0x71374491, 0xB5C0FBCF, 0xE9B5DBA5, 0x3956C25B, 0x59F111F1, 0x923F82A4, 0xAB1C5ED5,
   0xD807AA98, 0x12835B01, 0x243185BE, 0x550C7DC3, 0x72BE5D74, 0x80DEB1FE, 0x9BDC06A7, 0xC19BF174,
   0xE49B69C1, 0xEFBE4786, 0x0FC19DC6, 0x240CA1CC, 0x2DE92C6F, 0x4A7484AA, 0x5CB0A9DC, 0x76F988DA,
   0x983E5152, 0xA831C66D, 0xB00327C8, 0xBF597FC7, 0xC6E00BF3, 0xD5A79147, 0x06CA6351, 0x14292967,
   0x27B70A85, 0x2E1B2138, 0x4D2C6DFC, 0x53380D13, 0x650A7354, 0x766A0ABB, 0x81C2C92E, 0x92722C85,
   0xA2BFE8A1, 0xA81A664B, 0xC24B8B70, 0xC76C51A3, 0xD192E819, 0xD6990624, 0xF40E3585, 0x106AA070,
   0x19A4C116, 0x1E376C08, 0x2748774C, 0x34B0BCB5, 0x391C0CB3, 0x4ED8AA4A, 0x5B9CCA4F, 0x682E6FF3,
   0x748F82EE, 0x78A5636F, 0x84C87814, 0x8CC70208, 0x90BEFFFA, 0xA4506CEB, 0xBEF9A3F7, 0xC67178F2

Предварительная обработка:
m := message ‖ [единичный бит]
m := m ‖ [k нулевых бит], где k — наименьшее неотрицательное число, такое что
                (L + 1 + K) mod 512 = 448, где L — число бит в сообщении (сравнима по модулю 512 с 448)
m := m ‖ Длина(message) — длина исходного сообщения в битах в виде 64-битного числа
           с порядком байтов от старшего к младшему

Далее сообщение обрабатывается последовательными порциями по 512 бит:
разбить сообщение на куски по 512 бит
для каждого куска
    разбить кусок на 16 слов длиной 32 бита (с порядком байтов от старшего к младшему внутри слова): w[0..15]

    Сгенерировать дополнительные 48 слов:
    для i от 16 до 63
        s0 := (w[i-15] rotr 7) xor (w[i-15] rotr 18) xor (w[i-15] shr 3)
        s1 := (w[i-2] rotr 17) xor (w[i-2] rotr 19) xor (w[i-2] shr 10)
        w[i] := w[i-16] + s0 + w[i-7] + s1

    Инициализация вспомогательных переменных:
    a := h0
    b := h1
    c := h2
    d := h3
    e := h4
    f := h5
    g := h6
    h := h7

    Основной цикл:
    для i от 0 до 63
        Σ0 := (a rotr 2) xor (a rotr 13) xor (a rotr 22)
        Ma := (a and b) xor (a and c) xor (b and c)
        t2 := Σ0 + Ma
        Σ1 := (e rotr 6) xor (e rotr 11) xor (e rotr 25)
        Ch := (e and f) xor ((not e) and g)
        t1 := h + Σ1 + Ch + k[i] + w[i]

        h := g
        g := f
        f := e
        e := d + t1
        d := c
        c := b
        b := a
        a := t1 + t2
```

```
    Добавить полученные значения к ранее вычисленному результату:
    h0 := h0 + a
    h1 := h1 + b
    h2 := h2 + c
    h3 := h3 + d
    h4 := h4 + e
    h5 := h5 + f
    h6 := h6 + g
    h7 := h7 + h

Получить итоговое значение хеша:
digest = hash = h0 ‖ h1 ‖ h2 ‖ h3 ‖ h4 ‖ h5 ‖ h6 ‖ h7
```

SHA-224 идентичен *SHA-256*, за исключением:

- для инициализации переменных h0—h7 используются другие начальные значения,
- в итоговом хеше опускается значение h7.

