**SHA0:**

SHA-0 băm các thông báo có độ dài bất kỳ trong các khối 512 bit và tạo ra thông báo tổng hợp 160 bit.

1 Chia khối 512-bit Mj thành 16 từ 32-bit W0, W1, ..., W15.

2 Mở rộng từ bằng đệ quy:

Wi =Wi−3 ⊕ Wi−8 ⊕ Wi−14 ⊕ Wi−16 ( 16 ≤ i < 80)

3 Chia hj − 1 cho 5thanh ghi A, B, C, D và E :

h(j-1)=(A0, B0, C0, D0, E0)

4 Lặp 80 lần t=0 -> t=79:



|  |  |  |  |
| --- | --- | --- | --- |
| Round | Step i | fi(x,y,z) | Ki |
| 1 | 1≤i≤20 | fIF =(x∧y) ⨁ (∧z) | 0x5a827999 |
| 2 | 21≤i≤40 | fXOR =x ⨁ y ⨁ z | 0x6ed6eba1 |
| 3 | 41≤i≤60 | fM A J =(x∧y) ∨ (x∧z) ∨ (y∧z) | 0x8fabbcdc |
| 4 | 61≤i≤80 | fXOR =x ⨁ y ⨁ z | 0xca62c1d6 |

5 Đầu ra hàm nén:

hj = (A0 + A80, B0 + B80, C0 + C80, D0 + D80, E0 + E80).

**SHA1:**

Đối với mỗi khối 512 bit của tin nhắn có đệm, chia nó thành 16 thông báo dài 32-bit, (m0, m1, ...., m15), 80 bits từ (W0->W79). Sau đó được mở rộng theo công thức đệ quy:

Wi = ROL1(Wi−3 ⊕ Wi−8 ⊕ Wi−14 ⊕ Wi−16 ( 16 ≤ i < 80)

Khởi tạo năm biến làm việc, a, b, c, d và e, với giá trị băm thứ (i-1)



For t=0 to 79:

|  |  |  |  |
| --- | --- | --- | --- |
| Round | Step i | fi(x,y,z) | Ki |
| 1 | 0≤i≤19 | fIF =(x∧y) ⨁ (∧z) | 0x5a827999 |
| 2 | 20≤i≤39 | fXOR =x ⨁ y ⨁ z | 0x6ed6eba1 |
| 3 | 40≤i≤59 | fM A J =(x∧y) ∨ (x∧z) ∨ (y∧z) | 0x8fabbcdc |
| 4 | 60≤i≤79 | fXOR =x ⨁ y ⨁ z | 0xca62c1d6 |

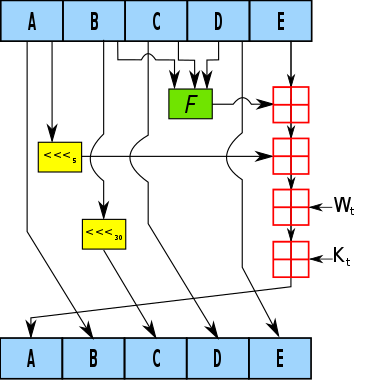


Tính toán giá trị băm trung gian



Kết quả:  N:Số lượng khối trong thư đệm.

Hình 2.1 biểu diễn lưu đồ một vòng xử lý của SHA1, trong đó A, B, C, D, E là các từ 32 bit của state, Wt: khối 32 bit thông điệp đầu vào, Kt là 32 bit hằng khác nhau cho mỗi vòng, <<<n là thao tác dịch trái n bit, ⊞ biểu diễn phép cộng modulo 32 bit và F là hàm phi tuyến tính.



**Hình 2. 1 Lưu đồ một vòng xử lý của SHA-1**

SHA2: gồm 4 giải thuật gồm 6 hàm băm: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA -512/256

**Giải thuật SHA2-256**

Hàm nén SHA-256 hoạt động trên một khối tin nhắn 512-bit và một giá trị băm trung gian 256- bit. Về cơ bản nó là một thuật toán mật mã khối 256-bit mã hóa giá trị băm trung gian sử dụng khối tin làm khóa.

Bước 1:

Với thông điệp dài L bit ta thêm bit 1 vào cuối thông điệp, thêm bit 0 đến khi dữ liệu là bội số 512,rồi bớt đi 64 bit cuối cùng

Thêm 64 bit cuối cùng là đội dài của thông điệp ở dạng nhị phân.

Bước 2:

Bây giờ chúng ta tạo 8 giá trị băm. Đây là các hằng số được mã hóa cứng đại diện cho 32 bit đầu tiên của phần phân số của căn bậc hai của 8 số nguyên tố đầu tiên: 2, 3, 5, 7, 11, 13, 17, 19

h0 := 0x6a09e667

h1 := 0xbb67ae85

h2 := 0x3c6ef372

h3 := 0xa54ff53a

h4 := 0x510e527f

h5 := 0x9b05688c

h6 := 0x1f83d9ab

h7 := 0x5be0cd19

### Bước 3:Khởi tạo Hằng số tròn (k)

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 4: Tách thành nhiều đoạn 512bit.Chia nhỏ tin nhắn thành các khối 512 bit cho mỗi đoạn tạo một mảng lịch thông điệp 64 mục nhập w [0..63] của các từ 32 bit.sao chép đoạn vào 16 từ đầu tiên w [0..15] của mảng lịch thông điệp Mở rộng 16 từ đầu tiên vào 48 từ còn lại w [16..63] mảng thông điệp:

for i from 16 to 63

s0 := (w[i-15] rightrotate 7) xor (w[i-15] rightrotate 18) xor (w[i-15] rightshift 3)

s1 := (w[i-2] rightrotate 17) xor (w[i-2] rightrotate 19) xor (w[i-2] rightshift 10)

