Tiểu luận ATBM - SHA - Bài tập lớn môn ATBM - Đề tài SHA

An toàn ứng dụng web và cơ sở dữ liệu (Học viện Công nghệ Bưu chính Viễn thông)

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN

Học phần: **An toàn bảo mật hệ thống thông tin**

Bài báo cáo:

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 hướng dẫn: TS. Đặng Minh Tuấn
Sinh viên thực hiện: Nhóm 11

Bùi Quang Danh	B15DCCN102
Ngô Thị Thu Hân	B15DCCN192
Đỗ Đình Tiến	B15DCCN553
Lê Tất Tiến	B15DCCN554

Hà Nội 2017

Mục lục

DANH SÁCH CÁC THUẬT NGỮ TIẾNG ANH VÀ VIẾT TẮT

Từ viết tắt	**Thuật ngữ tiếng Anh/Giải thích**	**Thuật ngữ tiếng Việt/Giải thích**
AES	Advanced Encryption Standard	Chuẩn mã hóa tiên tiến
OWHF	One Way Hash Functions	Hàm băm một chiều
CRHF	Collision Resistant Hash Functions	Hàm băm kháng xung đột
IPSec	Internet Protocol Security	An toàn giao thức Internet
MAC	Message Authentication Code	Mã xác thực thông điệp (sử dụng hàm băm có khóa)
MD	Message Digest	Chuỗi đại diện thông điệp
MDC	Modification Detection Code	Mã phát hiện sử đổi (sử dụng hàm băm không khóa)
NIST	National Institute of Standards and Technology	Viện Tiêu chuẩn và Công nghệ

NSA	National Security Agency	Cơ quan mật vụ liên bang Mỹ
PGP	Pretty Good Privacy	Chuẩn bảo mật PGP
PKI	Public Key Infrastructure	Hạ tầng khóa công khai
RSA	RSA Public Key Croptosystem	Hệ mật khóa công khai RSA
SHA	Secure Hash Algorithm	Giải thuật băm an toàn
SHS	Secure Hash Standard	Tiêu chuẩn băm an toàn
SSL/TLS	Secure Socket Layer / Transport Layer Security	Bộ giao thức bảo mật SSL / TLS

DANH MỤC CÁC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC HÌNH VẼ

LỜI MỞ ĐẦU

Với sự phát triển ngày càng nhanh chóng của Internet và các ứng dụng giao dịch điện tử trên mạng, nhu cầu bảo vệ thông tin trong các hệ thống và ứng dụng điện tử ngày càng được quan tâm và có ý nghĩa hết sức quan trọng. Vì thế việc nghiên cứu về chuẩn mật mã nâng cao và ứng dụng nó trong các lĩnh vực bảo mật thông tin là rất cần thiết.

Ứng dụng của chuẩn mật mã nâng cao đang được sử dụng ngày càng phổ biến trong nhiều ứng dụng khác nhau. Chuẩn mật mã nâng cao không chỉ đơn thuần là mã hóa và giải mã thông tin mà còn bao gồm nhiều vấn đề khác nhau cần được nghiên cứu và giải quyết như ứng dụng xây dựng hàm băm phục vụ việc chứng thực nguồn gốc nội dung thông tin (kỹ thuật chữ ky điện tử), xác thực tính nguyên vẹn dữ liệu,…

Một trong những hàm băm đang được sử dụng rộng rãi nhất hiện nay là hàm băm SHA được phát triển bởi cục an ninh quốc gia Mỹ (National Security Agency hay NSA). Với nhiều ưu điểm và cũng có nhiều phiên bản khác nhau được phát hành. Với bài tiểu luận với đề tài " Tìm hiểu giải thuật băm SHA0,1,2,3. Các điểm yếu, các dạng tấn công vào SHA. Cài đặt thử nghiệm SHA1" chúng ta sẽ cùng tìm hiểu về các hàm băm SHA và ứng dụng của nó để hiểu rõ hơn và tiến hành thử nghiệm kiểm chứng.

Chương 1. Khái quát về hàm băm mật mã

1. 1.Giới thiệu sơ lược về hàm băm mật mã

Hiểu theo nghĩa đơn giản, hàm băm là hàm cho tương ứng một mảng dữ liệu lớn với một mảng dữ liệu nhỏ hơn mà được dùng rộng rãi trong nhiều ứng dụng tin học, không chỉ thuộc phạm vi mật mã. Ở đây, chúng ta chỉ xét đến các hàm băm trong phạm vi các hàm băm mật mã, xem xét cụ thể đến các ứng dụng của chúng trong việc đảm bảo tính toàn vẹn của dữ liệu.

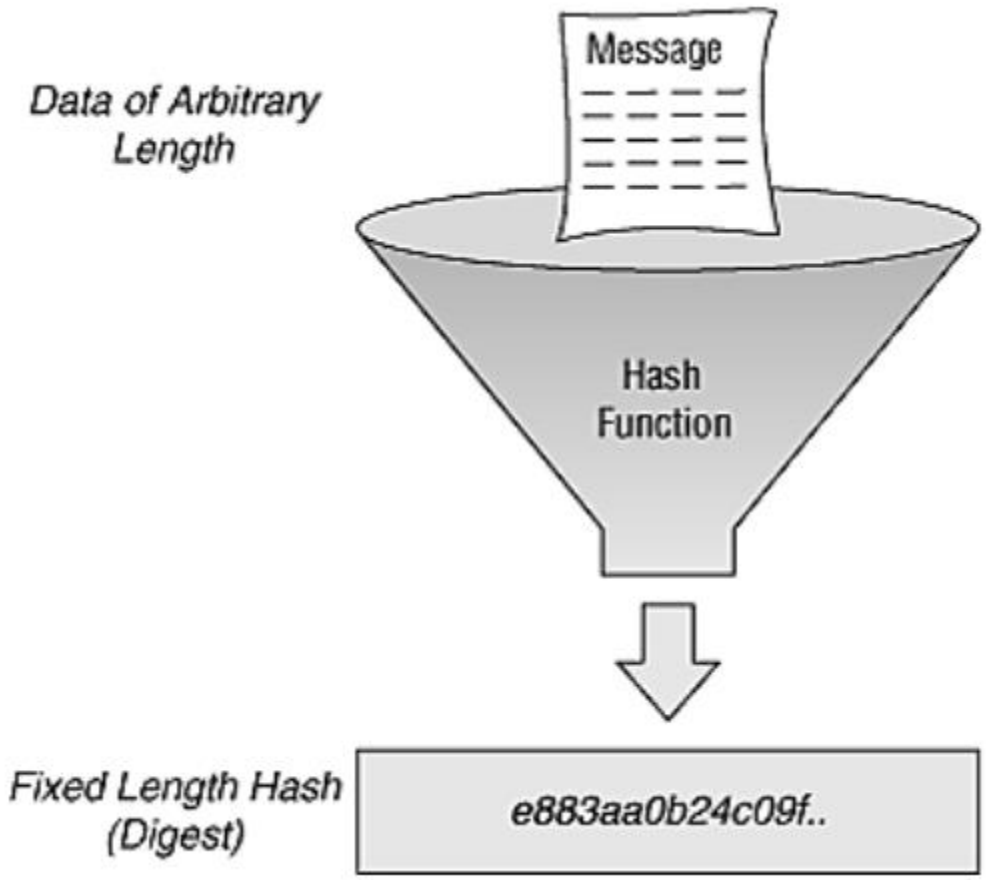
Các hàm băm nhận đầu vào là một chuỗi bit có chiều dài hữu hạn tùy ý và tạo ra một chuỗi bit có chiều dài cố định bằng n bit (n > 0) gọi là mã băm (hash code).

Trong mã hóa, mã băm được xem như là ảnh đại diện thu gọn (compact representative image) của một chuỗi bit có độ dài hữu hạn tùy ý và được dùng để nhận diện cho chuỗi bit đó. Kết hợp với công cụ tạo chữ ký số, các hàm băm được dùng cho việc đảm bảo tính toàn vẹn của dữ liệu. Trong lược đồ chữ ký số, mã băm của chuỗi bit được tính ở thời điểm T_1 và được bảo vệ để chống lại mọi sự thay đổi bất hợp pháp. Tại thời điểm T_2 sau đó, để kiểm tra xem chuỗi bit x có bị thay đổi hay không, người ta thường tính giá trị hàm băm của chuỗi bit này tại thời điểm T_2, mà ta ký hiệu là x_{T2}, sau đó so sánh giá trị vừa tính với mã băm tại thời điểm T_1. Nếu 2 giá trị bằng nhau thì người ta chấp nhận chuỗi bit tại thời điểm T_2 trùng khớp với chuỗi bit tại thời điểm T_1, tức chuỗi bit x vẫn chưa bị thay đổi. Như vậy vấn đề bảo đảm tính toàn vẹn của chuỗi bit có chiều dài tùy ý được thay bằng việc bảo vệ sự toàn vẹn của chuỗi bit có chiều dài cố định.

