

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO BÀI TẬP LỚN

Môn: Cơ sở an toàn thông tin

Tìm hiểu các giải thuật băm SHA-0,1,2,3

Các điểm yếu, các dạng tấn công vào SHA

Cài đặt thử nghiệm SHA1

Giảng viên: TS.Hoàng Xuân Dậu

Nhóm sinh viên: G06

1. Đinh Quang Hiếu-B19DCAT065
2. Phạm Ngọc Hiếu-B19DCAT071
3. Trần Đình Hiếu-B19DCAT072
4. Trần Trung Hiếu-B19DCAT073
5. Triệu Xuân Hùng-B19DCAT083
6. Lê Quốc Hùng -B19DCAT081

Hà Nội – 2021

Mục lục

Chương 1. Giới thiệu và kiến trúc của hàm băm	2
1. Giới thiệu về hàm băm.....	2
2. Định nghĩa tổng quát của hàm băm	2
3. Tính chất cơ bản của hàm băm.....	3
4. Trong lý thuyết mật mã, mức độ an toàn của hàm băm mật mã đã được xác định bằng các thuộc tính sau:	3
5. Phân loại hàm băm theo khóa sử dụng	3
6. Mô hình xử lý dữ liệu.....	4
7. Mô hình chi tiết xử lý dữ liệu của hàm băm.....	5
Chương 2. Các giải thuật hàm băm SHA.....	5
1. Giới thiệu về SHA.....	5
2. Các giải thuật SHA.....	6
2.1 SHA-0.....	6
2.2 SHA-1	7
2.3. SHA-2.....	8
2.4. SHA-3.....	11
Chương 3. Các điểm yếu và các dạng tấn công vào hàm băm	15
1. Các điểm yếu của SHA	15
2. Các cuộc tấn công:	15
Chương 4. Các ứng dụng của hàm băm	19
1. Che dấu dữ liệu gốc.....	19
2. Hashing trong định danh dữ liệu.....	19
3. Ứng dụng trong lập chỉ mục và truy xuất dữ liệu.....	20
4. Ứng dụng trong bằng chứng công việc (Proof of Work)	20
Chương 5. Cài đặt thử nghiệm	21
Kết luận :	22
Tài liệu tham khảo	22

Chương 1. Giới thiệu và kiến trúc của hàm băm

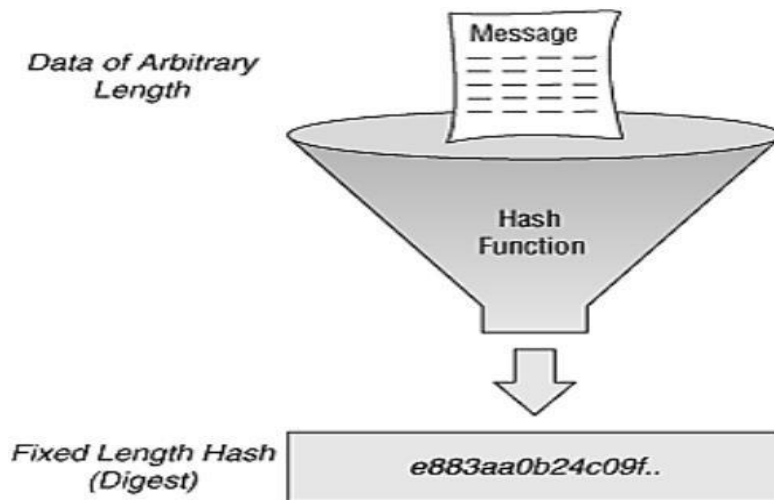
1. Giới thiệu về hàm băm

- Nếu bạn mua một chiếc điện thoại mới và lớp bọc co lại của nó bị rách hoặc hư hỏng, bạn có thể biết ngay rằng ai đó đã mở, sử dụng, thay thế hoặc làm hỏng điện thoại. Hàm băm mật mã trong mã hóa cũng giống như vậy nhưng đối với dữ liệu thay vì đối tượng vật lý. Theo cách tương tự, băm giống như đặt màng bọc thu nhỏ ảo lên một phần mềm, ứng dụng hoặc dữ liệu để thông báo cho người dùng nếu nó đã được sửa đổi theo bất kỳ cách nào.
- Nhưng băm là gì? Hashing, hoặc thuật toán băm, là một quá trình một chiều chuyển đổi dữ liệu đầu vào của bạn ở bất kỳ kích thước nào thành dữ liệu được nén có độ dài cố định. Trung tâm của quá trình là nơi bạn sẽ tìm thấy hàm băm. Về cơ bản, bạn có thể lấy một câu ngắn hoặc toàn bộ luồng dữ liệu, chạy nó thông qua một hàm băm và kết thúc với một chuỗi dữ liệu có độ dài cụ thể. Đó là một cách để ẩn dữ liệu ban đầu của bạn để làm cho việc thiết kế ngược trở nên khó khăn nhất có thể.
- Theo nghĩa kỹ thuật hơn, đó là một kỹ thuật sử dụng một phép toán học để thu nhỏ một lượng ngẫu nhiên dữ liệu đầu vào (được gọi là khóa băm) thành một chuỗi bit có độ dài cố định theo cách quá phi thực tế để đảo ngược với các máy tính hiện đại. Vì vậy, định nghĩa của hàm băm sẽ là thứ lấy dữ liệu đầu vào và sử dụng nó để tạo ra giá trị đầu ra có độ dài cố định, duy nhất và hầu như không thể thay đổi (cho tất cả các ý định và mục đích thực tế).

2. Định nghĩa tổng quát của hàm băm

Hàm băm (hash function) là một hàm toán học h có tối thiểu 2 thuộc tính:

- Nén (Compression): h là một ánh xạ từ chuỗi đầu vào x có chiều dài bất kỳ sang một chuỗi đầu ra $h(x)$ có chiều dài cố định n bit.
- Dễ tính toán (Ease of computation): cho trước hàm h và đầu vào x , việc tính toán $h(x)$ là dễ dàng.[1]



3. Tính chất cơ bản của hàm băm

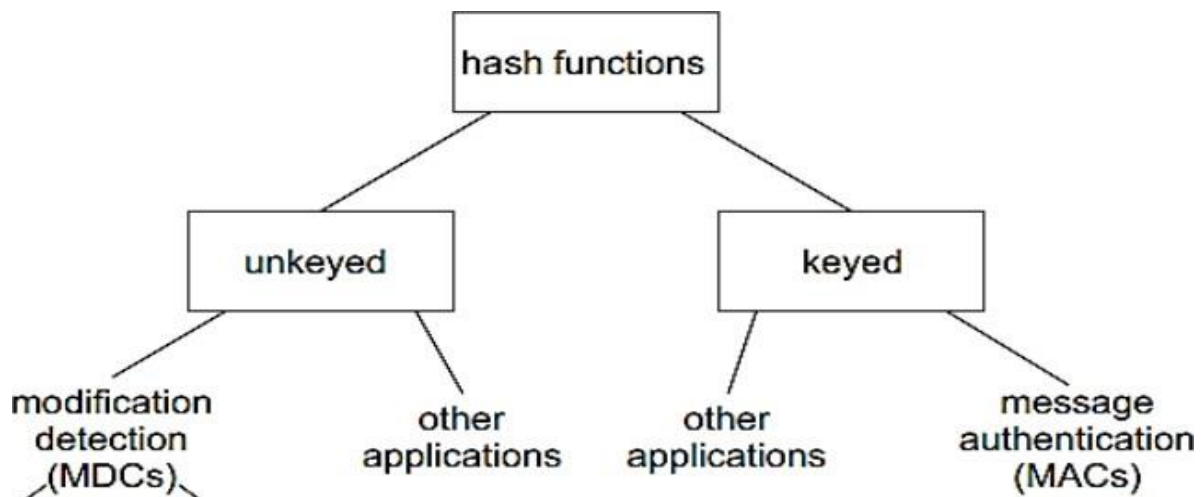
- **Tính một chiều:** không thể suy ra dữ liệu ban đầu từ kết quả, điều này tương tự như việc bạn không thể chỉ dựa vào một dấu vân tay lạ mà suy ra ai là chủ của nó được.
- **Tính duy nhất:** xác suất để có một vụ va chạm (hash collision), tức là hai thông điệp khác nhau có cùng một kết quả hash là cực kỳ nhỏ

