NIST đã phát triển thuật toán SHA-3 có tính bảo mật cao hơn và thay thế cho SHA-1, SHA-2.

**Comparison of SHA functions**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | |
| **Algorithm and variant** | | **Output size (bits)** | **Internal state size (bits)** | **Block size (bits)** | **Rounds** | **Operations** | **Security (in bits) against**[**collision attacks**](https://en.wikipedia.org/wiki/Collision_attack) | **Capacity against**[**length extension attacks**](https://en.wikipedia.org/wiki/Length_extension_attack) | **Performance on [Skylake](https://en.wikipedia.org/wiki/Skylake_(microarchitecture)" \o "Skylake (microarchitecture)) (median [cpb](https://en.wikipedia.org/wiki/Cycles_per_byte" \o "Cycles per byte))**[[39]](https://en.wikipedia.org/wiki/SHA-2#cite_note--39) | | **First published** |
| **long messages** | **8 bytes** |
| [**MD5**](https://en.wikipedia.org/wiki/MD5) (as reference) | | 128 | 128 (4 × 32) | 512 | 64 | And, Xor, Rot, Add (mod 232),Or | ≤18 (collisions found)[[40]](https://en.wikipedia.org/wiki/SHA-2#cite_note-40) | 0 | 4.99 | 55.00 | 1992 |
| [**SHA-0**](https://en.wikipedia.org/wiki/SHA-0) | | 160 | 160 (5 × 32) | 512 | 80 | And, Xor, Rot, Add (mod 232),Or | <34 (collisions found) | 0 | ≈ SHA-1 | ≈ SHA-1 | 1993 |
| [**SHA-1**](https://en.wikipedia.org/wiki/SHA-1) | | <63 (collisions found[[41]](https://en.wikipedia.org/wiki/SHA-2#cite_note-41)) | 3.47 | 52.00 | 1995 |
| **SHA-2** | *SHA-224* *SHA-256* | 224 256 | 256 (8 × 32) | 512 | 64 | And, Xor, Rot, Add (mod 232),Or, Shr | 112  128 | 32 0 | 7.62 7.63 | 84.50 85.25 | 2004 2001 |
| *SHA-384* *SHA-512* | 384 512 | 512 (8 × 64) | 1024 | 80 | And, Xor, Rot, Add (mod 264),Or, Shr | 192 256 | 128 (≤ 384) 0 | 5.12 5.06 | 135.75 135.50 | 2001 |
| *SHA-512/224* *SHA-512/256* | 224 256 | 112 128 | 288 256 | ≈ SHA-384 | ≈ SHA-384 | 2012 |
| [**SHA-3**](https://en.wikipedia.org/wiki/SHA-3) | *SHA3-224* *SHA3-256* *SHA3-384* *SHA3-512* | 224 256 384 512 | 1600 (5 × 5 × 64) | 1152 1088 832 576 | 24[[42]](https://en.wikipedia.org/wiki/SHA-2#cite_note-42) | And, Xor, Rot, Not | 112 128 192 256 | 448 512 768 1024 | 8.12 8.59 11.06 15.88 | 154.25 155.50 164.00 164.00 | 2015 |
| *SHAKE128* *SHAKE256* | *d* (arbitrary) *d* (arbitrary) | 1344 1088 | min(*d*/2, 128) min(*d*/2, 256) | 256 512 | 7.08 8.59 | 155.25 155.50 |

Giải thuật SHA2

SHA2 bao gồm bốn giải thuật SHA-224, SHA-256, SHA-384 và SHA-512.

Ba thuật giải SHA- 256, SHA-384 và SHA-512 được xuất bản lần đầu năm 2001 trong bản phác thảo FIPS PUB 180-2. Năm 2002, FIPS PUB 180-2, FIPS PUB 180-2, bao gồm cả SHA-1 được chấp nhận thành chuẩn chính thức. NĂm 2004, FIPS PUB 180-2 được bổ sung thêm một biến thể-SHA-224,với mục đích tạo ra một biến thể SHA-2 có độ dài khóa trùng với DÉ ba lần với 2 khóa (2TDES)-112 bit. Những biến thể SHA -2 này được đăng kí bằng sang chế hoa kì 6.829.355.