1.2. 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]



Hình 1: Mô hình nén dữ liệu của hàm băm

Hình 1.1 minh họa mô hình nén thông tin của hàm băm, theo đó thông điệp (Message) đầu vào với chiều dài tùy ý đi qua nhiều vòng xử lý của hàm băm để tạo chuỗi rút gọn, hay chuỗi đại diện (Digest) có kích thước cố định ở đầu ra.

1.3. Tính chất của hàm băm mật mã

Một hàm băm mật mã lý tưởng có các tính chất sau :

1.3.1. Tính kháng tiền ảnh (Preimage resistance)

Với mọi đầu ra y cho trước, không thể tìm được bất kỳ dữ liệu đầu vào x sao cho h(x) = y (hay không thể tìm được một thông điệp từ một giá trị băm cho trước).

1.3.2. Tính kháng tiền ảnh thứ hai (2nd - Preimage resistance)

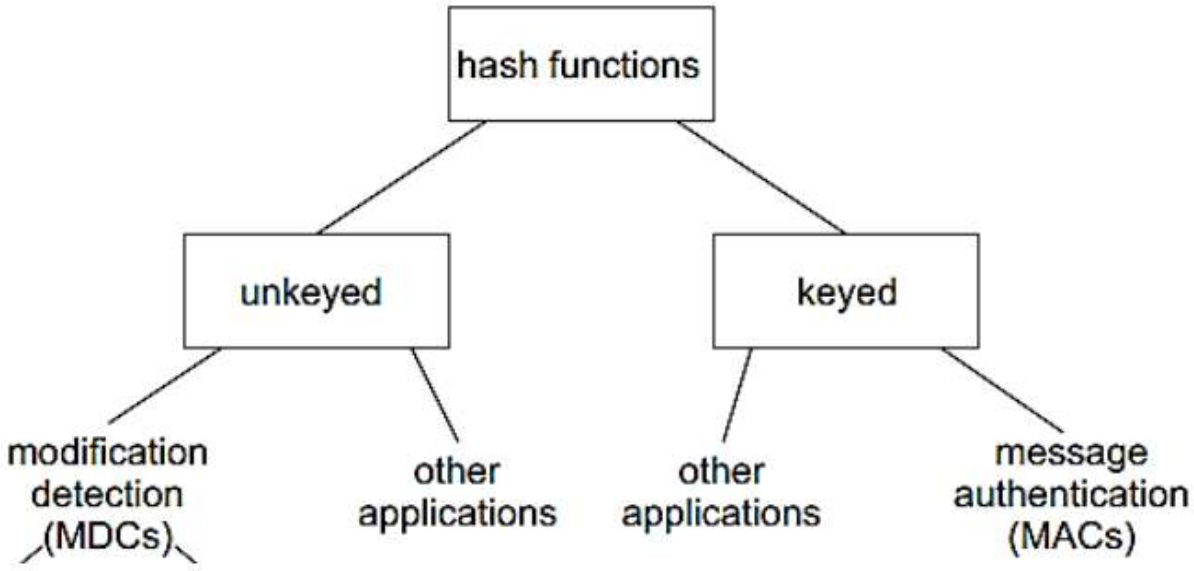
Với mọi dữ liệu đầu vào x cho trước và y = h(x), không thể tính toán để tìm ra được giá trị $x' \neq x$ sao cho $h(x')=h(x)$ (hay không thể tìm ra 2 thông điệp khác nhau mà có cùng giá trị băm).

1.3.3. Tính kháng xung đột (Collision resistance)

Không thể tính toán để tìm được hai dữ liệu đầu vào x và x’ phân biệt sao cho chúng có cùng giá trị băm h(x)=h(x’) (hay không thể sửa được một thông điệp mà không làm thay đổi giá trị băm của nó).

1.4. Phân loại hàm băm mật mã

Có thể phân loại các hàm băm theo khóa sử dụng hoặc theo chức năng. Theo khóa sử dụng, các hàm băm gồm 2 loại: hàm băm không khóa (unkeyed) và hàm băm có khóa (keyed), như biểu diễn trên Hình 1.2. Trong khi hàm băm không khóa nhận đầu vào chỉ là thông điệp (dạng h(x), với hàm băm h và thông điệp x), hàm băm có khóa nhận đầu vào gồm thông điệp và khóa bí mật (theo dạng h(x, K), với hàm băm h và thông điệp x và K là khóa bí mật). Trong các hàm băm không khóa, các mã phát hiện sửa đổi (MDC – Modification Detection Code) được sử dụng rộng rãi nhất, bên cạnh một số hàm băm không khóa khác. Tương tự, trong các hàm băm có khóa, các mã xác thực thông điệp (MAC - Message Authentication Code) được sử dụng rộng rãi nhất, bên cạnh một số hàm băm có khóa khác.[1]



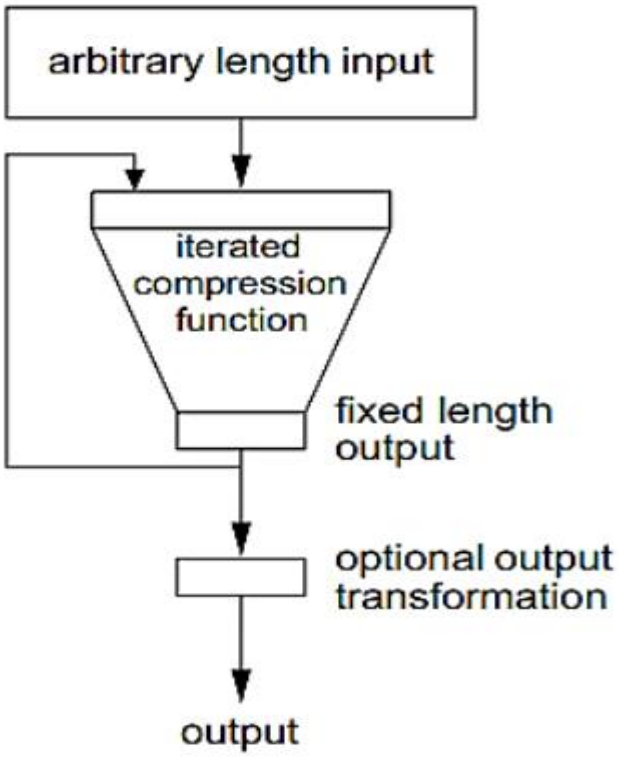
Hình 2Phân loại các hàm băm theo khóa sử dụng

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 [1].

1.5. Mô hình xử lý dữ liệu

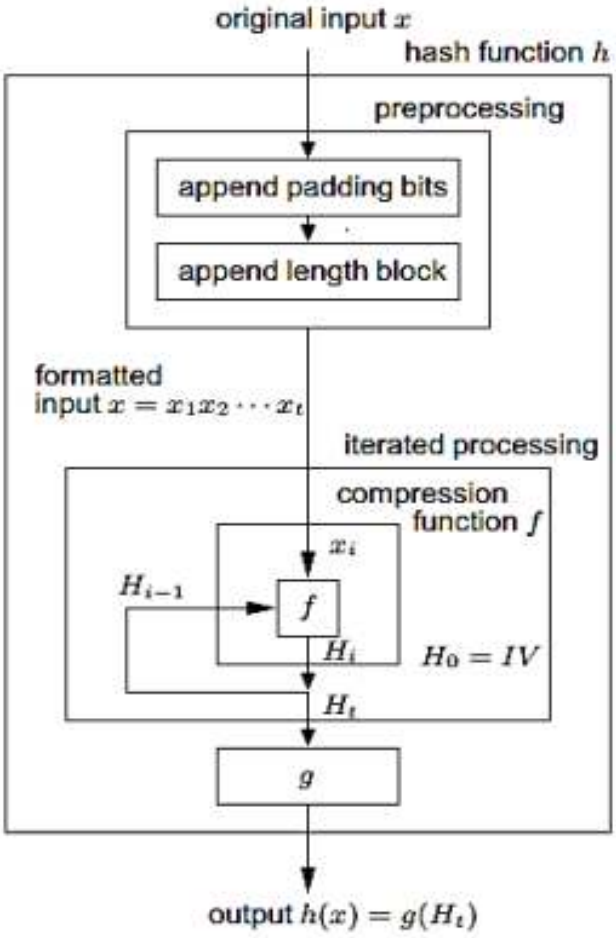


Hinh 1 . 1 Mô hình tổng quát sử lý dữ liệu của hàm băm

Hình 1.3 biểu diễn mô hình tổng quát xử lý dữ liệu của các hàm băm. Theo đó, thông điệp đầu vào với độ dài tùy ý (arbitrary length input) đi qua hàm nén lặp nhiều vòng

(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).