4. Trong lý thuyết mật mã, mức độ an toàn của hàm băm mật mã đã được xác định bằng các thuộc tính sau:

- Tính kháng tiền ảnh thứ nhất. Tính chất yêu cầu rằng với một giá trị băm h bất kỳ, sẽ khó tìm thấy bất kỳ thông điệp m nào sao cho $h = \text{hash}(m)$. Khái niệm này có liên quan đến tính chất một chiều của hàm băm.
- Tính kháng tiền ảnh thứ hai. Với đầu vào m_1 , sẽ khó tìm được đầu vào m_2 khác sao cho $\text{hash}(m_1) = \text{hash}(m_2)$.
- Tính kháng va chạm. Rất khó để tìm thấy hai thông điệp khác nhau m_1 và m_2 sao cho $\text{hash}(m_1) = \text{hash}(m_2)$. Một giá trị như vậy được gọi là va chạm của hàm băm mật mã.

5. Phân loại hàm băm theo khóa sử dụng

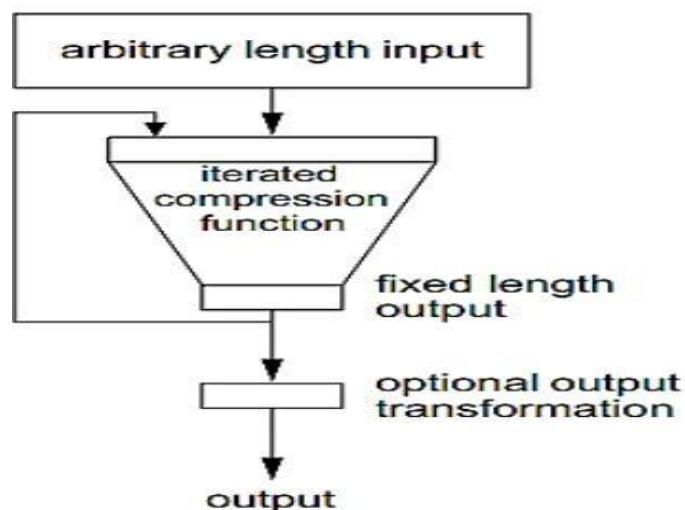
- Hàm băm không khóa (unkeyed): đầu vào chỉ là thông điệp
- Hàm băm có khóa (keyed): đầu vào gồm thông điệp và khóa



Theo chức năng, có thể chia các hàm băm thành 2 loại chính:

- Mã phát hiện sửa đổi (MDC - Modification Detection Code): MDC thường được sử dụng để tạo chuỗi đại diện cho thông điệp và dùng kết hợp với các kỹ thuật khác (như chữ ký số) để đảm bảo tính toàn vẹn của thông điệp. MDC thuộc loại hàm băm không khóa. MDC gồm 2 loại nhỏ:
 - Hàm băm một chiều (OWHF - One-way hash functions): Với hàm băm một chiều, việc tính giá trị băm là dễ dàng, nhưng việc khôi phục thông điệp từ giá trị băm là rất khó khăn;
 - Hàm băm chống đụng độ (CRHF - Collision resistant hash functions): Với hàm băm chống đụng độ, sẽ là rất khó để tìm được 2 thông điệp khác nhau nhưng có cùng giá trị băm.
- Mã xác thực thông điệp (MAC - Message Authentication Code): MAC cũng được dùng để đảm bảo tính toàn vẹn của thông điệp mà không cần một kỹ thuật bổ sung nào khác. MAC là loại hàm băm có khóa như đã đề cập ở trên, với đầu vào là thông điệp và một khóa bí mật.

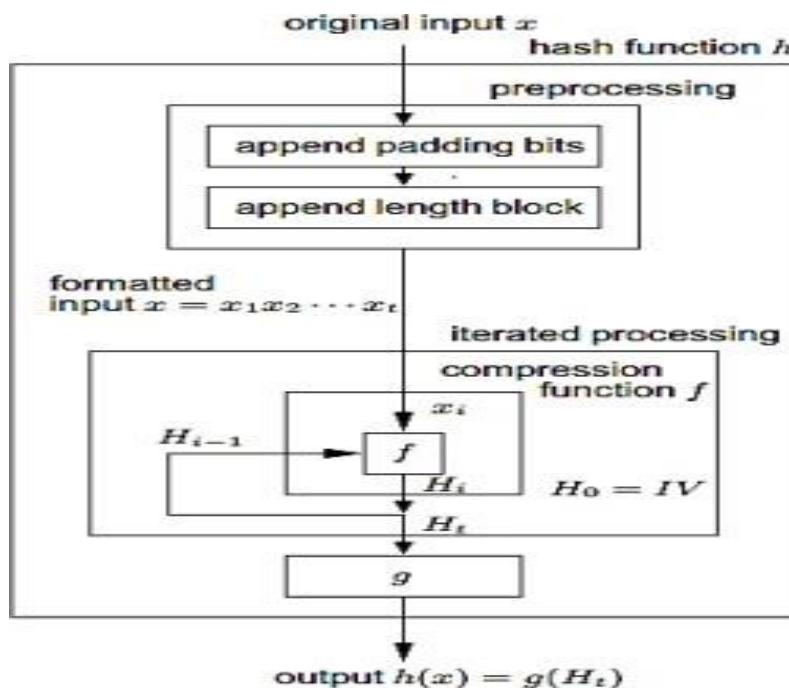
6. Mô hình xử lý dữ liệu



Thông điệp đầu vào với độ dài tùy ý (arbitrary length input) đi qua hàm nén lặp nhiều vòng 11 (iterated compression function) để tạo chuỗi đầu ra có kích thước cố định (fixed length output). Chuỗi này đi qua một khâu chuyển đổi định dạng tùy chọn (optional output transformation) để tạo ra chuỗi băm kết quả (output)

7. Mô hình chi tiết xử lý dữ liệu của hàm băm

Quá trình xử lý gồm 3 bước chính: (1) tiền xử lý (preprocessing), (2) xử lý lặp (iterated processing) và (3) chuyển đổi định dạng. Trong bước tiền xử lý, thông điệp đầu vào x trước hết được nối đuôi thêm một số bit và kích thước khối, sau đó chia thành các khối có kích thước xác định. Kết quả của bước này là t khối dữ liệu có cùng kích thước có dạng $x = x_1 x_2 \dots x_t$ làm đầu vào cho bước 2. Trong bước 2, từng khối dữ liệu x_i được xử lý thông qua hàm nén f để tạo đầu ra là H_i . Kết quả của bước 2 là chuỗi đầu ra H_t và H_t được chuyển đổi định dạng bởi hàm g để tạo chuỗi giá trị băm kết quả $h(x)$.