Về giải thuật, các biến thể của SHA-2 không khác nhau. Mặc dù chúng sử dụng giá trị biến và hằng số cũng như độ dài từ,v.v. khác nhau:

* SHA- 224: trả lại kết quả dài 224 bit
* SHA -256: trả lại kết quả dài 256 bit
* SHA- 384: trả lại kết quả dài 384 bit
* SHA- 512: trả lại kết quả dài 512 bit
* SHA-256: được tính toán với các từ 32 bits, 64 vòng lặp, còn SHA-512 được tính toán với các từ 64 bits, 80 vòng lặp. SHA-224 với digest dài 224 bits, là phiên bản rút gọn của SHA-256. SHA -384 với digest dài 384 bits, là phiên bản rút gọn của SHA-512.

Bước 1: Khởi tạo biến

(32 bit đầu tiên của phần phân số của căn bậc 2 của 8 số nguyên tố đầu tiên 2..19):

h0: = 0x6a09e667

h1: = 0xbb67ae85

h2: = 0x3c6ef372

h3: = 0xa54ff53a

h4: = 0x510e527f

h5: = 0x9b05688c

h6: = 0x1f83d9ab

h7: = 0x5be0cd19

Bước 2: khởi tạo hằng số

(32 bit đầu tiên của phần phân số của căn bậc 3 của 64 số nguyên tố đầu tiên 2..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

Bước 3: Tiền xử lý:

Thêm bit ‘1’ vào cuối đoạn dữ liệu gốc

Thêm k bit ‘0’, tròn đó k là số nhỏ nhất >=0 sao cho chiều dài của đoạn dữ liệu gốc (tính bằng bit) đồng dư với 448 (mod 512)

Thêm độ dài của đoạn dữ liệu gốc( trước giai đoạn tiền xử lý), tính bằng bits thể hiện bằng một số 64 bit big endian vào cuối đoạn dữ liệu.

Bước 4: Xử lý đoạn dữ liệu từng 512 bit một:

Tách đoạn dữ liệu ra thành từng nhóm 512 bit

Với mỗi nhóm

Tách nhóm ra thành 16 nhóm 32 bit big endian w[0..15]

- Mở rộng 16 từ đầu tiên vào 48 từ còn lại w [16..63] của mảng lịch trình tin nhắn:

**for** i **từ** 16 đến 63

s0: = (w [i-15] **quay vòng phải** 7) **xor** (w [i-15] **quay vòng phải** 18) **xor** (w [i-15] **dịch phải** 3)

s1: = (w [i-2] **quay vòng phải** 17) **xor** (w [i-2] **quay vòng phải** 19) **xor** (w [i-2] **dịch phải** 10)

w [i]: = w [i-16] **+** s0 **+** w [i-7] **+** s1

Khởi tạo các biến làm việc với giá trị băm hiện tại:

a: = h0

b: = h1

c: = h2

d: = h3

e: = h4

f: = h5

g: = h6

h: = h7

Bước 4: Xử lý đoạn dữ liệu từng 512 bit một

vòng lặp chính:

**for** i **từ** 0 đến 63

S0: = (a **xoay vòng phải**  2) **xor** (a xoay vòng phải 13) **xor** (a xoay vòng phải 22)

maj: = (a and b) **xor** (a and c) xor (b and c).

t2: = s0+maj

S1: = (e **xoay vòng phải**  6) **xor** (e **xoay vòng phải**  11) **xor** (e **xoay vòng phải**  25)

ch: = (e **và** f) **xor** (**(not e)and g**)

t1:=h+s1+ch+k[i]+w[i]

h: = g

g: = f

f: = e

e: = d **+** t1

d: = c

c: = b

b: = a

a: = t1 **+** t2

Cộng giá trị băm vừa tính vào kết quả:

h0: = h0 **+** a

h1: = h1 **+** b

h2: = h2 **+** c

h3: = h3 **+** d

h4: = h4 **+** e

h5: = h5 **+** f

h6: = h6 **+** g

h7: = h7 **+** h

Bước 5: Tạo kết quả cuối cùng (big endian):

digest: = hash: = h0 **chắp thêm** h1 **nối thêm** h2 **nối thêm** h3 **nối thêm** h4 **nối thêm** h5 **nối** h6 **nối thêm** h7