```
Начальные значения переменных h0–h7 в SHA-224:
h0 := 0xC1059ED8
h1 := 0x367CD507
h2 := 0x3070DD17
h3 := 0xF70E5939
h4 := 0xFFC00B31
h5 := 0x68581511
h6 := 0x64F98FA7
h7 := 0xBEFA4FA4
```

SHA-512 имеет идентичную структуру, но:

- слова имеют длину 64 бита,
- используется 80 раундов вместо 64,
- сообщение разбито на чанки по 1024 бит,
- начальные значения переменных и константы расширены до 64 бит,
- постоянные для каждого из 80 раундов — 80 первых простых чисел,
- сдвиг в операциях `rotr` и `shr` производится на другое число позиций.

```
Начальные значения переменных h0–h7 в SHA-512:
h0 := 0x6a09e667f3bcc908,
h1 := 0xbb67ae8584caa73b,
h2 := 0x3c6ef372fe94f82b,
h3 := 0xa54ff53a5f1d36f1,
h4 := 0x510e527fade682d1,
h5 := 0x9b05688c2b3e6c1f,
h6 := 0x1f83d9abfb41bd6b,
h7 := 0x5be0cd19137e2179
```

SHA-384 идентичен SHA-512, за исключением:

- переменные h0—h7 имеют другие начальные значения,
- в итоговом хеше опускаются значения h6 и h7.

```
Начальные значения переменных h0–h7 в SHA-384
(первые 64 бита дробных частей квадратных корней простых чисел с 9-го по 16-е [от 23 до 53]):
h0 := CBBB9D5DC1059ED8
h1 := 629A292A367CD507
h2 := 9159015A3070DD17
h3 := 152FECD8F70E5939
h4 := 67332667FFC00B31
h5 := 8EB44A8768581511
h6 := DB0C2E0D64F98FA7
h7 := 47B5481DBEFA4FA4
```

SHA-512/256 идентичен SHA-512, за исключением:

- переменные h0—h7 имеют другие начальные значения,
- итоговый хеш обрезается до левых 256 бит.

```
Начальные значения переменных h0–h7 в SHA-512/256:
h0 := 22312194FC2BF72C
h1 := 9F555FA3C84C64C2
h2 := 2393B86B6F53B151
h3 := 963877195940EABD
h4 := 96283EE2A88EFFE3
h5 := BE5E1E2553863992
h6 := 2B0199FC2C85B8AA
h7 := 0EB72DDC81C52CA2
```

SHA-512/224 идентичен SHA-512, за исключением:

- переменные h0—h7 имеют другие начальные значения,
- итоговый хеш обрезается до левых 224 бит.

```
Начальные значения переменных h0–h7 в SHA-512/224:
h0 := 8C3D37C819544DA2
h1 := 73E1996689DCD4D6
h2 := 1DFAB7AE32FF9C82
h3 := 679DD514582F9FCF
h4 := 0F6D2B697BD44DA8
h5 := 77E36F7304C48942
h6 := 3F9D85A86A1D36C8
h7 := 1112E6AD91D692A1
```

Примеры

Ниже приведены примеры хешей *SHA-2*. Для всех сообщений подразумевается использование кодировки ASCII.

```
SHA-224("The quick brown fox jumps over the lazy dog")
 = 730E109B D7A8A32B 1CB9D9A0 9AA2325D 2430587D DBC0C38B AD911525
```

```
SHA-256("The quick brown fox jumps over the lazy dog")
 = D7A8FBB3 07D78094 69CA9ABC B0082E4F 8D5651E4 6D3CDB76 2D02D0BF 37C9E592
```

```
SHA-384("The quick brown fox jumps over the lazy dog")
 = CA737F10 14A48F4C 0B6DD43C B177B0AF D9E51693 67544C49 4011E331 7DBF9A50
   9CB1E5DC 1E85A941 BBEE3D7F 2AFBC9B1
```

```
SHA-512("The quick brown fox jumps over the lazy dog")
 = 07E547D9 586F6A73 F73FBAC0 435ED769 51218FB7 D0C8D788 A309D785 436BBB64
   2E93A252 A954F239 12547D1E 8A3B5ED6 E1BFD709 7821233F A0538F3D B854FEE6
```