Hình 1.4 mô tả chi tiết quá trình xử lý dữ liệu của các hàm băm. Theo đó, 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 $x = x_1 x_2 \ldots x_t$ làm đầu vào cho bước 2. Trong bước 2, từng khối dữ liệu xi 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 hết quả h(x).



Hinh 1 . 2 Mô hình chi tiết xử lý dữ liệu của hàm băm

1.6. Ứng dụng của hàm băm

1.6.1 Xác thực mật khẩu

Mật khẩu thường không được lưu dưới dạng văn bản rõ (clear text), mà ở dạng tóm tắt. Để xác thực một người dùng, mật khẩu do người đó nhập vào được băm ra bằng hàm Hash và so sánh với kết quả băm được lưu trữ.

1.6.2. Xác thực thông điệp

Giá trị đầu vào (tin nhắn, dữ liệu…) bị thay đổi tương ứng giá trị băm cũng bị thay đổi. Do vậy nếu một kẻ tấn công phá hoại, chỉnh sửa dữ liệu thì sever có thể biết ngay lập tức.

1.6.3. Bảo vệ tính toàn vẹn của tập tin, thông điệp được gửi qua mạng

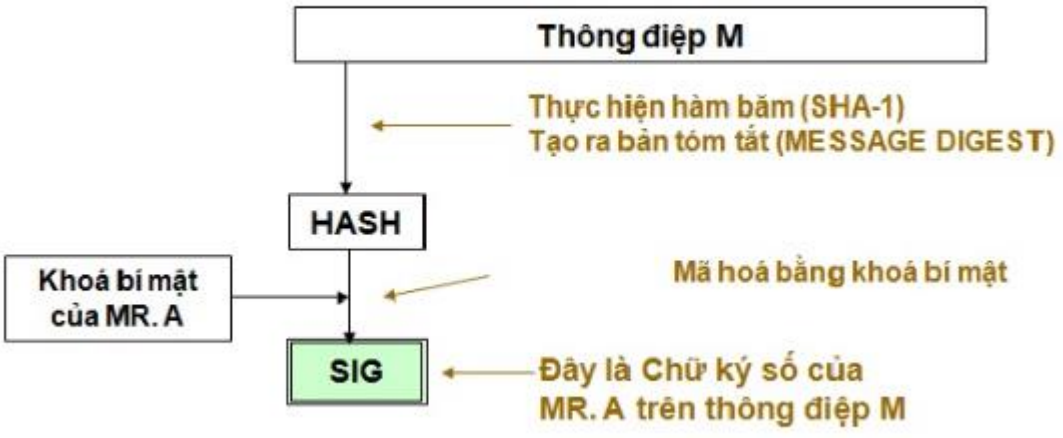
Hàm băm mật mã có tính chất là hàm băm một chiều. Từ khối dữ liệu hay giá trị đầu vào chỉ có thể đưa ra một giá trị băm duy nhất. Như chúng ta đã biết đối với tính chất của hàm một chiều. Một người nào đó dù bắt được giá trị băm của họ cũng không thể suy ngược lại giá trị, đoạn tin nhắn băm khởi điểm.

Ví dụ: việc xác định xem một file hay một thông điệp có bị sửa đổi hay không có thể thực hiện bằng cách so sánh tóm tắt được tính trước và sau khi gửi (hoặc một sự kiện bất kỳ nào đó). Còn có thể dùng tóm tắt thông điệp làm một phương tiện đáng tin cậy cho việc nhận dạng file.

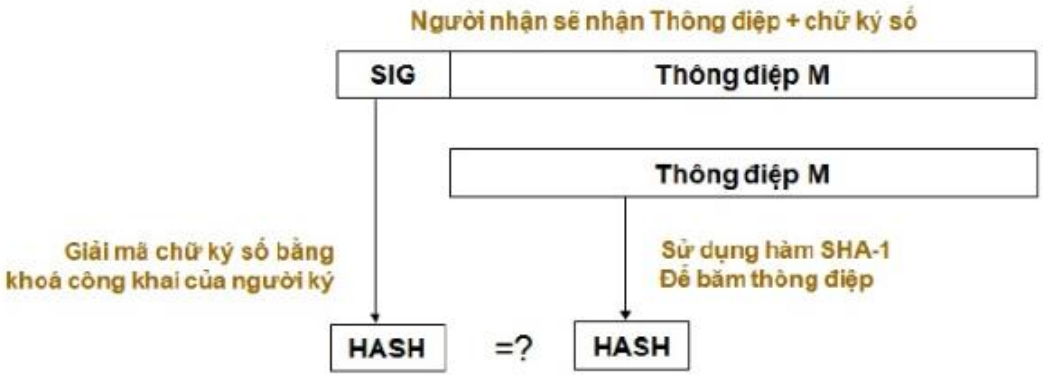
Hàm băm thường được dùng trong bằng băm nhằm giảm chi phí tính toán khi tìm một khối dữ liệu trong một tạp hợp. Giá trị băm đóng vai trò gần như một khóa để phân biệt các khối dữ liệu.

1.6.4. Tạo chữ ký điện tử (Digital signatures)

Chữ ký số có được bằng cách đem mã hóa bản tóm tắt của thông điệp bằng khóa bí mật của người ký. Mô tả trong hình 1.5.



Hinh 1 . 3 Chữ ký số



Hinh 1 . 4 Chứng thực bằng chữ ký số

Hình 1.6 biểu diễn quá trình chứng thực thông điệp bằng chữ ký số . Nếu kết quả băm giống nhau thì thông điệp được xác thực. Vì nếu bất kỳ bit nào của M hay SIG bị thay đổi, kết quả băm sẽ khác.

Chương 2 Các giải thuật hàm băm SHA

2.1. Giới thiệu 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:

"for a given algorithm, it is computationally infeasible 1) to find a message that corresponds to a given message digest, or 2) to find two different messages that produce the same message digest. Any change to a message will, with a very high probability, result in a different message digest" [2].

Tạm dịch đại ý là:

"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.

Bất cứ thay đổi nào trên đoạn dữ liệu gốc, dù nhỏ, cũng sẽ tạo nên một giá trị băm hoàn toàn khác với xác suất rất cao."

Các thuật giải SHA là SHA-1 (trả lại kết quả dài 160 bit), 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), và SHA-512(trả lại kết quả dài 512 bit). Thuật giải SHA là thuật giải băm mật được phát triển bởi cục an ninh quốc gia Mĩ (National Security Agency hay NSA) và được xuất bản thành chuẩn của chính phủ Mĩ bởi viện công nghệ và chuẩn quốc gia Mĩ (National Institute of Standards and Technology hay NIST). Bốn thuật giải sau thường được gọi chung là SHA-2. SHA-3 là phiên bản mới nhất của NIST nó bao gồm 6 phiên bản SHA3-224 , SHA3-256 .,SHA3-384 ,SHA3-512, SHAKE128 ,SHAKE256 các phiên bản là khá giống nhau và được phát hành vào tháng 8 năm 2015.

SHA-1 được sử dụng rộng rãi trong nhiều ứng dụng và giao thức an ninh khác nhau, bao gồm TLS và SSL, PGP, SSH, S/MIME, và IPSec. SHA-1 được coi là thuật giải thay thế MD5, một thuật giải băm 128 bit phổ biến khác.

Hiện nay, SHA-1 không còn được coi là an toàn bởi đầu năm 2005, ba nhà mật mã học người Trung Quốc đã phát triển thành công một thuật giải dùng để tìm được hai đoạn dữ liệu nhất định có cùng kết quả băm tạo ra bởi SHA-1.[3] Mặc dù chưa có ai làm được điều tương tự với SHA-2, nhưng vì về thuật giải, SHA-2 không khác biệt mấy so với SHA-1 nên nhiều nhà khoa học đã bắt đầu phát triển một thuật giải khác tốt hơn SHA. NIST cũng đã khởi đầu một cuộc thi phát triển thuật giải băm mới an toàn hơn SHA, giống như quy trình phát triển chuẩn mã hóa tiên tiến (Advanced Encryption Standard hay AES).