Chương 2. Các giải thuật hàm băm SHA

1. Giới thiệu về SHA

SHA là các thuật giải được chấp nhận bởi FIPS dùng để chuyển một đoạn dữ liệu nhất định thành một đoạn dữ liệu có chiều dài không đổi với xác suất khác biệt cao. Những thuật giải này được gọi là "an toàn" bởi vì, theo nguyên văn của chuẩn FIPS 180-2 phát hành ngày 1 tháng 8 năm 2002:

1) Cho một giá trị băm nhất định được tạo nên bởi một trong những thuật giải SHA, việc tìm lại được đoạn dữ liệu gốc là không khả thi.

2) Việc tìm được hai đoạn dữ liệu khác nhau có cùng kết quả băm tạo ra bởi một trong những thuật giải SHA là không khả thi.

2. Các giải thuật SHA

2.1 SHA-0

2.1.1 Giới thiệu về SHA-0

SHA-0 là phiên bản đầu tiên gồm các đặc tả ban đầu của thuật toán hàm băm an toàn đã được xuất bản vào năm 1993 dưới tiêu đề Secure Hash Standard , FIPS PUB 180, bởi cơ quan tiêu chuẩn Hoa Kỳ của cơ quan NIST(Viện Tiêu chuẩn và Công nghệ Quốc gia). Nó đã bị NSA thu hồi ngay sau khi xuất bản và bị thay thế bởi bản sửa đổi, được xuất bản vào năm 1995 trong FIPS PUB 180-1 và được gọi là SHA-1..

SHA-0 là một hàm băm dành riêng 160-bit dựa trên nguyên lý thiết kế của MD4. Nó áp dụng mô hình Merkle-Damgard cho một chức năng nén chuyên dụng. Đầu vào tin nhắn được đệm và chia thành k khối tin 512-bit. Tại mỗi lần lặp lại của hàm nén h, một biến 15 chuỗi 160 bit H_t được cập nhật bằng một khối tin M_{t+1} , tức là $H_{t+1} = h(H_t, M_{t+1})$. Giá trị ban đầu H_0 (còn gọi là IV) được xác định trước và H_k là đầu ra của hàm băm.

2.1.2 Giải thuật SHA-0

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 M_j thành 16 từ 32-bit W_0, W_1, \dots, W_{15} .

2 Mở rộng từ bằng đệ quy:

$$W_i = W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad (16 \leq i < 80)$$

3 Chia h_{j-1} cho 5 thành ghi A, B, C, D và E :

$$h_{(j-1)} = (A_0, B_0, C_0, D_0, E_0)$$

4 Lặp 80 lần $t=0 \rightarrow t=79$:

Round	Step i	$f_i(x,y,z)$	K_i
1	$1 \leq i \leq 20$	$f_{IF} = (x \wedge y) \oplus (\wedge z)$	0x5a827999
2	$21 \leq i \leq 40$	$f_{XOR} = x \oplus y \oplus z$	0x6ed6eba1
3	$41 \leq i \leq 60$	$f_{MAJ} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcde
4	$61 \leq i \leq 80$	$f_{XOR} = x \oplus y \oplus z$	0xca62c1d6

5 Đầu ra hàm nén:

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

2.2 SHA-1

2.2.1 Giới thiệu SHA-1

Hàm băm SHA-1 đã được NIST đưa ra vào năm 1995 như là một Tiêu chuẩn xử lý Thông tin Liên bang. Từ khi xuất bản, SHA-1 đã được chấp nhận bởi nhiều chính phủ và các tiêu chuẩn ngành an ninh, đặc biệt là các tiêu chuẩn về chữ ký số mà cần có hàm băm chống xung đột. Ngoài việc sử dụng chữ ký số, SHA-1 cũng đã được triển khai như một thành phần quan trọng trong các chương trình và giao thức mật mã khác nhau, chẳng hạn như xác thực người dùng, hợp đồng khóa và tạo ra số giả ngẫu nhiên. Do đó, SHA-1 đã được triển khai rộng rãi trong hầu hết các hệ thống và sản phẩm bảo mật thương mại.

2.2.2 Giải thuật SHA-1

Hàm băm SHA-1 nhận thông báo có chiều dài nhỏ hơn 2^{64} bit và tạo ra giá trị băm 160 bit. Thông điệp đầu vào được đệm và sau đó được xử lý trong các khối 512-bit trong cấu trúc lặp Damgard / Merkle. Mỗi lần lặp lại gọi hàm nén có giá trị ràng buộc 160 bit và một khối tin 512 bit và xuất ra một giá trị chuỗi khác 160 bit. Ban đầu giá trị chuỗi (gọi là IV) là một tập các hằng cố định, và giá trị chuỗi cuối cùng là băm của thông báo. Trong phần sau, chúng ta mô tả hàm nén của SHA-1.

Đối với mỗi khối 512 bit của tin nhắn có đệm, chia nó thành 16 thông báo dài 32-bit, (m_0, m_1, \dots, m_{15}), 80 bits từ ($W_0 \rightarrow W_{79}$). Sau đó được mở rộng theo công thức đệ quy:

$$W_i = \text{ROL1}(W_{i-3} \oplus W_{i-8} \oplus W_{i-14} \oplus W_{i-16} \quad (16 \leq i < 80)$$

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

For $t=0$ to 79:

{

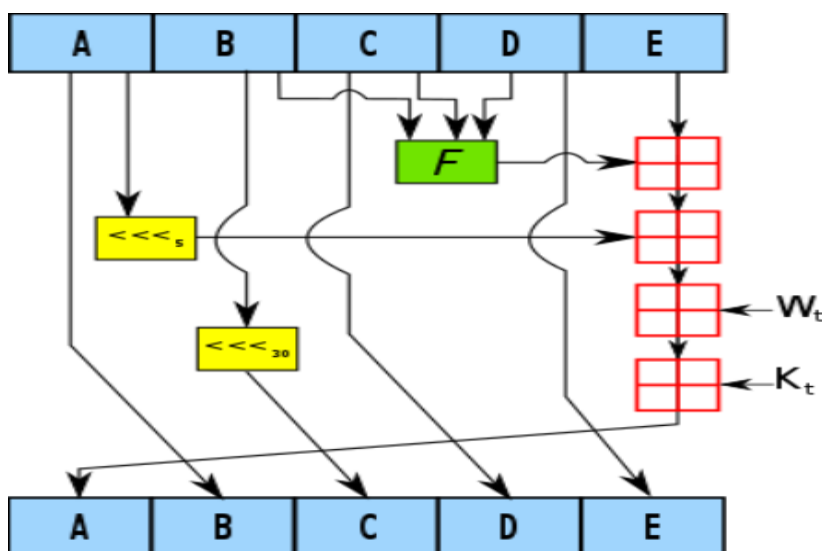
Round	Step i	$f_i(x,y,z)$	K_i
1	$0 \leq i \leq 19$	$f_{IF} = (x \wedge y) \oplus (\wedge z)$	0x5a827999
2	$20 \leq i \leq 39$	$f_{XOR} = x \oplus y \oplus z$	0x6ed6eba1
3	$40 \leq i \leq 59$	$f_{MAJ} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcdd
4	$60 \leq i \leq 79$	$f_{XOR} = x \oplus y \oplus z$	0xca62c1d6

}

Tính toán giá trị băm trung gian

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

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



2.3. SHA-2

2.3.1 Giới thiệu về SHA-2

SHA-2 (Secure Hash Algorithm 2) là một bộ các hàm băm mật mã được Thiết kế bởi Cơ quan An ninh Quốc gia Hoa Kỳ (NSA).