```
SHA-512/256("The quick brown fox jumps over the lazy dog")
 = DD9D67B3 71519C33 9ED8DBD2 5AF90E97 6A1EEEFD 4AD3D889 005E532F C5BEF04D
```

```
SHA-512/224("The quick brown fox jumps over the lazy dog")
 = 944CD284 7FB54558 D4775DB0 485A5000 3111C8E5 DAA63FE7 22C6AA37
```

Малейшее изменение сообщения в подавляющем большинстве случаев приводит к совершенно другому хешу вследствие лавинного эффекта. К примеру, при изменении dog на cog получится:

```
SHA-256("The quick brown fox jumps over the lazy cog")
 = E4C4D8F3 BF76B692 DE791A17 3E053211 50F7A345 B46484FE 427F6ACC 7ECC81BE
```

Криптоанализ

В 2003 году Гилберт и Хандшух провели исследование *SHA-2*, но не нашли каких-либо уязвимостей.[8] Однако в марте 2008 года индийские исследователи Сомитра Кумар Санадия и Палаш Саркар опубликовали найденные ими коллизии для 22 итераций *SHA-256* и *SHA-512*.[9] В сентябре того же года они представили метод конструирования коллизий для усечённых вариантов *SHA-2* (21 итерация).[10][11] Позднее были найдены методы конструирования коллизий для 31 итерации *SHA-256*[12] и для 27 итераций *SHA-512*[13].

Криптоанализ хеш-функции подразумевает исследование устойчивости алгоритма по отношению, по меньшей мере, к следующим видам атак:

- нахождение коллизий, то есть разных сообщений с одинаковым хешем,
- нахождение прообраза, то есть неизвестного сообщения по его хешу.

От устойчивости хеш-функции к нахождению коллизий зависит безопасность электронной цифровой подписи с использованием данного хеш-алгоритма. От устойчивости к нахождению прообраза зависит безопасность хранения хешей паролей для целей аутентификации.

Ввиду алгоритмической схожести *SHA-2* с *SHA-1* и наличия у последней потенциальных уязвимостей принято решение, что SHA-3 будет базироваться на совершенно ином алгоритме.[14][15] 2 октября 2012 года NIST утвердил в качестве SHA-3 алгоритм Keccak.

Применение и сертификация

См. также Применение хеширования

SHA-224, *SHA-256*, *SHA-384*, *SHA-512*, *SHA-512/256* и *SHA-512/224* законом США допускаются к использованию в некоторых правительственных приложениях, включая использование в рамках других криптографических алгоритмов и протоколов, для защиты информации, не имеющей грифа секретности. Стандарт также допускает использование *SHA-2* частными и коммерческими организациями.[16]

Хеш-функции *SHA-2* используются для проверки целостности данных и в различных криптографических схемах. На 2008 год семейство хеш-функций *SHA-2* не имеет такого широкого распространения, как *MD5* и *SHA-1*[17], несмотря на обнаруженные у последних недостатки.

Некоторые примеры применения *SHA-2* указаны в таблице:

Область применения	Детали
S/MIME	*SHA-224*, *SHA-256*, *SHA-384* или *SHA-512* дайджесты сообщений[18]
OpenLDAP	*SHA-256*, *SHA-384* или *SHA-512* хеши паролей[19]
DNSSEC	*SHA-256* дайджесты *DNSKEY* в протоколе *DNSSEC*[20]
X.509	*SHA-224*, *SHA-256*, *SHA-384* и *SHA-512* используются для создания электронной цифровой подписи сертификата[21]
PGP	*SHA-256*, *SHA-384*, *SHA-512* используются для создания электронной цифровой подписи[22]
IPSec	Некоторые реализации поддерживают *SHA-256* в протоколах *ESP* и *IKE*[23]
DSA	Семейство *SHA-2* используется для создания электронной цифровой подписи[24]
SHACAL-2	Блочный алгоритм шифрования SHACAL-2 построен на основе хеш-функции *SHA-256*
Bitcoin	Эмиссия криптовалюты Bitcoin осуществляется посредством поиска строк, *SHA-256*-хеш которых имеет заданную структуру

