TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

AN TOÀN THÔNG TIN- INFORMATION SECURITY

HÀM BĂM BẢO MẬT

Giảng viên: TS. Trần Thế Vinh

HÀM BĂM (HASH FUNCTION)

Khái niệm:

Về cơ bản băm (hashing) là quá trình biến một dữ liệu đầu vào có độ dài bất kỳ (bằng cách sử dụng các công thức toán học) thành một chuỗi đầu ra đặc trưng có độ dài cố định. Hashing được thực hiện thông qua hàm băm (hash function).

- Hàm băm được sử dụng để biểu diễn dữ liệu có kích thước tùy ý ở dạng ngắn, độ dài cố định.
- Hàm băm là không cần chìa khóa và bảo vệ tính toàn vẹn của dữ liệu.
- Theo quy luật, chúng được tạo bằng cách sử dụng các kỹ thuật lặp đặc biệt để xây dựng các hàm băm.

Có các nhóm hàm băm khác nhau như MD, SHA-1, SHA-2, SHA-3, RIPEMD và Whirlpool. Hàm băm thường được sử dụng cho chữ ký số và MAC (Mã xác thực thông báo).

Ví dụ: HMAC.

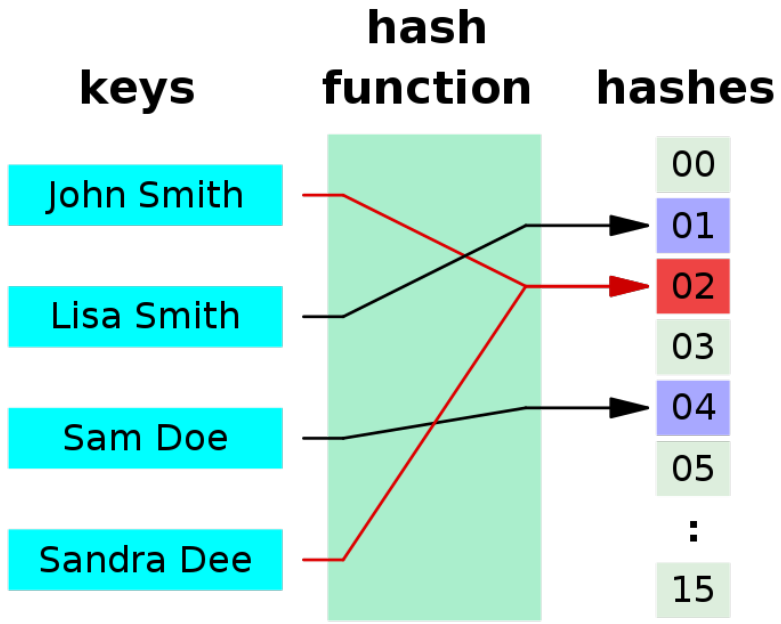
HÀM BĂM (HASH FUNCTION)

Ba đặc điểm chính của hàm băm:

- ✓ **Tính kháng tiền ảnh thứ nhất:** nếu một hàm băm h tạo ra giá trị băm a, thì sẽ làm một quá trình khó khăn để tìm bất kỳ giá trị đầu vào x nào có giá trị băm thành a. Tính chất mã hóa một chiều của hàm băm.

- ✓ **Tính kháng tiền ảnh thứ hai:** nếu một hàm băm h cho đầu vào x tạo ra giá trị băm h(x), thì sẽ khó tìm thấy bất kỳ giá trị đầu vào y nào khác sao cho h(y)=h(x)

- ✓ **Tính kháng va chạm:** đối với một hàm băm h, rất khó để tìm thấy bất kỳ 2 đầu vào khác nhau x, y sao cho h(x)=h(y)



HÀM BĂM (HASH FUNCTION)

Tính bảo mật trong hàm băm:

Cần lưu ý rằng sự tồn tại của các hàm băm không thể đảo ngược chưa được chứng minh, trong đó việc tính toán bất kỳ nguyên mẫu nào của một giá trị nhất định của hàm băm về mặt lý thuyết là không thể.

Cuộc tấn công "birthdays" cho phép tìm xung đột đối với một hàm băm có giá trị độ dài n bit trong trung bình $2^{n/2}$ phép tính của hàm băm. Do đó, một hàm băm $n - bit$ được coi là mật mã mạnh nếu độ phức tạp tính toán của việc tìm kiếm các xung đột đối với nó gần bằng $2^{n/2}$.

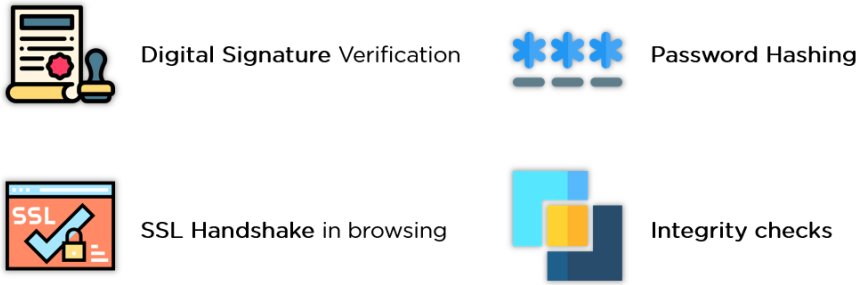
Đối với các hàm băm mật mã, điều quan trọng là ở một thay đổi nhỏ nhất trong đối số, giá trị của hàm sẽ thay đổi rất nhiều (thuộc tính này được gọi là hiệu ứng tuyết lở). Cụ thể, giá trị băm không được rò rỉ thông tin ngay cả về các bit riêng lẻ của đối số. Yêu cầu này là chìa khóa cho khả năng mật mã mạnh của các thuật toán băm mật khẩu của người dùng để lấy khóa.

HÀM BĂM (HASH FUNCTION)

Ứng dụng của hàm băm:

Thuật toán SHA đang được sử dụng ở rất nhiều nơi trên không gian kỹ thuật số, một số ứng dụng của nó như:

- Xác minh chữ ký số: Chữ ký số tuân theo phương pháp mã hóa bất đối xứng để xác minh tích xác thực của dữ liệu. Các thuật toán băm như SHA256 bảo đảm xác minh chữ ký
- Băm mật mã: Giúp thúc đẩy cảm giác riêng tư và giảm tải cho cơ sở dữ liệu trung tâm.
- SSL handshake: "bắt tay" SSL là một phần quan trọng của các phiên duyệt web và nó được thực hiện bằng các hàm SHA. Nó bao gồm các trình duyệt, và các máy chủ web đồng ý về các khóa mã hóa và xác thực băm để chuẩn bị kết nối an toàn.
- Kiểm tra tính toàn vẹn: Việc xác minh tính toàn vẹn của dữ liệu đã được sử dụng thuật toán SHA256. Nó giúp duy trì chức năng giá trị đầy đủ của các dữ liệu và đảm bảo chúng không bị thay đổi trong quá trình truyền.