SHA-2 bao gồm những thay đổi đáng kể so với tiền nhiệm của nó, SHA-1 . Họ SHA-2 bao gồm sáu hàm băm với digests (giá trị băm) đó là 224, 256, 384 hoặc 512 bit: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA -512/256 .

2.3.2 Giải thuật SHA-2

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 SHA-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 độ 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 phân 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

Bước 6: 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 nội chuỗi đơn giản sẽ làm được.

SHA-224:

Khác ở SHA-256 ở giá trị khởi tạo

h0 = c1059ed8 h4 = ffc00b31

h1 = 367cd507 h5 = 68581511

h2 = 3070dd17 h6 = 64f98fa7

h3 = f70e5939 h7 = befa4fa4

Và giá trị cuối cùng rút gọn còn 224bits

Giải thuật SHA-512

SHA-512 có cấu trúc giống với **SHA-256** , nhưng:

Tin nhắn được chia thành các phần 1024 bit

Giá trị băm ban đầu và hằng số vòng được mở rộng thành 64 bit,

Có 80 vòng thay vì 64

Mảng lịch trình thông báo có các từ 80 - 64 bit thay vì các từ 64 - 32 bit,

Để mở rộng mảng lịch trình thông báo w, vòng lặp là từ 16 đến 79 thay vì từ 16 đến 63

Các hằng số vòng dựa trên 80 số nguyên tố đầu tiên 2..409

Kích thước từ được sử dụng để tính toán dài 64 bit

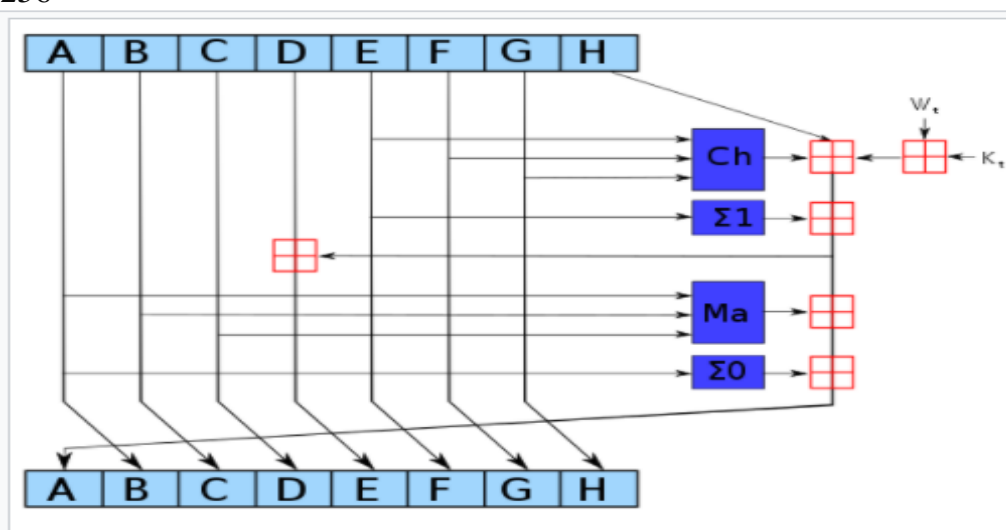
Độ dài thêm vào của thông báo (trước khi xử lý trước), tính bằng bit, là số nguyên big-endian 128 bit và số tiền thay đổi và luân chuyển được sử dụng là khác nhau.

Giải thuật SHA-384:

Khác SHA-512 ở giá trị khởi tạo và thông điệp cuối cùng chỉ có 384 bits

Giải thuật SHA-512/224,SHA-512/256:

Tương tự như SHA-512 nhưng khác giá trị khởi tạo và độ dài đầu ra lần lượt là 224 và 256



Một lần lặp lại trong hàm nén họ SHA-2. Các thành phần màu xanh lam thực hiện các hoạt động sau:


$$\text{Ch}(E, F, G) = (E \wedge F) \oplus (\neg E \wedge G)$$

$$\text{Ma}(A, B, C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$

$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$

$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

Xoay chiều bit sử dụng các hằng số khác nhau cho SHA-512. Các số đã cho là dành cho SHA-256.

Màu đỏ  là modulo bổ sung 2^{32} cho SHA-256 hoặc 2^{64} cho SHA-512.

2.4. SHA-3

2.4.1 Giới thiệu SHA-3

Hàm băm SHA-3 đã được chuẩn hoá bởi Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) vào tháng 8 năm 2015, như được quy định trong [FIPS PUB 202] (Tiêu chuẩn SHA-3). Họ SHA-3 bao gồm bốn hàm băm mật mã, được gọi là SHA3-224, SHA3-256, SHA3-384 và SHA3-512, và hai hàm mở rộng đầu ra (XOFs), được gọi là SHAKE128 và SHAKE256.

2.4.2 Giải thuật SHA-3

SHA3 gồm 4 hàm băm SHA3-224, SHA3-256, SHA3-384, SHA3-512

1. Đệm

Đây là quá trình tương tự như các thuật toán băm trước đây. Trước khi có thể bắt đầu băm thông điệp của mình, chúng tôi cần đảm bảo rằng chúng có độ dài tiêu chuẩn và để chúng tôi thực hiện quá trình đệm.

Trước khi tiếp tục, chúng ta cần biết kích thước tiêu chuẩn mà chúng ta cần đáp ứng là gì và để làm được điều đó, chúng ta sẽ xem xét cách Keccak tính toán kích thước trạng thái.

$b = 25 \times 2^l$; b là kích thước trạng thái.

$l = \{0, 1, 2, 3, 4, 5, 6\}$

$b = \{25, 50, 100, 200, 400, 800, 1600\}$

Đối với SHA-3, giá trị của $l=6$. Kích thước trạng thái càng cao thì càng tốt cho khả năng bảo mật mà nó cung cấp. Bây giờ, dựa trên giá trị của 'l', chúng tôi cũng quyết định số vòng tính toán cần được thực hiện cho mỗi phần của thông báo đệm.

$\text{rounds} = 12 + 2 \times l = 12 + 12$ với $l = 6$

Bây giờ, chúng ta biết rằng đối với SHA-3, chúng ta sẽ có kích thước trạng thái là 1600 bit và số vòng tính toán sẽ là 24.

Quay lại với padding, chúng ta cần nối các bit vào thông báo tùy thuộc vào độ dài băm mà chúng ta sẽ tính toán. Các giá trị phải là bội số mà tôi sẽ đề cập bên dưới. Bây giờ chỉ cần nhớ những giá trị này khi tôi giải thích về chúng sau này.

Type	Output Length	Rate (r)	Capacity (c)
SHA3-224	224	1152	448
SHA3-256	256	1088	512
SHA3-384	384	832	768
SHA3-512	512	576	1024

Phần đệm cần được thực hiện theo cách sao cho sau quá trình đệm, độ dài của thông báo đệm là bội số chính xác của 'r' cho hàm băm tương ứng.