2.2. Các giải thuật SHA

2.2.1 SHA-0

2.2.1.1. Giới thiệu 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

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.[4]

2.2.1.2. Giải thuật SHA-0

Hàm nén SHA-0 được xây dựng dựa trên cấu trúc Davis-Meyer. Nó sử dụng một hàm E như là một mật mã khối với H_t cho đầu vào tin nhắn và M_{t+1} cho đầu vào khóa, cần phải có một feed-forward để phá vỡ tính không thể đảo ngược của quá trình:

$H_{t+1} = E(H_t, M_{t+1}) \oplus H_t$,

Ở đây toán tử $\oplus$ biểu thị phép cộng modulo 2^{32} từ 32-bit bởi các từ 32-bit. Hàm này bao gồm 80 bước (4 vòng 20 bước), mỗi phần xử lý một từ tin 32 bit Wi để cập nhật 5 thanh ghi nội bộ 32-bit (A, B, C, D, E). Các feed-forward bao gồm việc cộng modulo 2^{32} trạng thái ban đầu với trạng thái cuối cùng của mỗi thanh ghi. Vì đã sử dụng nhiều bit tin hơn so với số liệu sẵn có, nên việc mở rộng tin nhắn được xác định.

Mở rộng thông điệp: đầu tiên, khối thông điệp M_t được chia thành 16 từ 32-bit $W_0, \ldots, W_{15}$. Sau đó 16 từ này được mở rộng theo tuyến tính như sau:

$W_i = W_{i-16} \oplus W_{i-14} \oplus W_{i-8} \oplus W_{i-3}$ với $16 \le i \le 79$.

Cập nhật trạng thái: Đầu tiên, biến chuỗi H_t được chia thành 5 từ 32 bit để điền vào 5 thanh ghi (A_0, B_0, C_0, D_0, E_0). Sau đó chuyển đổi tiếp theo được thực hiện 80 lần:

$$\text{STEP}_{i+1} := \begin{cases} A_{i+1} = (A_i \lll 5) + f_i(B_i, C_i, D_i) + E_i + K_i + W_i, \\ B_{i+1} = A_i, \\ C_{i+1} = B_i \ggg 2, \\ D_{i+1} = C_i, \\ E_{i+1} = D_i. \end{cases}$$

Trong đó K_i là các hằng số được xác định trước và f_i là hàm luận lý được định nghĩa trong bảng 1.

Feed-forward: Các tổng modulo 2^{32} : (A_0+A_{80}), (B_0+B_{80}), (C_0+C_{80}), (D_0+D_{80}), (E_0+E_{80}) được nối thành các biến chuỗi H_{t+1}.

Lưu ý rằng tất cả các thanh ghi được cập nhật nhưng thanh ghi A_{i+1} chỉ là những bản sao quay nên chúng ta chỉ cần xem xét thanh ghi A ở mỗi bước. Vì vậy, chúng ta có:

$$A_{i+1} = (A_i \lll 5) + f_i(A_{i-1}, A_{i-2} \ggg 2, A_{i-3} \ggg 2) + A_{i-4} \ggg 2 + K_i + W_i.$$

Round	Step i	$f_i(B,C,D)$	K_i
1	$1 \le i \le 20$	$f_{IF} = (B \wedge C) \oplus (\bar{B} \wedge C)$	0x5a827999

2	21≤i≤40	$f_{XOR} = B \oplus C \oplus D$	0x6ed6eba1
3	41≤i≤60	$f_{MAJ} = (B \wedge C) \oplus (B \wedge D) \oplus (C \wedge D)$	0x8fabbcdc
4	61≤i≤80	$f_{XOR} = B \oplus C \oplus D$	0xca62c1d6

Bảng 1. Các hàm Boolean và hằng số trong SHA-0

2.2.2. SHA-1

2.2.2.1. 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.[5]

SHA-1 khác với SHA-0 chỉ bằng một vòng quay đơn lẻ trong lịch trình thông báo của hàm nén . Theo NSA, điều này đã được thực hiện để sửa một lỗ hổng trong thuật toán ban đầu làm giảm độ an toàn mã hoá của nó, nhưng họ không cung cấp thêm lời giải thích nào.

2.2.4.2. 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 từ 32-bit, (m0, m1, ..., m15). Các từ của tin nhắn lần đầu tiên được mở rộng như sau:

for i = 16, ..., 79,

$m_i = (m_{i-3} \oplus m_{i-8} \oplus m_{i-14} \oplus m_{i-16}) \ll 1$

Các từ tin nhắn được mở rộng sau đó được xử lý trong bốn vòng, mỗi vòng gồm 20 bước. Hàm bước được định nghĩa như sau.

For i = 1, 2, ..., 80,

$a_i = (a_{i-1} \ll 5) + f_i(b_{i-1}, c_{i-1}, d_{i-1}) + e_{i-1} + m_{i-1} + k_i$

$b_i = a_{i-1}$

$c_i = b_{i-1} \ll 30$

$d_i = c_{i-1}$

$e_i = d_{i-1}$

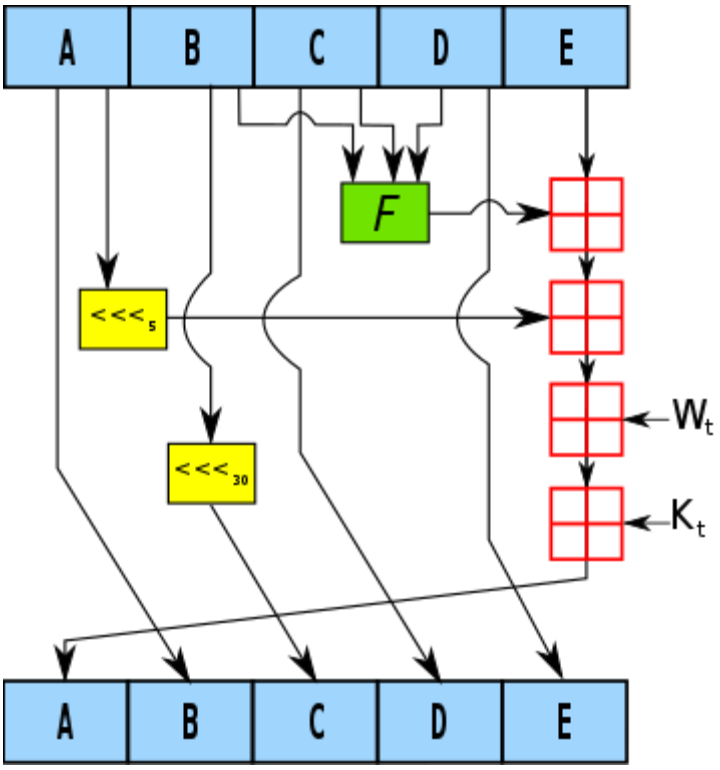
Giá trị chuỗi ban đầu IV = (a_0, b_0, c_0, d_0, e_0) được định nghĩa như sau:
(0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476, 0xc3d2e1f0)

Mỗi vòng sử dụng một hàm Boolean và hằng số k_i khác nhau , được tóm tắt trong bảng 2.

Round	Step i	$f_i(x,y,z)$	K_i
1	$1 \le i \le 20$	$f_{IF} = (x \wedge y) \oplus (\bar{x} \wedge z)$	0x5a827999
2	$21 \le i \le 40$	$f_{XOR} = x \oplus y \oplus z$	0x6ed6eba1
3	$41 \le i \le 60$	$f_{MAJ} = (x \wedge y) \vee (x \wedge z) \vee (y \wedge z)$	0x8fabbcdc
4	$61 \le i \le 80$	$f_{XOR} = x \oplus y \oplus z$	0xca62c1d6

Bảng 2. Các hàm Boolean và hằng số trong SHA-1

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, $\boxplus$ 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