HÀM BĂM (HASH FUNCTION)

Vì sao cần cấu trúc dữ liệu Hash:

Mỗi ngày, dữ liệu trên internet tăng lên gấp nhiều lần và việc lưu trữ dữ liệu này một cách hiệu quả luôn là một vấn đề khó khăn.
Trong lập trình hàng ngày, lượng dữ liệu này có thể không lớn nhưng vẫn cần được lưu trữ, truy cập và xử lý một cách dễ dàng và hiệu quả. Một cấu trúc dữ liệu rất phổ biến được sử dụng cho mục đích như vậy là cấu trúc dữ liệu Mảng.

Bây giờ câu hỏi đặt ra nếu Mảng đã ở đó thì cần gì cấu trúc dữ liệu mới. Mặc dù việc lưu trữ trong mảng mất T(1) thời gian nhưng việc tìm kiếm trong Mảng mất ít nhất T(log n) thời gian. Khoảng thời gian này có vẻ nhỏ nhưng đối với tập dữ liệu lớn, nó có thể gây ra rất nhiều vấn đề và điều này làm cho cấu trúc dữ liệu Mảng không hiệu quả.

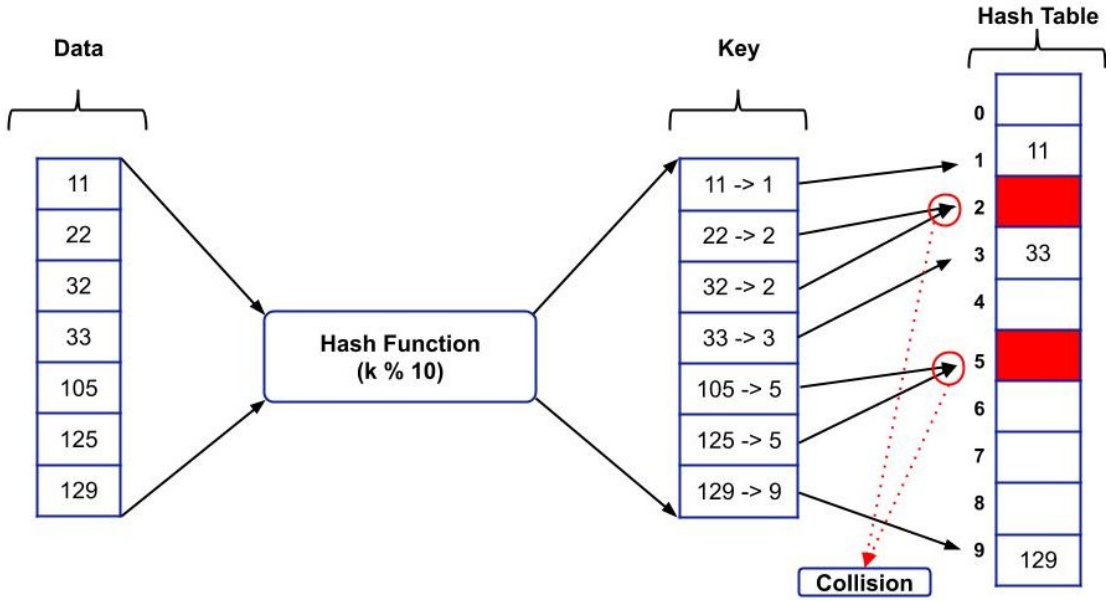
Vì vậy, chúng ta đang tìm kiếm một cấu trúc dữ liệu có thể lưu trữ dữ liệu và tìm kiếm với thời gian không đổi, tức là trong thời gian T(1). Đây là cách cấu trúc dữ liệu Băm phát huy tác dụng. Với sự ra đời của cấu trúc dữ liệu Hash, có thể dễ dàng lưu trữ dữ liệu và truy xuất dữ liệu trong thời gian không đổi.

HÀM BĂM (HASH FUNCTION)

Các thành phần của Hash:

Có ba thành phần chính của băm:

1. **Khóa(key) :** Khóa có thể là bất kỳ chuỗi hoặc số nguyên nào được cung cấp làm đầu vào trong hàm băm, kỹ thuật xác định index hoặc vị trí lưu trữ một mục trong cấu trúc dữ liệu.
2. **Hàm băm(Hash function):** Hàm **băm** nhận khóa đầu vào và trả về chỉ số của một phần tử trong mảng được gọi là bảng băm. Index được gọi là **index băm** .
3. **Bảng băm(Hash Table):** Bảng băm là một cấu trúc dữ liệu ánh xạ các khóa thành các giá trị bằng cách sử dụng một hàm đặc biệt gọi là hàm băm. Hash lưu trữ dữ liệu theo cách kết hợp một mảng trong đó mỗi giá trị dữ liệu có index duy nhất của riêng nó.



HÀM BĂM (HASH FUNCTION)

Băm hoạt động như thế nào?

Giả sử ta có một tập hợp các chuỗi {“ab”, “cd”, “efg”} và ta muốn lưu trữ nó trong một bảng.
Mục tiêu chính của ta ở đây là tìm kiếm hoặc cập nhập các giá trị được lưu trữ trong bảng một cách nhanh chóng trong thời gian T(1) và ta không quan tâm đến thứ tự của các chuỗi trong bảng. Vậy tập hợp các chuỗi đã cho có thể đóng vai trò là khóa và chính chuỗi đó sẽ đóng vai trò là giá trị của chuỗi nhưng làm cách nào để lưu trữ giá trị tương ứng với khóa?

Bước 1: Ta biết rằng, các hàm băm (là một số công thức toán học) được sử dụng để tính giá trị băm đóng vai trò là index của cấu trúc dữ liệu, nơi giá trị sẽ được lưu trữ.

Bước 2: Vì vậy, hãy chọn
“a” = 1,
“b” = 2, ..., ..v..v.. Co tất cả các ký tự chữ cái.