Bit đầu tiên và bit cuối cùng của phần đệm sẽ là '1' và tất cả các bit ở giữa sẽ là '0'.

Sau khi đệm chúng được chia thành các phần 'n' chẳng hạn như n lần r tương đương với độ dài của thông điệp được đệm. Về mặt toán học, nó có thể được biểu diễn như vậy.

$p = n \times r$

p = độ dài của tin nhắn sau khi đệm

n = số phần mà chúng ta chia p

r = độ dài của tỷ lệ

Tổng các giá trị của 'r' và 'c' sẽ luôn bằng 1600, tức là kích thước state

2. Kích thước trạng thái

Bây giờ chúng ta biết rằng độ dài của thông điệp được đệm là bội số chính xác của 'r' cho độ dài băm tương ứng nhưng để hiểu thêm về nó. Chữ 'r' và 'c' trong hình ảnh đại diện cho tốc độ và dung lượng của thuật toán băm tương ứng.

Vì thông báo đệm là bội số chính xác của 'r' và chúng tôi cần thực hiện thao tác mô-đun. Vì vậy, độ dài của 'r' và P_0 là như nhau.

Kích thước trạng thái là tổng của 'r' và 'c' và chúng có các giá trị khác nhau đối với độ dài băm khác nhau.

Thuật toán SHA-3 có thể được chia thành hai phần khác nhau, phần hấp thụ và phần ép. Chức năng hấp thụ là phần đầu tiên của hai bước chính của chức năng SHA-3.

3. Chức năng hấp thụ

Lý do chúng tôi gọi nó là hàm hấp thụ là trong phần đầu tiên của thuật toán Keccak, chúng tôi lấy tất cả các giá trị của thông báo đệm mà chúng tôi đã chia nhỏ thành 'n' số phần và sử dụng từng phần một để đưa ra kết quả cuối cùng.

Cách chúng tôi thực hiện điều này là chúng tôi cung cấp các thông điệp được đệm có độ dài 'r' trong hàm hấp thụ. Chúng ta bắt đầu với phép toán modulo giữa P_0 và 'r', giá trị khởi tạo của 'r' là tất cả các bit '0'. Khi hoạt động mô-đun được thực hiện, chúng tôi chuyển giá trị cho chức năng mà chức năng hấp thụ thực sự bắt đầu.

Bên trong hàm, chúng ta thực hiện lặp đi lặp lại cùng một nhóm năm hoạt động trong 24 lần. Khi tất cả các vòng kết thúc, chúng tôi tách các bit 'r' và 'c' và sau đó thực hiện lại thao tác modulo và chức năng bắt đầu lại từ đầu.

Hãy xem mã giả của năm hàm:

θ (theta) : Pseudo-Code

	for	x	in		θ	...
	4					
	C	[x]	=	
	A	[x	,	θ]
xor						
	A	[x	,	1]
xor						
	A	[x	,	2]
xor						
	A	[x	,	3]
xor						
	A	[x	,	4]
	,					

```

for x in 0 ...
4
D [ x ] =
C [ x - 1 ]
xor
rot ( C [ x +
1 ] , 1 ) ,

```

```

for
( x , y )
in ( 0 ...
4 , 0 ... 4 )
A [ x , y ]
=
A [ x , y ]
xor D [ x ]

```

ρ (rho) & π (pi) : Pseudo-Code

```

for
( x , y )
in ( 0 ...
4 , 0 ... 4 )
B [ y , 2 *
x + 3 * y ]
=
rot ( A [ x ,
y ] ,
r [ x , y ] )
,

```

χ (chi) : Pseudo-Code

```

for
( x , y )
in ( 0 ...
4 , 0 ... 4 )
A [ x , y ]
=
B [ x , y ]
xor ( ( not
B [ x + 1 ,
y ] ) and
B [ x + 2 ,
y ] )

```

ι (iota) : Pseudo-Code

```

 $\chi$  (chi) : Pseudo-Code
A [ 0 , 0 ]
=
A [ 0 , 0 ]
xor RC

```

Năm chức năng này được thực hiện lặp đi lặp lại 24 lần. Sau 24 vòng tính toán kết thúc, chúng tôi nhận được 1600 bit, sau đó chúng tôi tách riêng tùy thuộc vào độ dài của bit 'r' và 'c' và quá trình tiếp tục.

4. Chức năng Bóp

Chức năng vắt bắt đầu ngay lập tức sau khi chúng tôi kết thúc chức năng hấp thụ. Chúng tôi gọi nó là hàm bóp vì đây là bước mà chúng tôi trích xuất thông điệp băm của mình. Cách mà chúng tôi giải nén nó vô cùng đơn giản và dễ hiểu.

Khi tính toán băm, chúng ta đã biết độ dài đầu ra của giá trị băm có thể là 224, 256, 384 hoặc 512. Sau khi hoàn thành hàm hấp thụ, chúng ta nhận được đầu ra có độ dài 1600 bit cuối cùng. Chúng tôi tách riêng đầu ra trên cơ sở độ dài của bit 'r' và 'c' tùy thuộc vào giá trị băm mà chúng tôi đang cố gắng tính toán, điều này dẫn chúng tôi đến đầu ra của chúng tôi.

Đầu ra:

Bây giờ, chúng ta có các giá trị cho 'r' và 'c', sau đó chúng ta trích xuất một vài bit đầu tiên từ 'r' tùy thuộc vào thuật toán băm, vì vậy đối với thuật toán SHA3-256, chúng ta sẽ trích xuất 256 bit đầu tiên từ 1088 bit của 'r' và đối với SHA3-512, chúng tôi sẽ trích xuất 512 bit đầu tiên từ 576 bit của 'r'. Giá trị được trích xuất từ các bit đầu tiên của 'r' là giá trị băm của toàn bộ thông báo.

Chương 3. Các điểm yếu và các dạng tấn công vào hàm băm

1. Các điểm yếu của SHA

- SHA0: Khi 2 thông điệp muốn mã hóa có giá trị gần như nhau, trong trường hợp này họ phát hiện 142 bit trong số 160 bit bằng nhau và các va chạm hoàn toàn của SHA-0 giảm xuống còn 62 trong 80 vòng theo nghiên cứu trước đó của thuật toán. SHA0 không được sử dụng nhiều và được thay thế bởi SHA1.
- SHA1: Năm 2005, điểm yếu mật mã đã được phát hiện trong SHA-1. Hàm băm được thiết kế để giảm thiểu xác suất 2 dữ liệu đầu vào khác nhau lại cho đầu ra 2 giá trị băm giống nhau, điều đó có nghĩa là có thể có 2 dữ liệu đầu vào khác nhau cho đầu ra là 2 giá trị băm giống nhau, theo Cryptographic hash collision.
- Khi sử dụng SHA để mã hóa mật khẩu: phương pháp này cũng không khó khăn cho những kẻ tấn công khi chúng có thư viện mật khẩu.
- Hiện nay SHA-1 không còn được coi là an toàn tuyệt đối tuy SHA-2 vẫn an toàn tuy nhiên thuật toán của SHA-2 không khác biệt nhiều so với SHA-1 do đó việc phá được nó chỉ là không lâu nữa. Do đó cần phải phát triển các thuật toán băm mới an toàn hơn cho tương lai.