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

### Bước 5 - Nén

### Khởi tạo các biến **a, b, c, d, e, f, g, h** và đặt chúng bằng các giá trị băm hiện tại tương ứng. **h0, h1, h2, h3, h4, h5, h6, h7**

Chạy vòng lặp nén. Vòng lặp nén sẽ thay đổi các giá trị của ****… h**** . Vòng lặp nén như sau:

cho tôi từ 0 đến 63

S1 = (e hướng sang phải 6) xor (e hướng sang phải 11) xor (e hướng sang phải 25)

ch = (e và f) xor ((không phải e) và g)

temp1 = h + S1 + ch + k [i] + w [i]

S0 = (hướng sang phải 2) xor (hướng sang phải 13) xor (hướng phải 22)

maj = (a và b) xor (a và c) xor (b và c)

temp2: = S0 + maj

h = g

g = f

f = e

e = d + temp1

d = c

c = b

b = a

a = temp1 + temp2

B6: Thêm đoạn nén vào băm hiện tại giá trị:

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 7: Kết hợp Băm cuối cùng

Cuối cùng nhưng không kém phần quan trọng, hãy ghép tất cả chúng lại với nhau, một phép [nối chuỗi](https://qvault.io/golang/strings-builder-concatenation-golang/" \t "https://qvault.io/cryptography/how-sha-2-works-step-by-step-sha-256/_blank) đơn giản sẽ làm được.

**Giải thuật SHA3-512**

Bước 1. Thêm bit đệm.

Thông điệp được đệm để chiều dài của nó đồng nhất đến 896 modulo 1024. Padding luôn được thêm vào, ngay cả khi tin nhắn đã có chiều dài mong muốn. Như vậy, số padding bit nằm trong phạm vi từ 1 đến 1024. Phần đệm bao gồm một khối đơn bit tiếp theo là số cần thiết của 0 bit.

Bước 2. Thêm chiều dài.

Một khối 128 bit được nối vào thư. Khối này là được coi như là một số nguyên 128-bit không dấu (quan trọng nhất byte đầu tiên) và chứa chiều dài của thông báo ban đầu (trước khi đệm).

Kết quả của hai bước đầu tiên mang lại một thông báo đó là một số nguyên

nhiều 1024 bit chiều dài.

Bước 3. Khởi tạo bộ đệm băm.

Một bộ đệm 512-bit được sử dụng để giữ trung gian và cuối cùng

kết quả của hàm băm. Bộ đệm có thể được biểu diễn dưới dạng 8 thanh ghi 64 bit (a, b, c, d, e, f, g, h). Các thanh ghi này được khởi tạo cho các bit 64-bit sau

a = 6A09E667F3BCC908 e = 510E527FADE682D1

b = BB67AE8584CAA73B f = 9B05688C2B3E6C1F

c = 3C6EF372FE94F82B g = 1F83D9ABFB41BD6B

d = A54FF53A5F1D36F1 h = 5BE0CD19137E2179

Các giá trị này được lưu trữ ở định dạng big-endian, đây là yếu tố quan trọng nhất byte của một từ ở vị trí byte thấp (cực tả). Những từ này là thu được bằng cách lấy 64 bit đầu tiên của các phần phân đoạn của hình vuông

gốc của tám số nguyên tố đầu tiên.

Bước 4. Thông báo quy trình bằng các khối 1024-bit (128 từ).

Tâm của thuật toán là một mô-đun bao gồm 80 vòng; mô-đun này được dán nhãn F trong Hình 11.8. Logic được minh họa trong Hình 11.9.

Mỗi vòng mất như là đầu vào giá trị bộ đệm 512-bit, abcdefgh, và cập nhật

nội dung của bộ đệm.Tại đầu vào vòng đầu tiên, bộ đệm có giá trị

giá trị băm trung gian,. Mỗi vòng sử dụng một giá trị 64-bit,

xuất phát từ khối 1024-bit đang được xử lý. Các giá trị này là

bắt nguồn bằng cách sử dụng lịch trình thông báo được mô tả sau đó. Mỗi vòng cũng sử dụng một hằng số phụ gia, trong đó chỉ ra một trong số

80 vòng. Những từ này đại diện cho 64 bit đầu tiên của các phần phân đoạn của cube của 80 số nguyên tố đầu tiên.Các hằng số cung cấp một "ngẫu nhiên" tập hợp các mẫu 64-bit, sẽ loại bỏ bất kỳ sự thường xuyên nào trong dữ liệu đầu vào. Đầu ra của vòng thứ tám được thêm vào đầu vào cho vòng đầu tiên để sản xuất. Việc bổ sung được thực hiện độc lập cho mỗi một trong tám từ trong bộ đệm với mỗi từ tương ứng trong, sử dụng cộng modulo.

Bước 5. Kết quả đầu ra. Sau khi tất cả các khối 1024-bit đã được xử lý, đầu ra là thông báo thư thoại 512-bit.