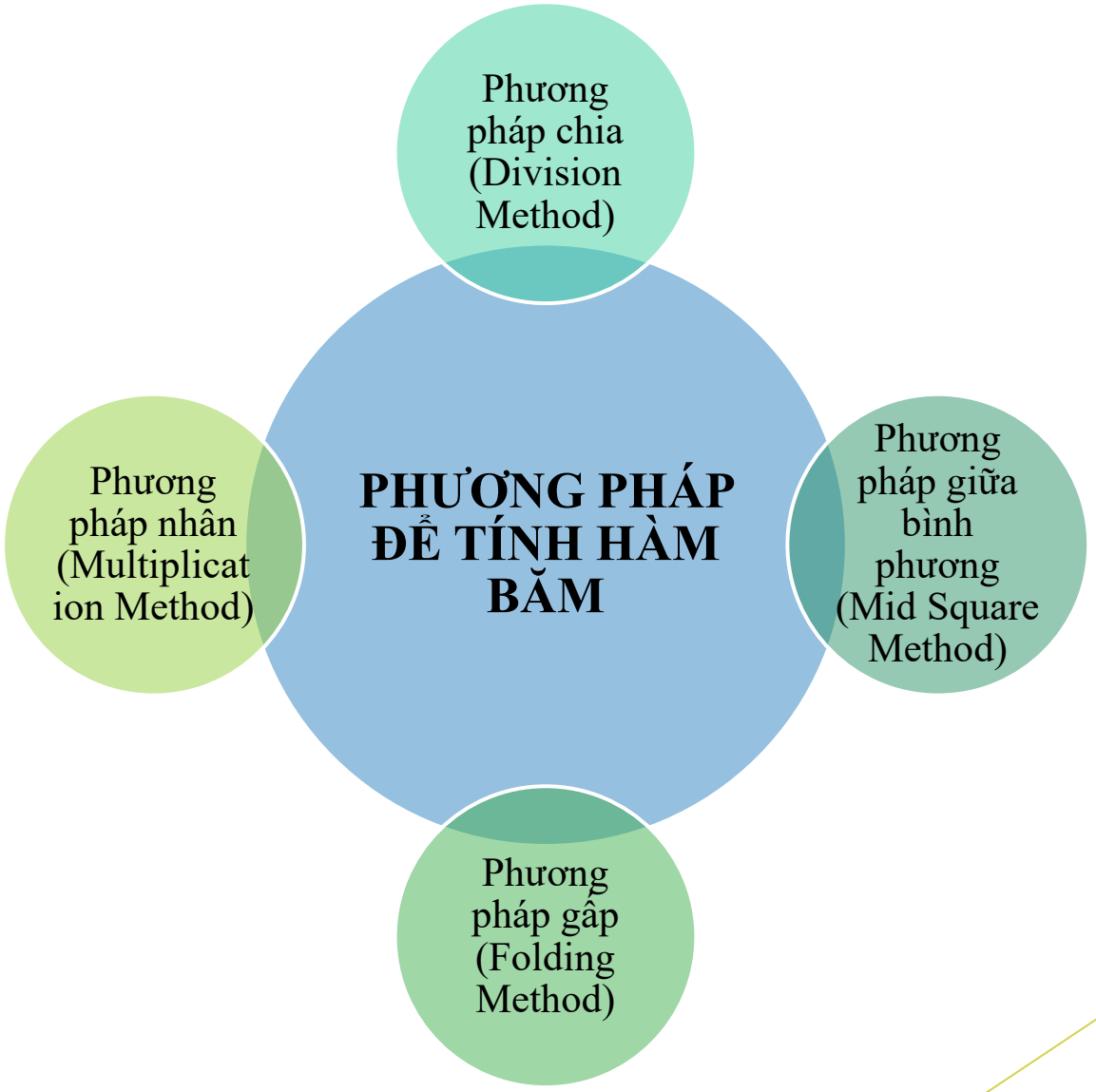
Bước 3: Do đó, gia trị số bằng tổng của tất cả các ký tự của chuỗi.
“ab”=1+2=3
“cd”=3+4=7
“efg”= 5+6+7=18

Bước 4: Bây giờ, giả sử rằng ta có một bảng kích thước 7 để lưu trữ các chuỗi này. Hàm băm được sử dụng ở đây là tổng của các ký tự trong key mod (Table size). Ta có thể tính toán vị trí của chuỗi trong mảng bằng cách lấy sum(string) mod (table size).

Bước 5: Vì vậy, ta sẽ lưu trữ
“ab” : 3 mod 7 =3
“cd” : 7 mod 7 = 0
“efg”: 18 mod 7 = 4.

0	1	2	3	4	5	6
cd			ab	efg		

HÀM BĂM (HASH FUNCTION)



HÀM BĂM (HASH FUNCTION)

1. Phương pháp phân chia:

Đây là phương pháp đơn giản nhất và dễ dàng nhất để tạo giá trị băm. Hàm băm chia giá trị k cho M và sau đó sử dụng phần dư thu được.

Công thức:

$$H(k) = k \bmod M$$

Với:

k là giá trị khóa và M là kích thước bảng băm.

M là một số nguyên tố vì điều đó có thể đảm bảo các khóa được phân bố đồng đều hơn. Hàm băm phụ thuộc vào phần còn lại của phép chia.

Ưu điểm:

- Phương pháp này khá tốt cho bất kỳ giá trị nào của M
- Phương pháp chia rất nhanh vì nó chỉ yêu cầu một phép toán chia duy nhất

Nhược điểm:

- Phương pháp này dẫn đến hiệu suất kém vì các khóa liên tiếp ánh xạ tới các giá trị băm liên tiếp trong bảng băm.
- Đôi khi cần cẩn thận hơn để chọn giá trị M.

HÀM BĂM (HASH FUNCTION)

2. Phương pháp giữa bình phương (Mid Square Method):

Phương pháp giữa bình phương là một phương pháp băm rất tốt. Nó bao gồm 2 bước để tính toán giá trị băm.

1. Bình phương giá trị của khóa k tức là k^2
2. Trích xuất các chữ số x ở giữa làm giá trị băm

Công thức:

$$\mathrm{H}(K) = H(k * k)$$

Với k là giá trị khóa

Giá trị của x có thể được quyết định dựa trên kích thước bảng.

Ưu điểm:

- Hiệu suất của phương pháp này tốt vì hầu hết hoặc tất cả các chữ số của giá trị khóa đóng góp vào kết quả. Điều này là do tất cả các chữ số trong khóa góp phần tạo ra các chữ số ở giữa của kết quả bình phương.
- Kết quả không bị chi phối bởi sự phân phối của chữ số trên cùng hoặc chữ số dưới cùng của bảng giá trị khóa đầu vào.

Nhược điểm:

- Kích thước khóa là một trong những hạn chế của phương pháp này, vì khóa có kích thước lớn thì bình phương của nó sẽ tăng gấp đôi số chữ số.
- Một điều bất lợi nữa là sẽ có va chạm nhưng chúng ta có thể cố gắng giảm bớt va chạm.