2. Các cuộc tấn công:

a. Tấn công đụng độ (Collision Attack)

- Là một cuộc tấn công vào hàm băm trong mật mã học, trong đó tin tặc sẽ tìm ra hai input sinh ra cùng một giá trị băm.
- Có 2 loại collision attack là:
 - Classical collision attack:
 - Về mặt toán học, một cuộc tấn công va chạm tìm thấy hai thông điệp khác nhau m_1 và m_2 , sao cho $\text{hash}(m_1) = \text{hash}(m_2)$
 - Giống như mật mã khóa đối xứng(symmetric-key ciphers) dễ bị tấn công duyệt toàn bộ brute force, mọi hàm băm dễ bị tấn công va chạm(collision attack) khi sử dụng phương pháp tấn công sinh nhật(birthday attack). Do sinh nhật, những cuộc tấn công này nhanh hơn nhiều so với duyệt toàn bộ. Một hàm băm gồm n bit có thể bị phá vỡ trong $2^{n/2}$ thời gian (các đánh giá của hàm băm).
 - Các cuộc tấn công hiệu quả hơn có thể bằng cách sử dụng phân tích mật mã cho các hàm băm cụ thể. Khi một cuộc tấn công va chạm được phát hiện và được phát hiện là nhanh hơn một cuộc tấn công sinh nhật thường một hàm băm thường "bị hỏng". các cuộc tấn công va chạm đã được công bố chống lại hai hàm băm được sử dụng rất phổ biến ngày trước là MD5 và SHA-1. Các cuộc tấn công va chạm chống lại MD5 đã được cải thiện rất nhiều, đến năm 2007, nó chỉ mất vài giây trên một máy tính thông thường. Các xung đột băm được tạo ra theo cách này thường có độ dài không đổi và phần lớn không có cấu trúc, do đó không thể trực tiếp áp dụng để tấn công các định dạng hoặc giao thức tài liệu phổ biến.
 - SHA-1 – một trong những thuật toán băm được sử dụng rộng rãi nhất Internet đã bị giải mã dễ dàng với tấn công đụng độ:
 - Vào đầu năm 2005, Vincent Rijmen và Elisabeth Oswald đã công bố một cuộc tấn công vào SHA-1 - 53 với 80 vòng và phát hiện sự va chạm với số vòng là 80 . Cùng năm đó Xiaoyun Wang, Andrew Yao and Frances Yao at the CRYPTO 2005 Rump Session đã giảm số vòng xuống là 63
 - Vào tháng 2 năm 2017, CWI Amsterdam và Google thông báo họ đã thực hiện một cuộc tấn công va chạm chống lại SHA-1, xuất bản hai tệp PDF khác nhau tạo ra cùng một hàm băm SHA-1
 - Chosen-prefix collision attack:
 - Đây phần mở rộng của cuộc tấn công xung đột, cụ thể được sử dụng cho các hàm băm Merkle – Damgård. Trong trường hợp này, kẻ tấn công chọn hai tài liệu khác nhau tùy ý, sau đó nối các giá trị được tính toán khác nhau dẫn đến toàn bộ tài liệu có giá trị băm bằng nhau. Cách tấn công này thì khó hơn nhiều so với kiểu cổ điển.
 - Về mặt toán học, với hai tiền tố khác nhau p_1, p_2 , cuộc tấn công tìm thấy hai phần phụ m_1 và m_2 sao cho $\text{băm}(p_1 \parallel m_1) = \text{băm}(p_2 \parallel m_2)$ (trong đó \parallel là phép toán nối)
 - Vào năm 2007, một cuộc tấn công xung đột tiền tố đã chọn đã được phát hiện chống lại MD5, yêu cầu khoảng 50 rounds đánh giá của chức năng MD5.

b. Tấn công hàm Hash theo kiểu ngày sinh nhật

- Birthday attack hay tấn công sinh nhật là một loại tấn công mật mã thuộc loại tấn công duyệt toàn bộ(brute force attack) . Nó khai thác một vấn đề toán học về sinh nhật trong lý thuyết xác suất. Sự thành công của cuộc tấn công này phần lớn phụ thuộc vào khả năng xảy ra va chạm(collission attack) cao hơn khi tìm thấy giữa các lần tấn công ngẫu nhiên với một mức độ hoán vị cố định, Được mô tả trong bài toán nghịch lý sinh nhật dưới đây.
- Xét ví dụ về một lớp học có 30 học sinh và một giáo viên. Giáo viên mong muốn tìm được các cặp học sinh có ngày sinh giống nhau. Do đó giáo viên yêu cầu sinh nhật của mọi người để tìm những cặp như vậy. Về mặt trực quan, giá trị này có vẻ nhỏ. Ví dụ, nếu giáo viên ấn định một ngày cụ thể là ngày 1 tháng 1, thì xác suất để ít nhất một học sinh sinh vào ngày đó là $1 - (364/365)^{30}$ là khoảng 7,9%. Tuy nhiên, xác suất để ít nhất một sinh viên có cùng ngày sinh với bất kỳ sinh viên nào khác là khoảng 70% theo công thức sau:

$$1 - 365!/(365-n!)*(365^n) \text{ với } n = 30$$

Chứng minh:

Giả sử 1 năm có 365 ngày và mỗi người có xác suất để trùng với ngày sinh người khác là như nhau

Coi $n = 2$, P là xác suất 2 người có cùng ngày sinh và P' là xác suất 2 người có ngày sinh khác nhau

$$\begin{aligned} P &= 1 - P' \\ &= 1 - (365/365)*(364/365) \\ &= 1 - 1*(364/365) \\ &= 1 - 364/365 \\ &= 1/365 \end{aligned}$$

Vậy với n người thì xác suất để tất cả có ngày sinh khác nhau là:

$$\begin{aligned} P'(n) &= (365/365)*(365-1/365)*(365-2/365)*....(365-n+1)/365 \\ &= 365!/((365-n)! * 365^n) \end{aligned}$$

Áp dụng vào hàm hash:

Ta có đầu vào là m và đầu ra $H(m)$ với đầy đủ các tính chất của 1 hàm hash

Với đầu vào giả sử là x , 1 hàm hash gọi là hàm hash không va chạm khi $H(x) \neq H(y)$ ngược lại nếu $H(x) = H(y)$ thì được coi là va chạm mạnh

Giờ coi đầu vào chỉ có bit 0 và bit 1 và độ dài n ta sẽ có hàm hash là $|M| \gg 2^n$

Theo thuật toán chung tìm va chạm tại độ phức tạp là $O(2^{n/2})$

Thuật toán :

1. Chọn $2^{n/2}$ đầu vào ngẫu nhiên ở $m : m_1, m_2, m_3, \dots m_{n/2}$

2. For $I = 1, 2, 3, \dots 2^{n/2}$ có được $H(m_i) \Rightarrow \{0,1\}^n$

3. Xét xem $H(m_i)$ có bằng $H(m_j)$ hay không? Nếu không quay trở lại bước 1

Từ một tập các giá trị H , chúng ta chọn ngẫu nhiên đồng nhất n giá trị do đó cho phép lặp lại. Gọi $p(n; H)$ là xác suất để trong quá trình thử nghiệm này ít nhất một giá trị được chọn nhiều hơn một lần. Xác suất này có thể được tính gần đúng như sau:

$$p(n; H) = 1 - ((365-1) / 365) * (365-2) / 365 * \dots (365-n + 1/365))$$

c. Tấn công gặp nhau ở giữa(Meet-in-the-middle)