2.2.3. SHA-2

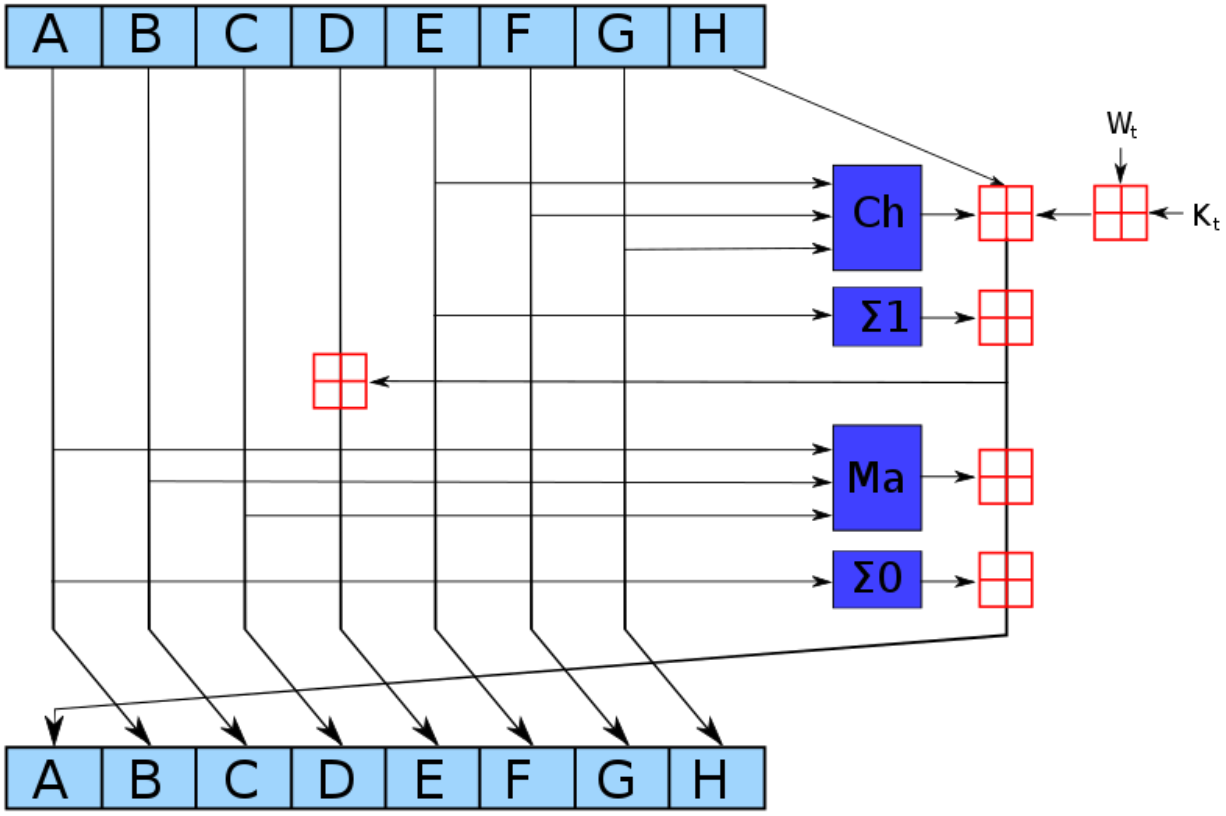
2.2.3.1. Giới thiệu họ SHA-2

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).[6]

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 .

SHA-256 và SHA-512 là những hàm băm mới được tính bằng các từ 32-bit và 64-bit. Chúng sử dụng số lượng thay đổi và hằng số phụ khác nhau, nhưng cấu trúc của là hầu như giống hệt nhau, chỉ khác nhau về số vòng. SHA-224 và SHA-384 chỉ đơn giản là các phiên bản cắt ngắn của hai phiên bản đầu tiên, được tính với các giá trị ban đầu khác nhau. SHA-512/224 và SHA-512/256 cũng là phiên bản rút ngắn của SHA-512, nhưng các giá trị ban đầu được tạo ra bằng cách sử dụng phương pháp được mô tả trong Tiêu chuẩn xử lý thông tin liên bang (FIPS) PUB 180-4. SHA-2 đã được Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST) công bố năm 2001 theo tiêu chuẩn của Mỹ (FIPS). Họ thuật toán SHA-2 được cấp bằng sáng chế trong patent Mỹ 6829355. Hội đồng United States đã phát hành bằng sáng chế theo một giấy phép miễn phí bản quyền.

Mặc dù Gilbert và Handschuh (2003) đã nghiên cứu và không tìm ra điểm yếu của những biến thể này, chúng vẫn chưa được kiểm chứng kĩ như SHA-1.



Hình 2. 2 Lưu đồ một vòng lặp của SHA-2

Hình 2.2 biểu diễn lưu đồ một lần lặp trong hàm nén của họ SHA-2. Các thành phần màu xanh lam thực hiện các hoạt động sau:

Ch(E,F,G)= (E∧F) ⊕ ($\bar{E}$ ∧G)

Ma(A,B,C)=(A∧B) ⊕ (A∧C) ⊕ (B∧C)

$\sum_0(A) = (A \gg 2) \oplus (A \gg 13) \oplus (A \gg 22)$

$\sum_1(E) = (E \gg 6) \oplus (E \gg 11) \oplus (E \gg 25)$

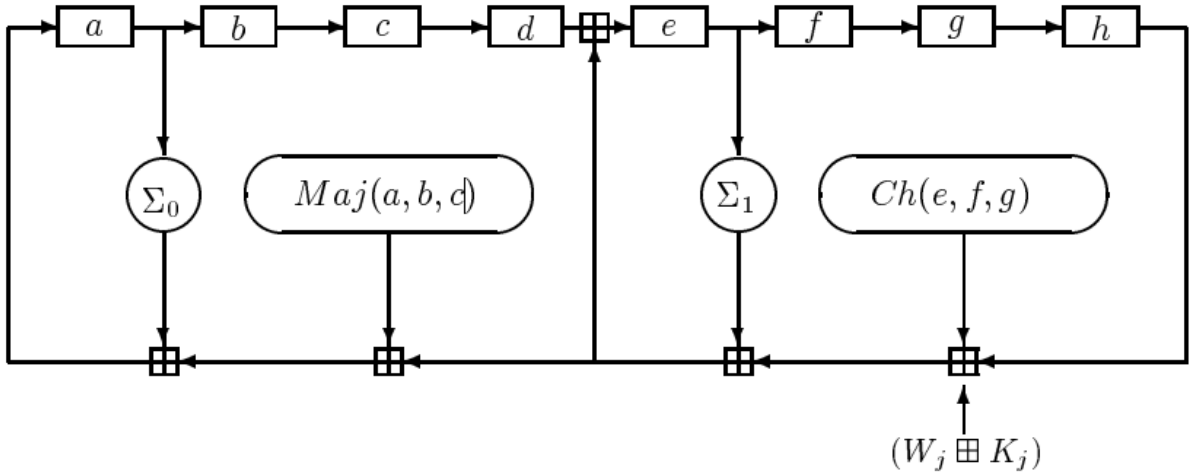
Sự quay bit sử dụng các hằng số khác nhau cho SHA-512. Các số đã cho cho SHA-256. Toán tử ⊞ cộng modulo 2^{32} cho SHA-256 hoặc 2^{64} cho SHA-512.

2.2.3.2.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. Do đó có hai thành phần chính để mô tả: (1) hàm nén SHA-256, và (2) lịch thông báo của SHA-256.[7]

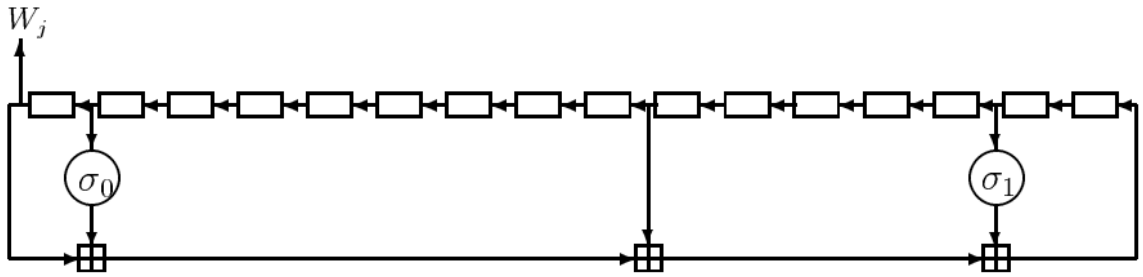
Hàm nén SHA-256 được minh họa dưới đây:



Hình 2. 3 Bước nội bộ thứ j của hàm nén SHA-256 C

Trong đó $\boxplus$ biểu thị cộng mod $2^{32.}$

Lịch trình thông báo có thể được rút ra như sau:



Hình 2. 4 Lịch trình thông điệp SHA-256

Các thanh ghi ở đây được nạp với W0; W1; …; W15:

Lưu ý 1: Tất cả các biến là 32 bit unsigned integers và được tính theo modulo 2^{32}

Lưu ý 2: Với mỗi vòng tròn, một hằng vòng k [i] và một mục trong mảng lịch thông điệp w [i], $0 \leq i \leq 63$

Lưu ý 3: chức năng nén sử dụng 8 biến làm việc, một đường h

Lưu ý 4: Ước Big-endian được sử dụng khi thể hiện các hằng số trong giả này, và khi phân tích dữ liệu khối nhắn từ byte lời, ví dụ, từ đầu tiên của thông tin đầu vào "abc" sau khi đệm là 0x61626380

B1: Khởi tạo các giá trị băm:

(32 bit đầu tiên của các phần phân đoạn của hình vuông gốc 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

B2: Khởi tạo mảng các hằng số tròn:

(32 bit đầu tiên của các phần phân đoạn của các phần tử lập phương của 64 số nguyên tố đầu tiên 2..311):