Ví dụ:

Giả sử bảng băm có 100 vị trí bộ nhớ. Với r=2 cần có 2 chữ số để ánh xạ khóa tới vị trí bộ nhớ

$$k = 50$$

$$k * k = 2500$$

$$H(K) = 50$$

Giá trị băm thu được là 50

HÀM BĂM (HASH FUNCTION)

3. Phương pháp gấp (Folding Method):

Phương pháp này bao gồm 2 bước:

1. Chia khóa-giá trị k thành một số phần tức là $k1, k2, k3, .., kn$, trong đó mỗi phần có cùng số chữ số trừ phần cuối cùng có thể ít chữ số hơn các phần khác.
2. Thêm các bộ phận riêng rẻ. Giá trị băm có được bằng cách bỏ qu lần mang cuối cùng nếu có.

Công thức:

$$k = k1, k2, \dots, kn$$
$$s = k1 + k2 + \dots + kn$$
$$H(k) = s$$

ở đây: s nhận được bằng các cộng các phần của khóa k.

Lưu ý:

Số lượng chữ số mỗi phần thay đổi tùy thuộc vào kích thước của bảng băm.

Kích thước của bảng băm là 100 thì mỗi phần phải có 2 chữ số trừ phần cuối cùng có thể có số chữ số ít hơn

Ví dụ:

$$k = 1234567$$
$$k1 = 12, k2 = 34, k3 = 56, k4 = 7$$
$$S = k1 + k2 + k3 + k4 = 12 + 34 + 56 + 7 = 109$$
$$H(k) = 109$$

HÀM BĂM (HASH FUNCTION)

4. Phương pháp nhân (Multiplication Method)

Phương pháp này bao gồm các bước sau:

1. Chọn một giá trị không đổi a sao cho 0< a < 1
2. Nhân giá trị khóa với a.
3. Trích phần phân số của k*a.
4. Nhân kết quả của bước trên với kích thước của bảng băm M.
5. Kết quả giá trị băm thu được bằng cách lấy giá trị phần nguyên FLOOR của kết quả thu được ở bước 4.

Công thức:

$$H(K) = FLOOR(M(ka \bmod 1))$$

Ở đây:

M là kích thước bảng băm

k là giá trị khóa

a là giá trị không đổi

Lưu ý: Số lượng chữ số mỗi phần thay đổi tùy thuộc vào kích thước của bảng băm. Ví dụ: kích thước của bảng băm là 100 thì mỗi phần phải có 2 chữ số trừ phần cuối cùng có thể có số chữ số ít hơn.

Ưu điểm:

- Phương pháp nhân có thể hoạt động với bất kỳ giá trị nào từ 0 đến 1, mặc dù có 1 số giá trị có xu hướng cho kết quả tốt hơn các giá trị còn lại.

Nhược điểm:

- Phương pháp nhân nói chung phù hợp khi kích thước là lũy thừa của 2, khi đó toàn bộ quá trình tính toán index bởi khóa sử dụng phép nhân băm là rất nhanh.

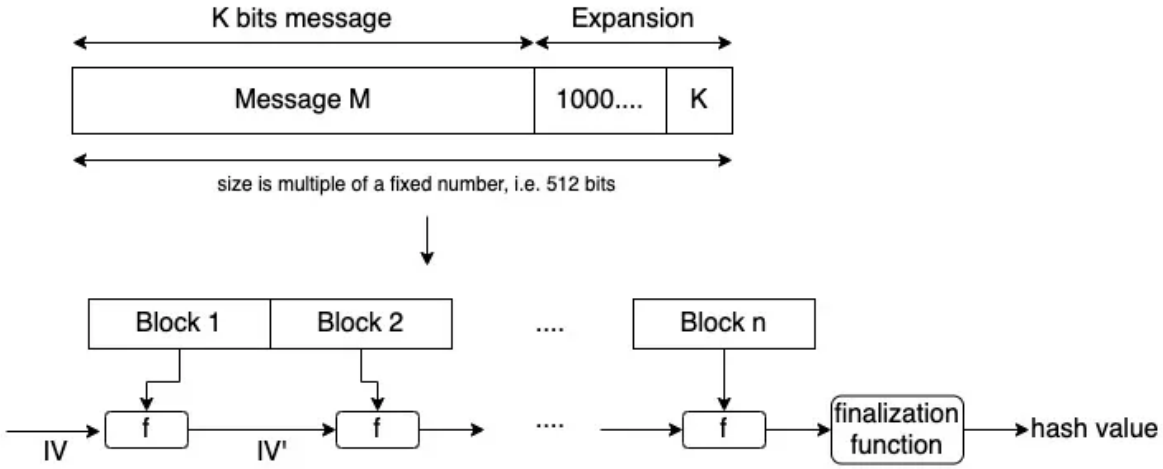
Ví dụ:

$$K = 123456$$
$$A = 0.3536$$
$$M = 100$$
$$H(k) = Floor\big(100(123456 * 0{,}3536 \bmod 1)\big) = Floor\big(100(43654{,}0416 mod 1)\big) = Floor(4{,}16) = 4$$

HÀM BĂM (HASH FUNCTION)

Cấu trúc Merkle-Damgard:

Trong mật mã học, hàm băm Merkle-Damgard là một phương pháp xây dựng các hàm băm mật mã chống va chạm từ các hàm nén một chiều chống va chạm. Cấu trúc này được sử dụng trong thiết kế của nhiều thuật toán băm phổ biến như MD5, SHA-1, SHA-2.



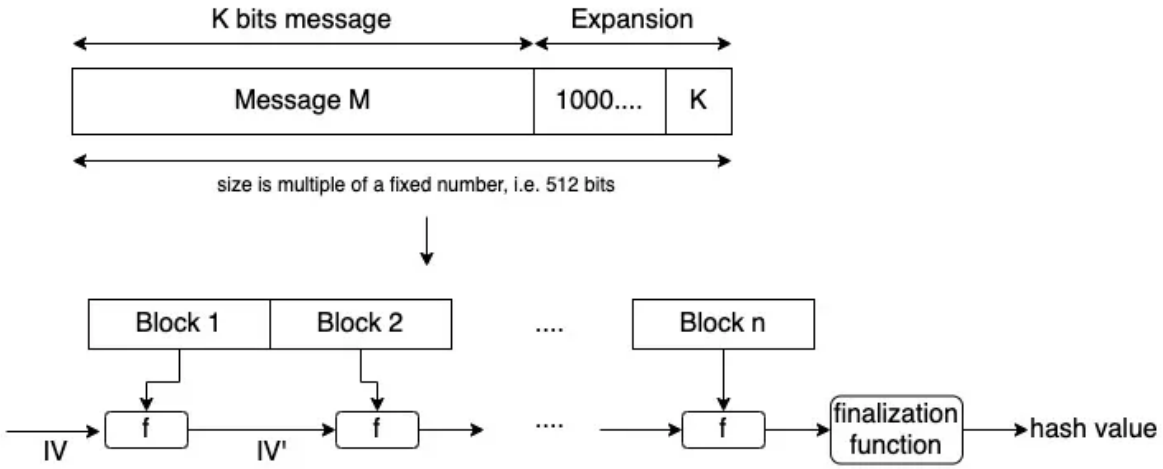
Bước đầu tiên là mở rộng thông điệp (M) đến độ dài là bội số của một số bit cụ thể. Vì các chức năng nén không thể nhận các đầu vào có kích thước tùy ý.