- Một cuộc tấn công Gặp nhau giữa (MitM) là một loại tấn công phân tích mật mã trong đó kẻ tấn công sử dụng một số loại đánh đổi không gian hoặc thời gian để hỗ trợ cuộc tấn công.
- Một mật mã, sẽ bị phá vỡ bằng cách sử dụng cuộc tấn công gặp gỡ giữa, có thể được định nghĩa là hai thuật toán, một thuật toán để mã hóa và một để giải mã.
- Với C là bản mã:

P là bản chữ viết

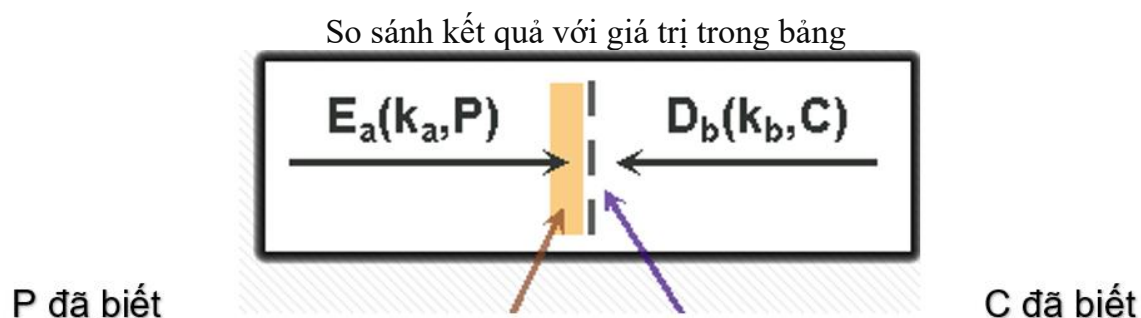
E là thuật toán mã hóa

D là thuật toán giải mã

K_b, k_a là 2 khóa bí mật

Bước đầu tiên của cuộc tấn công là tạo một bảng với tất cả các giá trị có thể có cho một mật của phương trình. nên tính toán tất cả các bản mã có thể có của P đã biết được tạo ra bằng cách sử dụng khóa bí mật đầu tiên, do đó, $E_a(k_a, P)$. Một số hàng trong bảng bằng một số khóa bí mật có thể có. sắp xếp bảng đã nhận dựa trên các mật mã nhận được $E_a(k_a, P)$, để đơn giản hóa việc tìm kiếm thêm.

Bước thứ hai của cuộc tấn công là tính toán các giá trị $D_b(k_b, C)$ cho về thứ hai của phương trình. so sánh chúng với các giá trị của về đầu tiên của phương trình, được tính toán trước đó và được lưu trữ trong bảng. Kẻ tấn công tìm kiếm một cặp khóa bí mật k_a và k_b , trong đó giá trị $E_a(k_a, P)$ được tìm thấy trong bảng và giá trị vừa tính được $D_b(k_b, C)$ giống nhau.



Kazumaro Aoki; Jian Guo; Krystian Matusiewicz; Yu Sasaki & Lei Wang năm 2009 đã công bố nghiên cứu về cuộc tấn công gặp nhau ở giữa vào tính chất PreImage của hàm hash sha-2 .

Chương 4. Các ứng dụng của hàm băm

1. Che dấu dữ liệu gốc

Hash được sử dụng cho mật mã vì nó che dấu dữ liệu gốc bởi một giá trị khác. Vì sử dụng trong mật mã nên một "**hàm băm tốt**" là một hàm băm không thể đảo ngược. Ví dụ về việc sử dụng hash trong xác minh mật khẩu.

Khi lưu password trong cơ sở dữ liệu (database) người ta thường dùng kỹ thuật "password hashing" chứ không lưu trực tiếp password. Vì nếu database của chúng ta bị xâm phạm (bởi các lực lượng bên ngoài hoặc bên trong, bên trong ở đây có thể là chính người quản trị cơ sở dữ liệu, bên ngoài ở đây có thể là các hacker, kẻ phá hoại,...) thì tất cả các thông tin về user trong đó có password sẽ bị lộ.

Thay vì lưu password dưới dạng plaintext (dạng text bình thường mà con người có thể đọc hiểu được) thì password sẽ được đưa vào một hash function, hash value sinh ra từ hash function sẽ được dùng để lưu và database đại diện cho password. Ta sẽ thấy rõ hơn mô hình này qua sơ đồ dưới đây

Cụ thể, nếu đăng ký tài khoản trên trang web với username là "Student" và password là "12345678", webserver sử dụng thuật toán hash để tính hash value của password và lưu xuống database. Giả sử, hash value của "12345678" là "7c222fb2927d828af22f592134e8932480637c0d"

Lần sau, khi đăng nhập nếu nhập "12345678" trong ô password thì webserver sẽ lấy hash value của dữ liệu vừa nhập vào và so sánh với giá trị "7c222fb2927d828af22f592134e8932480637c0d" lưu trong database. Trường hợp hai giá trị hash bằng nhau thì đăng nhập thành công. Ngược lại, nếu nhập "123456789" thì có giá trị hash nhận được khi này sẽ là "f7c3bc1d808e04732adf679965ccc34ca7ae3441".

2. Hashing trong định danh dữ liệu

a. Định danh tệp hoặc dữ liệu

Giá trị băm cũng có thể được sử dụng như một phương tiện để định danh tập tin một cách đáng tin cậy. Một số hệ thống quản lý mã nguồn, như Git, Mercurial hay Monotone, sử dụng giá trị sha1sum của nội dung tệp, cây thư mục, thông tin thư mục gốc, v.v. để định danh chúng. Những giá trị này còn được sử dụng để xác định các tệp trên các mạng chia sẻ ngang hàng nhằm cung cấp đầy đủ thông tin để định vị nguồn gốc của tệp, xác minh nội dung tệp tải xuống. Giá trị ứng dụng của chúng còn được mở rộng ra khi áp dụng các cấu trúc dữ liệu bổ sung như danh sách băm hoặc cây băm.

b. Tạo và xác nhận giá trị tổng kiểm hoặc chữ ký số

Checksum là giá trị tổng kiểm, được sử dụng với mục đích xác nhận tính toàn vẹn của file. Checksum là giá trị nhỏ được tạo dựa trên các bit trong một file hoặc khối dữ liệu như image đĩa. Khi chức năng checksum chạy trên một bản sao của file (chẳng hạn như

file được tải từ Internet), nó sẽ tạo ra giá trị băm giống như file gốc. Nếu file không tạo cùng một checksum, có nghĩa là file đã bị thay đổi. Ngoài ra, cũng giống như checksum, hash cũng được sử dụng trong tạo và xác nhận chữ ký số an toàn và hiệu quả.

3. Ứng dụng trong lập chỉ mục và truy xuất dữ liệu

Một trong những ví dụ cụ thể mà mình biết đó là **hashtable**. Nếu bạn đã từng học lập trình thì hashtable được hỗ trợ trong rất nhiều các ngôn ngữ, là một trong những nền tảng về thuật toán và cấu trúc dữ liệu.

Cho các bạn chưa biết thì: hash table là một cấu trúc dữ liệu dùng để lưu theo các cặp key-value, nó dùng hash function để tính toán ra một index (nơi lưu trữ một bucket các giá trị) rồi từ đó sẽ retrieve ra value.