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

B3: Quá trình xử lý

Bắt đầu với thông điệp ban đầu của độ dài L bit nối thêm một bit '1' vào bit
K '0', trong đó K là số tối thiểu> = 0, ví dụ: rằng L + 1 + K + 64 là một bội số của 512 phụ L như là một số nguyên lớn-64-endian, làm cho tổng chiều dài sau xử lý một bội số của 512 bit

B4: Xử lý thông báo trong các khối 512 bit kế tiếp:

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 (Các giá trị ban đầu trong w [0 63] không thành vấn đề, quá nhiều hiện thực không cho họ ở đây) sao chép đoạn vào 16 từ đầu tiên w [0..15] của mảng lịch thông báo Mở rộng 16 từ đầu tiên vào 48 từ còn lại w [16..63] mảng thông điệp: cho i từ 16 đến 63

```
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
```

B5: Khởi tạo 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
```

B6: Nén vòng lặp chính: cho i từ 0 đến 63

```
for i from 0 to 63
S1 := (e rightrotate 6) xor (e rightrotate 11) xor (e rightrotate 25)
ch := (e and f) xor ((not e) and g)
temp1 := h + S1 + ch + k[i] + w[i]
S0 := (a rightrotate 2) xor (a rightrotate 13) xor (a rightrotate 22)
maj := (a and b) xor (a and c) xor (b and c)
temp2 := S0 + maj
h := g
g := f
f := e
e := d + temp1
```

d := c

c := b

b := a

a := temp1 + temp2

B7: 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

B8: Tạo ra giá trị băm cuối cùng (big-endian):

digest := hash := h0 append h1 append h2 append h3 append h4 append h5 append h6 append h7

SHA-384 giống hệt với SHA-512, ngoại trừ:

giá trị băm ban đầu h0 thông qua h7 là khác nhau (lấy từ số 9 đến số 16), và

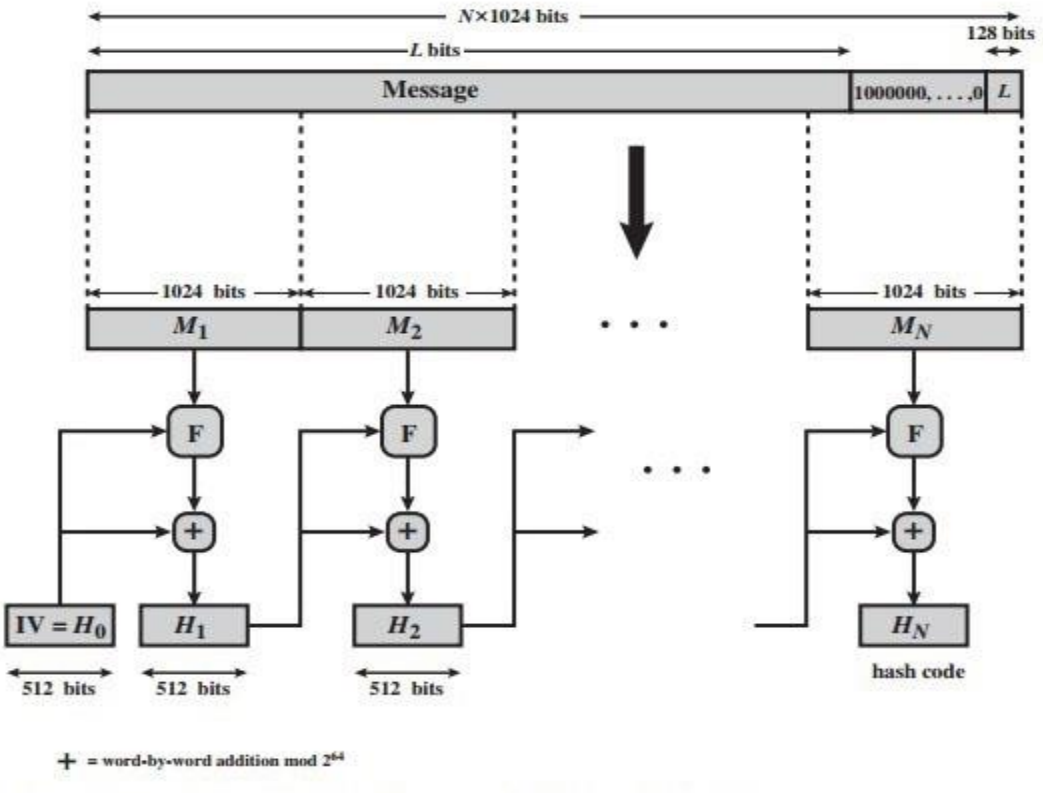
đầu ra được xây dựng bằng cách bỏ qua h6 và h7.

2.2.4 .SHA-3

2.2.4.1. Giới thiệu họ 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. Các XOF khác với hàm băm, nhưng, như đã nêu trong tiêu chuẩn SHA-3, "có thể sử dụng chúng trong những cách tương tự, với sự linh hoạt để được thích nghi trực tiếp với các yêu cầu của các ứng dụng cá nhân, tùy thuộc vào sự cân nhắc về an ninh bổ sung ". Các chức năng SHA-3 dựa trên Keccak được sản xuất bởi G. Bertoni, J. Daemen, M. Peeters, G. Van Assche. Keccak đã được chọn cho mục đích này vì nó đã được tuyên bố vào ngày 2 tháng 10 năm 2012, người chiến thắng trong Cạnh tranh hàm băm NIST do NIST tổ chức [8].

2.2.4.2. Giải thuật SHA3-512



Hình 2. 5 Tạo tin nhắn số trong 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

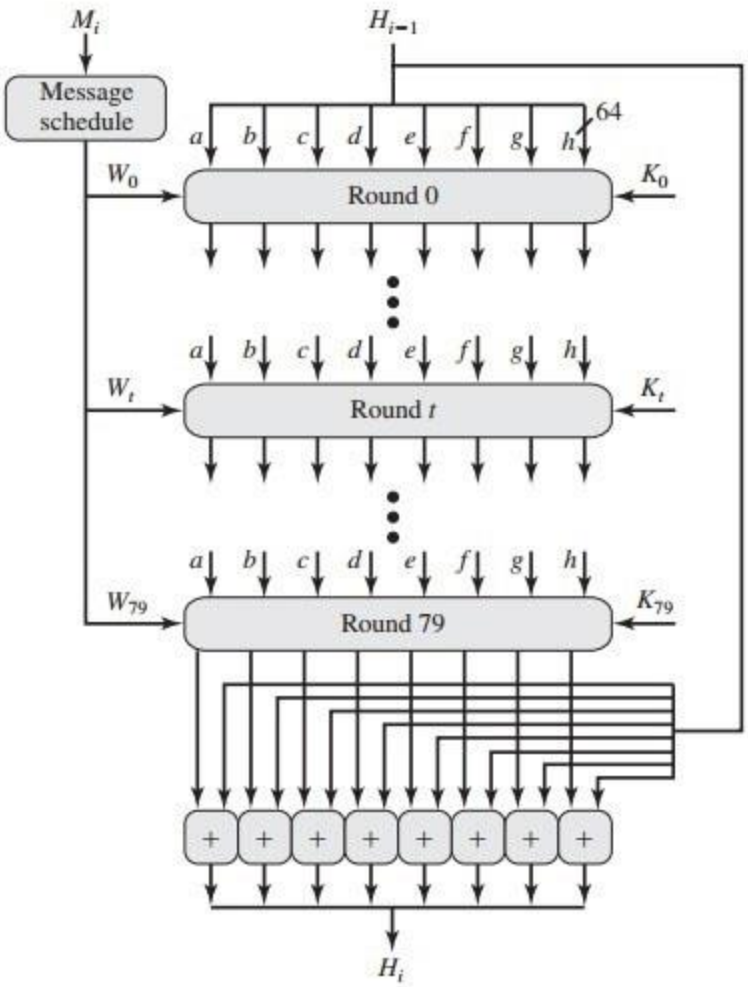
nội dung của bộ đệm.Tại đầu vào vòng đầu tiên, bộ đệm có giá trị

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.



Hình 2. 6 Xử lý SHA3-512 của một khối đơn 1024 bit

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

Chúng ta có thể tóm tắt hành vi của SHA-512 như sau:

$H_0 = IV$

$H_i = SUM_{64}(H_{i=1}, abcdefgh_i)$

$MD = H_N$

IV = giá trị ban đầu của bộ đệm abcdefgh, được định nghĩa trong bước 3

$Abcdefgh_i$ = đầu ra của vòng xử lý cuối cùng của thông điệp thứ khối

N = số khối trong tin nhắn (bao gồm cả padding và chiều dài lĩnh vực)

MD = Giá trị cuối cùng[9]

VD

SHAKE128("The quick brown fox jumps over the lazy dog", 256)

f4202e3c5852f9182a0430fd8144f0a74b95e7417ecae17db0f8cfeed0e3e66e
SHAKE128("The quick brown fox jumps over the lazy dof", 256)
853f4538be0db9621a6cea659a06c1107b1f83f02b13d18297bd39d7411cf10c

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Max message size (bits)	Rounds	Operations	Security bits (Info)	Capacity against length extension attacks	Performance on Skylake (median cpb)[39]		First Published
										long messages	8 bytes	
MD5 (as reference)		128	128 (4 × 32)	512	Unlimited[40]	64	And, Xor, Rot, Add (mod 2^{32}), Or	<64 (collisions found)	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	$2^{64} - 1$	80	And, Xor, Rot, Add (mod 2^{32}), Or	<34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1								<63 (collisions found[41])		3.47	52.00	1995
SHA-2	*SHA-224*	224	256 (8 × 32)	512	$2^{64} - 1$	64	And, Xor, Rot, Add (mod 2^{32}), Or, Shr	112	32	7.62	84.50	2004
	SHA-256	256						128	0	7.63	85.25	2001
	SHA-384	384	512 (8 × 64)	1024	$2^{128} - 1$	80	And, Xor, Rot, Add (mod 2^{64}), Or, Shr	192	128 (≤ 384)	5.12	135.75	
	SHA-512	512						256	0	5.06	135.50	
	SHA-512/224	224						112	288	≈ SHA-384	≈ SHA-384	
	SHA-512/256	256						128	256			
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	Unlimited[42]	24[43]	And, Xor, Rot, Not	112	448	8.12	154.25	2015
	SHA3-256	256		1088				128	512	8.59	155.50	
	SHA3-384	384		832				192	768	11.06	164.00	
	SHA3-512	512		576				256	1024	15.88	164.00	
	SHAKE128	*d* (arbitrary)		1344				min(*d*/2, 128)	256	7.08	155.25	
	SHAKE256	*d* (arbitrary)		1088				min(*d*/2, 256)	512	8.59	155.50	

Hình 2. 7 So sánh các hàm băm

Hình 2.7 so sánh các hàm băm MD5, SHA-0, SHA-1, họ SHA-2, họ SHA-3 trên các chỉ tiêu.

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

3.1. 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 bít bằng nhau và các va chạm hoàn toàn của SHA-0 giảm xống 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.

3.2.Các dạng tấn công vào SHA

3.2.1. Tấn công va chạm