Để mở rộng thông điệp ban đầu, ta sử dụng 2 khối. Đầu tiên là khối đệm, khối thứ 2 biểu thị kích thước của thông điệp.

Ta tạo khối đệm dưới dạng ‘1’ theo sau là ‘0’, như hình trên ở phần mở rộng (expansion). Lý do để khối đệm bắt đầu bằng 1 là để tránh va chạm. Giả sử ta có một thông điệp M và ta thêm một khối đệm 0. Hãy tạo một thông điệp M’ là M thêm 0 ở cuối. Sau đó, hàm băm của M sẽ bằng hàm băm của M’. Ta sẽ không thể biết liệu số 0 sau M là khối đệm được thêm vào M hay đó là phần cuối của M’.

HÀM BĂM (HASH FUNCTION)

Cấu trúc Merkle-Damgard (tiếp):



Kích thước của khối đệm cộng với kích thước của thông điệp phải là bội số của một số cố định (tức là 512) trừ đi 64. Điều này là do ta phải sử dụng 64 bit cuối cùng (khối K) để chỉ định kích thước của khối đệm (tin nhắn gốc). Ta sử dụng kích thước của tin nhắn tạo hàm băm để trách cuộc tấn công mở rộng độ dài.
Ta xử lý thông điệp mở rộng trong các khối bằng cách nối chúng bằng vector ban đầu (IV) và hàm chống va chạm một chiều f. Giá trị vector ban đầu sẽ thay đổi với mỗi lần nối và khi kết thúc. Từ đó ta thu được giá trị băm.

Chức năng chống va chạm một chiều sẽ dựa trên các phép toán logic đã học (AND, OR, XOR, ..v..v.)
Kích thước của vector đầu vào (IV) sẽ xác định kích thước của giá trị băm.

HÀM BĂM (HASH FUNCTION)

Danh sách các thuật toán băm an toàn phổ biến nhất (Secure Hash Algorithms, SHA):

- **SHA-0**: **Hàm 160** bit này được NIST đề xuất vào năm 1993;
- **SHA-1**: Tính năng này được NIST đề xuất vào năm 1995 để thay thế cho SHA-0. SHA-1 cũng là một hàm băm 160 bit và được sử dụng rộng rãi trong SSL và TLS. Cần lưu ý rằng tính năng này hiện được coi là **không an toàn** và không được các cơ quan cấp chứng chỉ chấp thuận. Những phát triển mới không còn chấp nhận SHA-1;
- **SHA-2**: Nhóm hàm này bao gồm bốn hàm (**SHA-224, SHA-256, SHA-384 và SHA-512**) khác nhau về kích thước băm tính bằng bit;
- **SHA-3**: Thế hệ chức năng SHA mới nhất, bao gồm các chức năng **SHA-3-224, SHA-3-256, SHA-3-384 và SHA-3-512**. SHA-3 là một phiên bản của Keccak đã được NIST phê duyệt làm tiêu chuẩn và phát hành vào 5/8/2015. Thuật toán SHA-3 dựa trên cái gọi là hàm bọt biển(hoặc cấu trúc bọt biển), trái ngược với cấu trúc Merkle – Damgard được sử dụng rộng rãi trong SHA-1 và SHA-2;
- **RIPEMD** là viết tắt của **RACE Integrity Primaries Assessment Message Digest (Chức năng Tóm tắt Đánh giá Toàn vẹn Nguyên thủy RACE).** Thuật toán dựa trên các nguyên tắc **tương tự như MD4**. Có nhiều phiên bản RIPEMD khác nhau, bao gồm **128-bit, 160-bit, 256-bit và 320-bit**;
- **Whirpool** - Thuật toán này dựa trên phiên bản sửa đổi của mật mã **Randal (Rijndael),** được gọi là W. Nó sử dụng hàm nén Miaguchi – Prenel, một hàm một chiều nén hai thông điệp đầu vào có độ dài cố định thành một đầu ra có độ dài cố định thông điệp.

HÀM BĂM (HASH FUNCTION)

Khái niệm thuật toán:

Họ thuật toán SHA (Secure hash standard) bao gồm 5 thuật toán tính toán hàm băm: SHA-1, SHA-224, SHA-256, SHA-384, SHA-512.
Bốn hàm băm cuối cùng được kết hợp thành phân họ SHA-2. Thuật toán SHA-1 được Cơ quan An ninh Quốc gia Hoa Kỳ (NSA) phát triển vào năm 1995. Phân họ thuật toán SHA-2 cũng được Cơ quan An ninh Quốc gia Hoa Kỳ phát triển và được xuất bản bởi Viện Tiêu chuẩn và Công nghệ Quốc gia trong Tiêu chuẩn Xử lý Thông tin Liên bang FIPS PUB 180-2 vào tháng 8 năm 2002.

So với SHA-1, SHA-2 an toàn hơn nhiều (có độ dài bit lớn hơn) và được yêu cầu trong tất cả các chữ ký số, chứng chỉ kể từ năm 2016. Thông thường các cuộc tấn công như brute-force có thể mất nhiều năm hoặc thậm chí nhiều thập kỷ để bẻ khóa mã băm, vì vậy SHA-2 được tạm coi là thuật toán băm an toàn nhất.

Các thuật toán này được sử dụng trong SSL, SSH, S/MIME, DNSSEC, X.509, PGP, IPSec, khi truyền tệp qua mạng (BitTorrent).