Как показали исследования[25], алгоритмы *SHA-2* работают в 2—3 раза медленнее других популярных хеш-алгоритмов MD5, SHA-1, Tiger и RIPEMD-160.

Сертификация

Реализации *SHA-2*, как и всех Федеральных стандартов обработки информации, могут быть сертифицированы для использования в некоторых приложениях на территории США. Сертификация происходит в рамках процедуры Cryptographic Module Validation Program, которая проводится Национальным институтом стандартов и технологий США совместно с канадским Бюро безопасности связи.

На 5 ноября 2008 года было сертифицировано более 250 реализаций *SHA-2*, четыре из которых могли оперировать сообщениями с длиной в битах, не кратной восьми.[26]

Сертифицировано FIPS PUB 180-4, CRYPTREC и NESSIE.

См. также

- *MD5*
- *SHA-1*
- Коллизия хеш-функции
- Федеральные стандарты обработки информации
- Хеширование

Примечания

1. FIPS PUB 180-2 (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf) (англ.). — первоначальный вариант стандарта для SHA-2. Проверено 19 ноября 2008. Архивировано (https://www.webcitation.org/66GOBWCaL?url=http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf) 18 марта 2012 года.
2. FIPS PUB 180-2 with change notice (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf) (англ.). — вариант стандарта с SHA-224. Проверено 19 ноября 2008. Архивировано (https://www.webcitation.org/66GOBwst5?url=http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf) 18 марта 2012 года.
3. FIPS PUB 180-3 (http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf) (англ.). — редакция Secure Hash Standard от октября 2008 года. Проверено 19 ноября 2008. Архивировано (https://www.webcitation.org/66GOCOl1l?url=http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf) 18 марта 2012 года.
4. FIPS PUB 180-4 (http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf) (англ.). — редакция Secure Hash Standard от августа 2015 года. Проверено 28 августа 2015.
5. US patent 6829355 (http://www.google.com/patents/US6829355) (англ.). — Device for and method of one-way cryptographic hashing.
6. «Licensing Declaration for US patent 6829355. (https://datatracker.ietf.org/ipr/858/)». Проверено 2008-02-17. (англ.)
7. "Crypto++ 5.6.0 Benchmarks". Retrieved 2013-06-13. (http://www.cryptopp.com/benchmarks-amd64.html).
8. *Gilbert H., Handschuh H.* Security Analysis of SHA-256 and Sisters // *Selected Areas in Cryptography*: *10th Annual International Workshop, SAC 2003, Ottawa, Canada, August 14-15, 2003. Revised Papers* / M. Matsui, R. J. Zuccherato — Springer Berlin Heidelberg, 2004. — P. 175–193. — (Lecture Notes in Computer Science; Vol. 3006) — ISBN 978-3-540-21370-3 — ISSN 0302-9743 (https://www.worldcat.org/issn/0302-9743) — doi:10.1007/978-3-540-24654-1_13 (http://dx.doi.org/10.1007/978-3-540-24654-1_13)
9. Somitra Kumar Sanadhya, Palash Sarkar. 22-Step Collisions for SHA-2 (http://arxiv.org/abs/0803.1220) (англ.)
10. Somitra Kumar Sanadhya, Palash Sarkar. Deterministic Constructions of 21-Step Collisions for the SHA-2 Hash Family (https://dx.doi.org/10.1007/978-3-540-85886-7_17) (англ.)
11. Презентация «Deterministic Constructions of 21-Step Collisions for the SHA-2 Hash Family» (http://isc08.twisc.org/slides/S5P4_Deterministic_Constructions_of_21-Step_Collisions_for_the_SHA-2_Hash_Family.pdf) (англ.)