Giả sử ta có các object với data bất kỳ. Điều đầu tiên, khi đưa dữ liệu qua hash function thì lợi ích chúng ta có được là dữ liệu đầu ra được đồng bộ (đó là hash codes). Tuy nhiên, kích thước bảng vẫn còn lớn nên ta sẽ sử dụng kỹ thuật “Modulo table size” để giảm kích thước xuống và có được một hash table.

Một trong những vấn đề của hash đó là “collision” (trùng vị trí index). Điều này sẽ xảy ra nếu thuật toán hash không được tốt. Và thực tế, hầu như không có thuật toán hash nào thực sự hoàn hảo để sinh ra unique key nếu lưu trữ một lượng lớn dữ liệu. Giải pháp cho vấn đề này là dùng linked list để lưu trữ thêm một tầng nữa các phần tử cho index đó.

Có thể coi việc sinh hash là để tạo keyHash và lưu vào trong mảng giá trị nơi chứa một linked list có phần tử chứa key thật, sau khi truy xuất tới đó chúng ta sẽ dùng key thật để retrieve giá trị.

Như vậy, việc sử dụng hashtable sẽ giúp tạo một chỉ mục nhỏ hơn nhiều so với dữ liệu gốc. Nhờ đó việc tìm kiếm và truy cập dữ liệu sẽ hiệu quả hơn.

4. Ứng dụng trong bằng chứng công việc (Proof of Work)

Proof of work (PoW) là một biện pháp kinh tế để ngăn chặn các cuộc tấn công từ chối dịch vụ và các hành vi lạm dụng dịch vụ khác như spam bằng cách yêu cầu người dùng dịch vụ thực hiện một số công việc nhất định, thường đòi hỏi nhiều thời gian xử lý. Bằng chứng công việc cần đảm bảo tính bất đối xứng tức là: công việc phải có độ khó vừa phải (nhưng khả thi) về phía người dùng nhưng dễ kiểm chứng đối với nhà cung cấp dịch vụ.

PoW cũng là thuật toán đồng thuận đầu tiên được tạo ra trong mạng Blockchain. Được sử dụng để xác nhận giao dịch và sản xuất các block mới trong chuỗi.

PoW yêu cầu những người sở hữu các máy tính trong mạng phải giải một bài toán phức tạp để có thể thêm một block (khối) vào chuỗi. Tuy nhiên, không dễ để có câu trả lời cho vấn đề toán học.

Chương 5. Cài đặt thử nghiệm

Ngôn ngữ lập trình java có hỗ trợ class chuyên dụng để tạo hàm băm là MessageDigest. Class này hỗ trợ 3 thuật toán băm tiêu chuẩn là MD5, SHA-1 và SHA-256.

```
1 package baiTap.sha1.main;
2
3 import java.security.MessageDigest;
4 import java.security.NoSuchAlgorithmException;
5
6 public class Main {
7
8     Run | Debug
9     public static void main(String[] args) throws NoSuchAlgorithmException{
10         System.out.println("");
11         System.out.println("");
12         System.out.println("");
13         System.out.println("");
14         System.out.println(sha1("Co so an toan thong tin"));
15         System.out.println("");
16         System.out.println("");
17         System.out.println("");
18     }
19     static String sha1 (String input) throws NoSuchAlgorithmException{
20         MessageDigest mDigest = MessageDigest.getInstance("SHA");
21         byte[] byteArray = mDigest.digest(input.getBytes());
22         StringBuffer res = new StringBuffer();
23         for (int i = 0; i < byteArray.length; i++){
24             res.append(Integer.toString((byteArray[i] & 0xff) + 0x100, 16).substring(1));
25         }
26         return res.toString();
27     }
28 }
```

Kết quả:

```
PS G:\Cua Hung\ĐẠI HỌC\trên lớp\k1n3\java\baiTap\baiTap> g.; cd '
dmin\.vscode\extensions\vscjava.vscode-java-debug-0.36.0\scripts\l
ble-preview' '-XX:+ShowCodeDetailsInExceptionMessages' '-Dfile.enc
oding=UTF-8' '-cp' 'C:\Users\admin\AppData\Roaming\Code\User\works
paceStorage\20d6a829ffe2297d69496d7c613efee8\redhat.java\jdt_ws\jd
t.ls-java-project\bin' 'baiTap.sha1.main.Main'

e0cf6873a143375e8673155d3864d6eaa66cdd9d
```

So sánh với kết quả của SHA-1 online:

Home Page | [SHA1 in JAVA](#) | [Secure password generator](#) | [Linux](#)

SHA1 and other hash functions online generator

Co so an toan thong tin hash

sha-1

Result for

sha1: e0cf6873a143375e8673155d3864d6eaa66cdd9d

Kết luận :

Bài báo cáo đã trình bày được các khái niệm tổng quát về hàm băm mật mã, hàm băm SHA cùng với các tính chất, phân loại và ứng dụng của chúng từ đó ứng dụng vào phân tích các giải thuật băm SHA để thấy được hoạt động cũng như sự thay đổi trong các phiên bản của họ hàm băm SHA. Với các tính chất của hàm băm một chiều SHA đã được sử dụng vào hầu hết các ứng dụng thương mại điện tử, xác thực mật khẩu, thông điệp và chữ ký số giúp làm giảm thời gian mã hóa/ ký số, đưa ra được kết quả duy nhất cho mã hóa, đảm bảo tính toàn vẹn của thông tin và giảm thiểu thời gian truyền tin qua mạng.

Bài báo cáo cũng đi sâu tìm hiểu vào một số điểm yếu, các dạng tấn công vào SHA để tìm ra giải pháp nghiên cứu mới, các hướng đi mới. Mặc dù SHA-1 được tuyên bố không an toàn bởi các nhà nghiên cứu từ hơn một thập kỷ trước và Microsoft trong tháng 10/2013 đã công bố sau năm 2016 sẽ không chấp nhận chứng thư số SHA-1, tuy nhiên SHA1 vẫn được sử dụng rộng rãi trên Internet.

Tài liệu tham khảo

1. <https://sectigostore.com/blog/hash-function-in-cryptography-how-does-it-work/>
2. Bài giảng cơ sở an toàn thông tin – Hoàng Xuân Dậu
3. Wikipedia https://vi.wikipedia.org/wiki/H%C3%A0m_b%C4%83m
4. <https://cryptoviet.com/hash-la-gi>
5. <https://www.aditya12anand.com/post/breaking-down-sha-3-algorithm>
6. Chabaud F, Joux A (1998) Differential collisions in SHA-0. In: Krawczyk H (ed) Advances in cryptology – CRYPTO'98. Lecture notes in computer science, vol 1462. Springer, Berlin, pp 56–71
7. De Canniere C, Rechberger C (2008) Preimages for reduced SHA-0 and SHA-1. In: Wagner D (ed) Advances in cryptology – CRYPTO 2008. Lecture notes in computer science, vol 5157. Springer, Berlin, pp 179–202\
8. Security hash standard
9. Advanced meet-in-the-middle preimage attacks: First results on full Tiger, and improved results on MD4 and SHA-2
10. Hash Collision Q&A".
11. <https://www.geeksforgeeks.org/birthday-attack-in-cryptography/>
12. https://en.wikipedia.org/wiki/Collision_attack
13. VnReview.vn