HÀM BĂM (HASH FUNCTION)

Mô tả thuật toán:

Các thuật toán khác nhau về khả năng kháng phá mã, được cung cấp cho dữ liệu được băm, cũng như kích thước của các khối dữ liệu và các từ được sử dụng trong quá trình băm. Sự khác biệt chính của các thuật toán có thể được trình bày dưới dạng Bảng:

Thuật toán	Độ dài kết quả của dữ liệu (bits)	Độ dài trạng thái bên trong (bits)	Độ dài khối (bits)	Độ dài dữ liệu (bits)	Độ dài từ (bits)	Số lần lặp trong chu kỳ (bits)
SHA–1	160	160	512	$< 2^{64}$	32	80
SHA–224	224	256	512	$< 2^{64}$	32	64
SHA–256	256	256	512	$< 2^{64}$	32	64
SHA–384	384	512	1024	$< 2^{128}$	64	80
SHA–512	512	512	1024	$< 2^{128}$	64	80

HÀM BĂM (HASH FUNCTION)

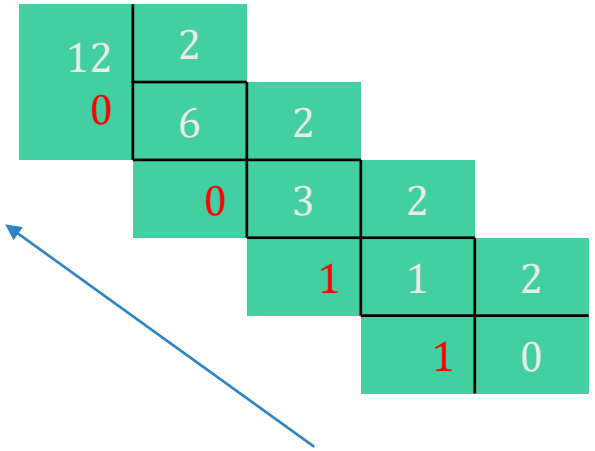
Từ năm 2011 đến 2015, SHA-1 đã từng là thuật toán chủ yếu. Tuy nhiên, một số lượng ngày càng lớn các nghiên cứu khóa học chỉ ra điểm yếu của SHA-1 đã thúc đẩy quy trình kiểm tra lại tính bảo mật của thuật toán này.

Google thậm chí còn tạo ra một "va chạm mã hóa" SHA-1 (khi hai tệp dữ liệu riêng biệt tạo ra cùng một mã băm, chúng được gọi là đã "va chạm"). Vì vậy, từ năm 2016 trở đi, SHA-2 trở thành tiêu chuẩn mới. Ngày nay nếu bạn nhận được một chứng thư số SSL/TLS , chứng thư ấy phải dùng SHA-2 ở mức tối thiểu.

SECURE HASH ALGORITHM 256 – SHA256

bit	{0,1}
Byte	8 bit
Word	Nhóm 32 bit (8 chữ số hex)
Hex	Số trong hệ 16
Hex digit	Được biểu diễn 4 bit (1 số trong hệ Hex)

Dec → Bin



Bin → Dec

$1100_2 \rightarrow 12_{10}$

2^3	2^2	2^1	0	$= 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 0 \cdot 2^0 = 12$
1	1	0	0	

SECURE HASH ALGORITHM 256 – SHA256

Bin → Hex

$11100_2 \rightarrow 1C_{16}$

2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
0	0	0	1	1	1	0	0
$0\cdot2^3 + 0\cdot2^2 + 0\cdot2^1 + 1\cdot2^0 = 1$				$1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 0\cdot2^0 = 12 = C_{16}$			
1C							

Hex → Bin

$1C_{16} \rightarrow 11100_2$

1 = 1				C = 8 + 4			
2^3	2^2	2^1	2^0	2^3	2^2	2^1	2^0
8	4	2	1	8	4	2	1
0	0	0	1	1	1	0	0

SECURE HASH ALGORITHM 256 – SHA256

SHA 256

1. Thuật toán sử dụng các phép toán tử bitwise $'\wedge', '\vee', '\oplus', '\neg', '\ll', '\gg'$ (and, or, xor, not, shift left, shift right)

2. Phép toán dịch chuyển sang phải ($SHR^n(x) = x \gg n$)

- $SHR^1(1010) = 1010 \gg 1 = 0101_{(1)}$
- $SHR^3(1010) = 1010 \gg 3 = 0101_{(1)} = 0010_{(2)} = 0001_{(3)}$

3. Dịch tuần hoàn sang phải ($ROTR^n(x)$)

- $ROTR^1(1010) = 0101$

- $ROTR^3(1010) = 0101_{(1)} = 1010_{(2)} = 0101_{(3)}$

4. Cộng Modulo 2^{32} ($x = (a + b)\ mod\ 2^{32}$)

Ví dụ:

a = $100_{(10)} = 1100100_{(2)}$

b = $200_{(10)} = 11001000_{(2)}$

Dec:

$x = (100 + 200)\ mod\ 2^7 = 300\ mod\ 2^7 = 44_{10} = 101100_2$

Bin:

$x = (1100100 + 11001000)\ mod\ 10000000 = 100101100\ mod\ 10000000 = 101100$

SECURE HASH ALGORITHM 256 – SHA256

Function

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

x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
$ROTR^7$	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	
$ROTR^{18}$	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	XOR
SHR^3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	XOR
$\sigma_0(x)$	1	1	1	1	0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1	1	1	1	0	0	0	0	0	0	0	

Dịch tuần hoàn sang phải 7 đơn vị.

Dịch sang phải 3 đơn vị.

SECURE HASH ALGORITHM 256 – SHA256

Function

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

x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$ROTR^7$	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$ROTR^{19}$	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	XOR
SHR^{10}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	XOR

$\sigma_1(x)$	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	1	1	1	1

SECURE HASH ALGORITHM 256 – SHA256

Function

$$\sum_{0}^{\{256\}}(x) = \boldsymbol{ROTR^{2}}(x) \oplus \boldsymbol{ROTR^{13}}(x) \oplus \boldsymbol{ROTR^{22}}(x)$$

x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
$ROTR^{2}$	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	
$ROTR^{13}$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	XOR
$ROTR^{22}$	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	XOR
$\sum_{0}^{\{256\}}(x)$	0	0	1	1	1	1	1	1	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	0	

SECURE HASH ALGORITHM 256 – SHA256

Function

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

x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	
$ROTR^6$	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	
$ROTR^{11}$	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	XOR
$ROTR^{25}$	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	XOR

$\sum_1^{\{256\}}(x)$	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1	0	0	0

SECURE HASH ALGORITHM 256 – SHA256

Function

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

x	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
z	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Ch	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

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

x	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
y	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
z	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Maj	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

SECURE HASH ALGORITHM 256 – SHA256

- **Constants (Hằng số)**

64 từ (32 bit mỗi từ) từ các phần thập phân căn bậc ba của 64 số nguyên tố đầu tiên được biểu diễn như sau:

$K_0^{\{256\}}, K_1^{\{256\}}, \ldots, K_{63}^{\{256\}}$

$K_0 = tp(\sqrt[3]{2}) = 0.259921049894873 =$ 01000010 10001010 00101111 10011000

$K_1 = tp(\sqrt[3]{3}) =$

$K_2 = tp(\sqrt[3]{5}) =$

…

$K_{63} = tp(\sqrt[3]{311}) = 0.775168952273312$

Chuyển phần thập phân → Bin

$tp(\sqrt[3]{2}) = 0.259921049894873$

0.2599	×2			
0	0.5198	×2		
	1	0.0396	×2	
		0	0.0792	×2
			0	0.1584

```
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
```

SECURE HASH ALGORITHM 256 – SHA256

- **Tiền xử lý:**

1. Chuyển đổi tin nhắn sang hệ nhị phân (sử dụng ASCII)

Ví dụ:
input: abc
byte [97,98,99]
Bin: 01100001’01100010’01100011

2. Thêm bộ đệm tin nhắn (padding the message)

Thêm bộ đệm tin nhắn cho đến khi số bit của nó đạt đến bội số gần nhất của 512
Lưu ý: SHA-256 làm việc với 512 bit

2.1. Thêm một bit “1” vào cuối:

+ Đóng vài trò là dấu phân cách giữa tin nhắn gốc và bit “0” sẽ được đệm vào tin nhắn.
message:
Input: abc
bytes: [97,98,99]
message: 01100001’01100010’01100011

Thêm đệm (padding): 1 bit (“1”)
Message: 01100001’01100010’011000111

2.2 Đệm với bit “0”

Đệm tin nhắn với bit “0” cho đến khi nó chỉ còn 64 bit. Với k là số bit “0” được đệm. Giả sử độ dài tin nhắn (M) là l bit. Nối bit “1” vào cuối tin nhắn theo sau là k bit “0”, trong đó k là nghiệm không âm nhỏ nhất của phương trình:

$$l + 1 + k \equiv 448 \; mod \; 512 \; (1)$$

Lưu ý: 64 bits cuối được dành riêng để chỉ ra độ dài của tin nhắn thực tế ở dạng nhị phân

Ở đây ta có: Tin nhắn: abc, l = 24 bit. Thay vào (1):

$$24+1 + k \equiv 448 \; mod \; 512 \implies k = 423$$

Khi đó tin nhắn abc (độ dài l=24) có dạng: $\underbrace{01100001}_{a}\ \underbrace{01100010}_{b}\ \underbrace{01100011}_{c}\ 1\ \overbrace{00\ldots00}^{423}\ \overbrace{00\ldots0\underbrace{11000}_{l=24}}^{64\ bit}$

SECURE HASH ALGORITHM 256 – SHA256

- **Tiền xử lý:**

3. Phân tích cú pháp tin nhắn đệm:

+ Tin nhắn đã thêm đệm được chia thành các khối tin nhắn trong 512 bit được ký hiệu:

$$M^{(1)}, M^{(2)}, \dots, M^{(N)}$$

+ Tin nhắn 512 được chia thành 16 khối (mỗi khối 32 bit):

$M_0^{(i)}, M_1^{(i)}, \dots M_{15}^{(i)}$, gọi là lịch trình tin nhắn

+ Mở rộng "lịch trình tin nhắn" sao cho có tổng cộng 64 từ (words).

3.1. Lịch trình tin nhắn

$M_0 = W_0$	0	1	1	0	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	0	0	0	1	1	1	0	0	0	0	0	0	0	32 bit
$M_1 = W_1$	0																														0	32 bit
...																																
$M_{15} = W_{15}$	0	0																										1	1	0	0	0	

SECURE HASH ALGORITHM 256 – SHA256

- **Tiền xử lý:**

3. Phân tích cú pháp tin nhắn đệm:

3.2. Mở rộng lịch trình tin nhắn:

+ Tạo 48 từ (word) (mỗi từ 32 bit) sử dụng công thức bên dưới:

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

Ở đây:

– $M_t^{(i)}$: 1 từ (32 bit) lấy từ 512 bit (khối tin nhắn)

- W_t: 1 từ (32 bit) được tạo từ công thức trên

Lưu ý:

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

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

$W_0 = M_0$

$W_1 = M_1$

……

$W_{15} = M_{15}$

$W_{16} = \sigma_1^{\{256\}}(W_{14}) + W_9 + \sigma_0^{\{256\}}(W_1) + W_0$

……..

$W_{63} = \sigma_1^{\{256\}}(W_{61}) + W_{56} + \sigma_0^{\{256\}}(W_{48}) + W_{47}$

SECURE HASH ALGORITHM 256 – SHA256

- **Tiền xử lý:**

4. Thiết lập giá trị băm ban đầu:

+ Giá trị băm ban đầu phải được đặt trước khi tính toán băm được bắt đầu

+ Nó bao gồm 8 từ (word) (mỗi từ 32 bit) bắt nguồn từ 32 bit đầu tiên của các phần "thập phân" của căn bậc 2 (√) của 8 số nguyên tố đầu tiên

Chuyển qua hệ Hex
$H_0^{(0)} = ps(\sqrt{2}) = 6a09e667$
$H_1^{(0)} = ps(\sqrt{3}) = bb67ae85$
$H_2^{(0)} = ps(\sqrt{5}) = 3c6ef372$
$H_3^{(0)} = ps(\sqrt{7}) = a54ff53a$
$H_4^{(0)} = ps(\sqrt{11}) = 510e527f$
$H_5^{(0)} = ps(\sqrt{13}) = 9b05688c$
$H_6^{(0)} = ps(\sqrt{17}) = 1f83d9ab$
$H_7^{(0)} = ps(\sqrt{19}) = 5be0cd19$

Chuyển phần thập phân → Bin
$\sqrt{2} = 1.414213562373095$

0.4142	×2			
0	0.8284	×2		
	1	0.6568	×2	
		1	0.3136	×2
			0	0.6272

Lấy 32 bit đầu tiên

$$H_0^{(0)} = \underbrace{0110\ldots}_{6_{(16)}}$$

SECURE HASH ALGORITHM 256 – SHA256

- **Tính toán băm:**

1. Mở rộng lịch trình tin nhắn sao cho tổng cộng 64 từ (word)
2. Khởi tạo 8 biến làm việc: Vị trí nơi các từ sẽ được lưu trữ trong quá trình nén