12. *Mendel F., Nad T., Schläffer M.* Improving Local Collisions: New Attacks on Reduced SHA-256 (https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=69018) // *Advances in Cryptology – EUROCRYPT 2013*: *32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings* / T. Johansson, P. Q. Nguyen — Springer Berlin Heidelberg, 2013. — P. 262–278. — 736 p. — (Lecture Notes in Computer Science; Vol. 7881) — ISBN 978-3-642-38347-2 — ISSN 0302-9743 (https://www.worldcat.org/issn/0302-9743) — doi:10.1007/978-3-642-38348-9_16 (http://dx.doi.org/10.1007/978-3-642-38348-9_16)

13. Christoph Dobraunig, Maria Eichlseder, and Florian Mendel (2016). «Analysis of SHA-512/224 and SHA-512/256 (https://eprint.iacr.org/2016/374.pdf)».
14. Schneier on Security: NIST Hash Workshop Liveblogging (5) (http://www.schneier.com/blog/archives/2005/11/nist_hash_works_4.html) (англ.)
15. Hash cracked — heise Security (http://www.heise-online.co.uk/security/Hash-cracked--/features/75686/2) (англ.)
16. *FIPS 180-2: Secure Hash Standard (SHS): 6. Applicability* (http://csrc.nist.gov/publications/fips/fips180-2/fips180-2withchangenotice.pdf) (англ.)
17. SHA-1 (http://www.google.com/search?q=SHA-1), SHA-256 (http://www.google.com/search?q=SHA-256) в результатах поисковой системы Google
18. draft-ietf-smime-sha2-08 (http://tools.ietf.org/html/draft-ietf-smime-sha2-08) (англ.): Using SHA2 Algorithms with Cryptographic Message Syntax
19. SHA-2 hash support in OpenLDAP (http://www.openldap.org/its/index.cgi/Contrib?id=5660) (англ.)
20. RFC 4509: Use of *SHA-256* in DNSSEC Delegation Signer (DS) Resource Records (RRs)
21. RFC 4055: Additional Algorithms and Identifiers for RSA Cryptography for use in the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile
22. RFC 4880: OpenPGP Message Format
23. Overview of Windows Vista Service Pack 1: New Standards (https://technet.microsoft.com/en-us/library/cc749132.aspx) (англ.)
24. FIPS-186-2 (http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf) Архивировано (https://web.archive.org/web/20090518185707/http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf) 18 мая 2009 года.: Digital Signature Standard (DSS)]
25. Speed Comparison of Popular Crypto Algorithms [1] (http://www.cryptopp.com/benchmarks.html) (англ.)
26. SHS Validation List (http://csrc.nist.gov/groups/STM/cavp/documents/shs/shaval.htm) (англ.)

Литература

- *Лапонина О.Р.* Криптографические основы безопасности (http://www.intuit.ru/department/security/networksec/9/2.html). — М.: Интернет-университет информационных технологий - ИНТУИТ.ру, 2004. — С. 320. — ISBN 5-9556-00020-5.
- *Нильс Фергюсон, Брюс Шнайер.* Практическая криптография = Practical Cryptography: Designing and Implementing Secure Cryptographic Systems. — М. : Диалектика, 2004. — 432 с. — 3000 экз. — ISBN 5-8459-0733-0, ISBN 0-4712-2357-3.
- Анализ усечённого варианта SHA-256 (https://online.tu-graz.ac.at/tug_online/voe_main2.getvolltext?pDocumentNr=85215) (недоступная ссылка) (англ.)
- Коллизии усечённого варианта SHA-256 (https://dx.doi.org/10.1007/978-3-540-71039-4_1) (англ.)
- Нелинейные атаки на усечённые варианты хеш-функций SHA-2 (https://dx.doi.org/10.1007/978-3-540-70500-0_19) (англ.)
- Детерминированное конструирование коллизий для семейства хешей SHA-2 с 21 итерацией (https://dx.doi.org/10.1007/978-3-540-85886-7_17) (англ.)

Ссылки

- FIPS 180-3 (http://csrc.nist.gov/publications/fips/fips180-3/fips180-3_final.pdf) : Secure Hash Standard (SHS)
- RFC 3874: A 224-bit One-way Hash Function: SHA-224
- RFC 4634: US Secure Hash Algorithms (SHA and HMAC-SHA)

Источник — https://ru.wikipedia.org/w/index.php?title=SHA-2&oldid=92682893