Về mặt toán học, một cuộc tấn công va chạm tìm thấy hai thông điệp khác nhau m_1 và m_2, như vậy hash (m_1) = hash (m_2). Trong một cuộc tấn công va chạm cổ điển, kẻ tấn công không kiểm soát nội dung của một trong hai thông báo, nhưng chúng được tùy ý chọn theo thuật toán.

Giống như mật mã khóa đối xứng là dễ bị tấn công bằng vũ lực, mọi hàm băm mật mã vốn có thể dễ bị va chạm sử dụng một cuộc tấn công sinh nhật. Do vấn đề sinh nhật, những cuộc tấn công này nhanh hơn nhiều so với một lực lượng tàn bạo. Một băm của n bit có thể bị hỏng trong $2^{n/2}$ thời gian (đánh giá của hàm băm).

Có thể thực hiện các cuộc tấn công hiệu quả hơn 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 tìm thấy là nhanh hơn một cuộc tấn công sinh nhật, một hàm băm thường bị tố cáo là "bị hỏng". Sự cạnh tranh hàm băm NIST phần lớn gây ra bởi các cuộc tấn công va chạm được công bố chống lại hai hàm băm được sử dụng rất phổ biến, MD5[10] và SHA-1.

Các cuộc tấn công va chạm chống lại MD5 đã được cải thiện rất nhiều, kể từ năm 2007, chỉ mất vài giây trên một máy tính thông thường.[11] Va chạm Hash tạo ra theo cách này thường có chiều dài không đổi và phần lớn không có cấu trúc, vì vậy không thể trực tiếp được áp dụng để tấn công các định dạng tài liệu phổ biến hoặc các giao thức.

Tuy nhiên, cách giải quyết có thể bằng cách lạm dụng các cấu trúc động có trong nhiều định dạng. Bằng cách này, sẽ tạo ra hai tài liệu tương tự như có thể để có cùng một giá trị băm. Một tài liệu sẽ được hiển thị cho một cơ quan để được ký kết, và sau đó chữ ký có thể được sao chép vào tập tin khác. Một tài liệu độc hại như vậy sẽ chứa hai thông điệp khác nhau trong cùng tài liệu, nhưng điều kiện hiển thị một hoặc nhiều thông tin khác qua các thay đổi tinh tế đối với tệp:

- Một số định dạng tài liệu như PostScript, hoặc macro trong Microsoft Word, có cấu trúc có điều kiện[12] (if-then-else) cho phép kiểm tra xem vị trí trong tệp tin có giá trị này hay vị trí khác để kiểm soát những gì được hiển thị.
- TIFF có thể chứa các hình ảnh được cắt, với một phần khác nhau của một hình ảnh được hiển thị mà không ảnh hưởng đến giá trị băm.[12]
- Các tệp PDF dễ bị tấn công va chạm bằng cách sử dụng giá trị màu (văn bản của một thư được hiển thị với màu trắng trộn vào nền và văn bản của thư khác được hiển thị với màu tối) sau đó có thể thay đổi để thay đổi nội dung của văn bản có chữ ký. [12]

Sự mở rộng của cuộc tấn công va chạm là cuộc tấn công va chạm tiền tố được chọn, cụ thể đối với các hàm băm Merkle-Damgård. Trong trường hợp này, kẻ tấn công có thể chọn hai tài liệu tùy tiện khác nhau, và sau đó nối các giá trị tính khác nhau dẫn đến toàn bộ tài liệu có một giá trị băm bằng nhau. Cuộc tấn công này mạnh mẽ hơn nhiều so với một vụ va chạm cổ điển.

Toán học cho biết, đưa ra hai tiền tố khác nhau p_1, p_2, cuộc tấn công tìm thấy hai phụ m_1 và m_2 như hash ($p_1 \parallel m_2$) = hash ($p_1 \parallel m_2$) (trong đó $\parallel$ là phép nối).

Trong năm 2007, một cuộc tấn công va chạm tiền tố đã được tìm thấy đối với MD5, đòi hỏi khoảng 250 đánh giá chức năng MD5. Bài báo cũng chứng tỏ hai chứng chỉ X.509 cho các tên miền khác nhau, với các giá trị băm va chạm. Điều này có nghĩa là một cơ quan chứng chỉ có thể được yêu cầu ký một chứng chỉ cho một tên miền và sau đó chứng chỉ đó có thể được sử dụng để giả mạo một tên miền khác.

Một cuộc tấn công va chạm thực sự đã được xuất bản vào tháng 12 năm 2008 khi một nhóm các nhà nghiên cứu bảo mật xuất bản một chứng chỉ ký kết X.509 giả mạo có thể được sử dụng để giả mạo một cơ quan chứng nhận, lợi dụng một cuộc tấn công va chạm tiền tố chống lại chức năng băm MD5. Điều này có nghĩa là kẻ tấn công có thể giả mạo bất kỳ trang web được bảo mật SSL nào như người đàn ông ở giữa, do đó làm mất đi sự chứng thực của chứng chỉ được xây dựng trong mọi trình duyệt web để bảo vệ thương mại điện tử. Chứng chỉ giả mạo có thể không được các cơ quan thực sự có thể tiết lộ và cũng có thể có thời gian hết hạn giả mạo tùy ý. Mặc dù MD5 đã rất yếu trong năm 2004, các cơ quan

chứng nhận vẫn sẵn sàng ký chứng chỉ đã được kiểm chứng MD5 vào tháng 12 năm 2008, và ít nhất một chứng chỉ ký kết mã của Microsoft vẫn sử dụng MD5 vào tháng 5 năm 2012.

Phần mềm độc hại Flame đã sử dụng thành công một biến thể mới của cuộc tấn công va chạm tiền tố được lựa chọn để giả mạo việc ký kết các thành phần của nó bằng một chứng chỉ gốc của Microsoft vẫn sử dụng thuật toán MD5 bị xâm nhập.

3.2. 2.Tấn công hàm Hash theo kiểu ngày sinh nhật

Nếu kích thước của hàm Hash nhỏ, thì có thể tìm được 2 văn bản có cùng giá trị hàm băm, tức là có va chạm mà không phụ thuộc vào số lượng biến đổi của hàm, cách tấn công này có tên ngày sinh nhật. Ý tưởng của phương pháp tấn dựa trên bài toán ngày sinh nhật sau. Cần phải chọn một nhóm bao nhiêu người để xác suất hai người có cùng ngày sinh nhật là 0.5? Vấn đề ở chổ là xác suất trùng ngày sinh nhật đối với một cặp ngẫu nhiên là p'=1/365, còn trong nhóm gồm n người có $n(n-1)/2 \approx n^2$ cặp khác nhau. Từ đây dễ dàng nhận được đánh giá gần đúng. Xác suất để tồn tại ít nhất một cặp có cùng ngày sinh là $p = p'n^2/2$, từ đây p=1/2 thì chúng ta thu được $n \approx \sqrt{1/p'}$.