Biến làm việc
$a = H_0^{(i-1)} = tp(\sqrt{2}) = 0.4142135624 \times 2^{32} = 1779033703_{(10)} = 01101010'00001001'11100110'01100111_{(2)}$
b= $H_1^{(i-1)} = tp(\sqrt{3}) = 0.7320508075 \times 2^{32} = 3144134277_{(10)} = 10111011\ 01100111\ 10101110\ 10000101_{(2)}$
c= $H_2^{(i-1)} = tp(\sqrt{5}) = 0.236 \ldots \ldots \times 2^{32} = \ldots.$
d= $H_3^{(i-1)} = tp(\sqrt{7}) = 0.6457 \ldots \times 2^{32} = \ldots.$
e= $H_4^{(i-1)} = tp(\sqrt{11}) = 0.3166 \ldots \times 2^{32} = \ldots.$
f= $H_5^{(i-1)} = tp(\sqrt{13}) = 0.6055 \ldots \times 2^{32} = \ldots.$
g= $H_6^{(i-1)} = tp(\sqrt{17}) = 0.1231 \ldots \times 2^{32} = \ldots.$
h= $H_7^{(i-1)} = tp(\sqrt{19}) = 0.3588 \ldots \times 2^{32} = \ldots..$

SECURE HASH ALGORITHM 256 – SHA256

- **Tính toán băm:**

3. Chạy chu kỳ nén:

Sửa đổi các giá trị của các biến làm việc bằng cách

for t =1 to 63

{

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

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

h=g

g=f

f=e

e=d+ T_1

d=c

c=b

b=a

a= T_1+ T_2

}

+ Ở đây:

T_1: Từ tạm 1

T_1: Từ tạm 2

a, b, c, d, e, f, g, h: Biến làm việc

$K_t^{\{256\}}$: Hằng số (constant)

W_t: Từ trong lịch trình tin nhắn

$Maj(a,b,c)$: Hàm Majority tham số (a,b,c)

$Ch(e,f,g)$: Hàm chọn tham số (e,f,g)

SECURE HASH ALGORITHM 256 – SHA256

- **Tính toán băm:**

4. Tính giá trị hàm trung gian:

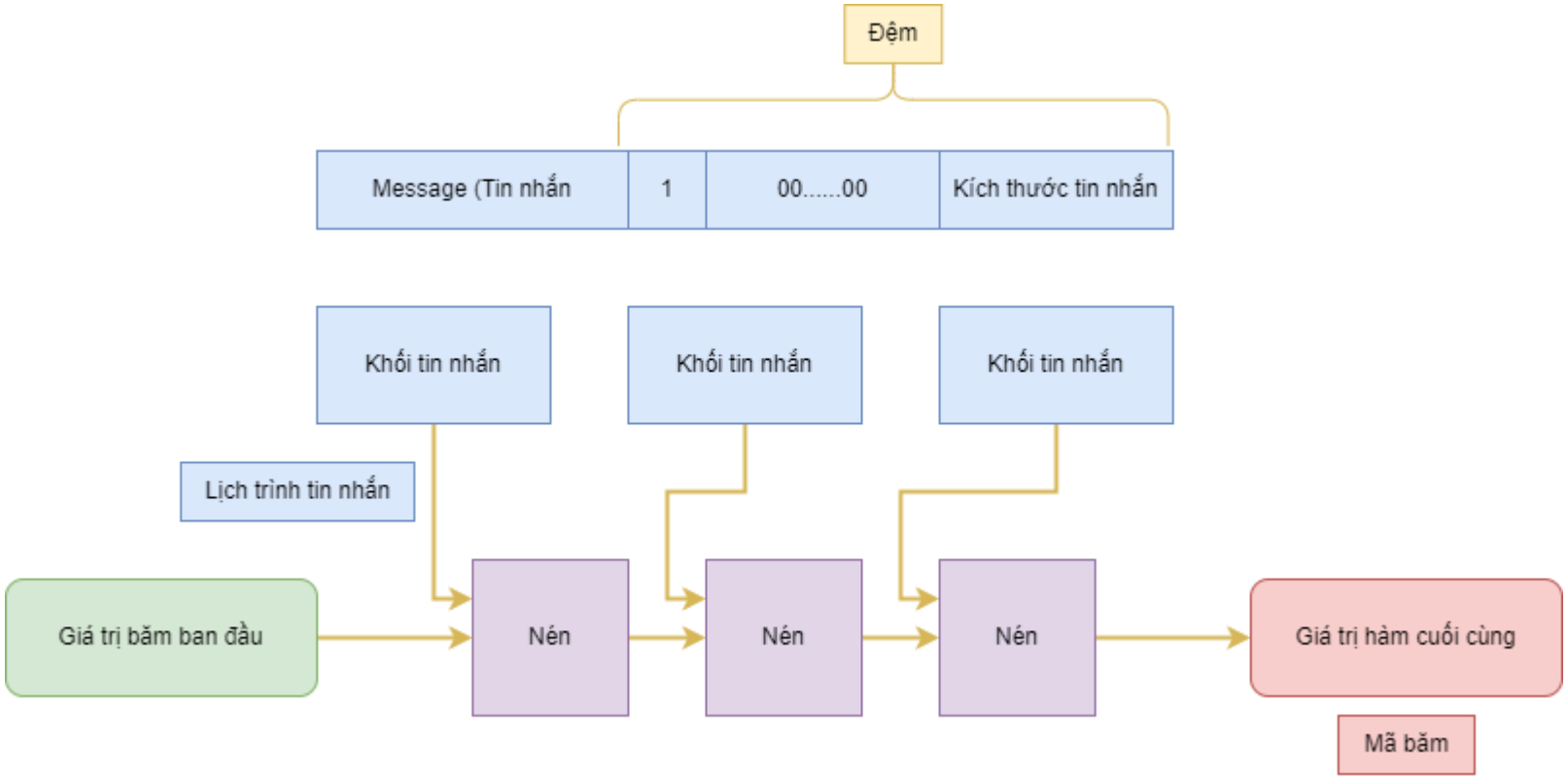
+ Lấy giá trị băm ban đầu và thêm biến làm việc được cập nhập tương ứng

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

SECURE HASH ALGORITHM 256 – SHA256

- **Tính toán băm:**

5. Lặp lại từ (1) đến (4) cho mỗi khối tin nhắn 512 bit:



SECURE HASH ALGORITHM 256 – SHA256

- **Tính toán băm:**

6. Chuyển đổi giá trị băm.

Chuyển đổi giá trị băm hệ nhị phân thành hệ Hex(16) sau đó nối các giá trị lại với nhau

a = 10111010’01111000’00010110’10111111=ba7816bf
b=
…
h=

abcdefgh= ba7816bf……

Kết quả: “abc” = ba7816bf……