Chúng ta xem sử dụng bài toán ngày sinh nhật để tìm va chạm trong hàm Hash như thế nào. Giả sử cho H là hàm Hash với kích thước đầu ra là m bít. Chúng ta có N bản tin khác nhau $M^{(i)}, i = 1...N$, tính toán giá trị băm của các bản tin này, giá trị tương ứng là $H^{(i)}$. Nếu như hàm Hash là hàm biến đổi giả ngẫu nhiên thì dễ dàng tính được xác suất sao cho giữa N giá trị $H^{(i)}$ không tìm được hai như nhau.

Đầu tiên chúng ta xem đánh giá dựa trên tính toán gần đúng, sao cho số lượng va chạm là đủ nhỏ. Chúng ta chọn một số giá trị $H^{(1)}$. Xác suất để nó không trùng với các phần còn lại là

$P^{(1)} \approx (1-2^{-m})^{N-1}$.

Tiếp tục chọn một số giá trị mới $H^{(2)}$. Xác suất để nó không trùng với phần còn lại là

$P^{(2)} \approx (1-2^{-m})^{N-2}$.

Chúng ta chọn tương tự như vậy đối với giá trị mới của hàm Hash, trong bước thứ I chúng ta thu được xác suất không trùng là

$P^{(i)} \approx (1-2^{-m})^{N-i}$

Tất cả chúng ta cần thực hiện N-1 bước kiểm tra không trùng. Xác suất để không một giá trị trong chúng không trùng là:

$P' \approx (1-2^{-m})^{N-1} \cdot (1-2^{-m})^{N-2} \cdot ... \cdot (1-2^{-m})^{N-i} \cdot ... \cdot (1-2^{-m}) = (1-2^{-m})^{\sum}$

Với $\sum = 1+2+...+(N-1) = N(N-1)/2$.

Xác suất tìm thấy va chạm là

$p = 1-p' = 1-(1-2^{-m})^{\sum} \approx 1-(1-\sum \cdot 2^{-m}) \approx N^2 \cdot 2^{-m-1}$.

Với xác suất va chạm là ½ thì ta có biểu thức:

$N^2 \cdot 2^{-m-1} = 1/2$,

Từ đây chúng ta xác định gía trị $N_{1/2}$:

$N_{1/2} \approx \sqrt{2^m}$.

Chúng ta xem tính cách tính chính xác hơn xác suất tìm va chạm trong tập hợp $H^{(i)}, i = 1...N$, có thể nhận được bằng cách rất đơn giản sau. Chúng ta chọn $H^{(2)}$. Xác suất để $H^{(2)}$ không trùng $H^{(1)}$ là $p^{(2)} = (1 - 2^{-m})$. Tiếp tục chọn $H^{(3)}$. Xác suất để $H^{(3)}$ không trùng với $H^{(2)}$ và $H^{(1)}$, với điều kiện $H^{(1)}$ và $H^{(2)}$ không trùng nhau là $p^{(3)} = (1 - 2^{-m})(1 - 2 \cdot 2^{-m})$. Một cách tương tự, chúng ta xác định xác suất $p^{(N)}$ để $H^{(N)}$ không trùng với một trong các giá trị $H^{(1)}, H^{(2)}, \ldots, H^{(N-1)}$, với điều kiện là các giá trị $H^{(1)}, H^{(2)}, \ldots, H^{(N-1)}$ khác nhau từng đôi một. Chúng ta nhận được $p^{(N)} = p^{(2)} \cdot p^{(3)} \cdot ... \cdot p^{(N-1)} \cdot [1 - (N-1) \cdot 2^{-m}]$. Như vậy giá trị chính xác của xác suất không có sự va chạm là:

$$p' = p^{(N)} = (1 - 2^{-m}) \cdot (1 - 2 \cdot 2^{-m}) \cdot ... \cdot [1 - (i-1) \cdot 2^{-m}] \cdot ... \cdot [1 - (N-1) \cdot 2^{-m}] = \prod_{i=1}^{N-1} (1 - i \cdot 2^{-m}) .$$

Từ đây chúng ta xác định xác suất có sự va chạm là p=1-p'. Áp dụng công thức gần đúng $1 - x \approx e^{-x}$. Chúng ta thu được:

$$p' = \prod_{i=1}^{N-1} (1 - i \cdot 2^{-m}) \approx \prod_{i=1}^{N-1} \exp(-i \cdot 2^{-m}) = \exp(-\sum_{i=1}^{N-1} i \cdot 2^{-m}) = \exp[-N(N-1) \cdot 2^{-m-1}] .$$

Xác suất tồn tại ít nhất một va chạm là:

$p \approx 1 - p' = 1 - \exp[-N(N-1) \cdot 2^{-m-1}]$.

Từ đây ta dể dạng nhận được điều sau:

$$N^2 + N \approx 2^{m+1} \ln\left(\frac{1}{1-p}\right),$$

Hay

$$N^2 \approx 2^{m+1} \ln\left(\frac{1}{1-p}\right),$$

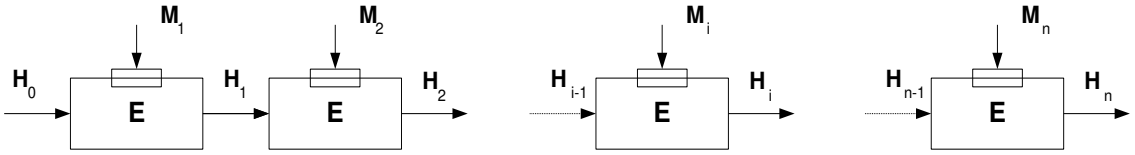
$$N \approx \sqrt{2^{m+1} \ln\left(\frac{1}{1-p}\right)}.$$

Với p=1/2 chúng ta có $N_{1/2} \approx 1.17\sqrt{2^m}$. Chúng ta thấy kết quả ở phương án tính này chính xác hơn phương án đầu tiên. Và từ công thức này chúng ta thấy, trong số 23 người chọn ngẫu nhiên thì có ít nhất một cặp trùng ngày sinh với xác suất là ½. Như vậy để thực hiện tấn công thì cần bộ nhớ là $1.17 \cdot 2^{m/2} m$ bít và cần thực hiện $1.17 \cdot 2^{m/2}$ tính toán hàm

Hash và thực hiện sắp xếp $1.17 \cdot 2^{m/2}$ số. Và từ đây chúng thấy rằng nếu như m không đủ lớn thì sẽ dễ dàng tìm ra được số lượng bản tin mà có sự va chạm. Với công nghệ hiện nay thì đòi hỏi $m \geq 128$ bít.

3.2.3. Tấn công hàm hash theo kiểu gặp nhau ở giữa (meet – in – the – middle attack)

Phương pháp tấn công gặp nhau ở giữa áp dụng cho các hàm Hash xây dựng trên cơ sở mã khối, mà chúng ta tìm hiểu ở phần trước. Phương pháp này cho kết quả tốt hơn phương pháp tấn công theo ngày sinh nhật. Trong tấn công theo kiểu ngày sinh nhật tìm được va chạm nhưng giá trị nhận được của hàm Hash đối với tìm kiếm va chạm là ngẫu nhiên. Tấn công đầu tiên được đề xuất là tấn công trên hàm Hash xây dựng trên cơ sở sơ đồ Rabin xem hình 3.1.



Hình 3. 1 Sơ đồ tạo hàm Hash Rabin

Sơ đồ này dựa trên thuật toán mã khối an toàn. Sơ đồ dựa trên ý tưởng về tính toán phức tạp xác định khóa khi biết đầu vào và đầu ra của khối dữ liệu. Khối dữ liệu Mi được sử dụng như khóa tương ứng với một vòng tính toán của hàm Hash. Tìm kiếm va chạm liên quan đến bài toán tính toán khóa. Ví dụ, tấn công có thể thay thế một số khối M_k thành M'_k. Điều này dẫn đến nhận được giá trị mới của vòng hàm hash H'_k. Có thể tồn tại một số khóa mã M'_{k+1}, mà chúng ta nhận được đẳng thức sau:

$H'_{k+1} = E_{M'_{k+1}}(H'_k) = H_{k+1}$.

Nếu như cách thám mã tìm được M'_{k+1} đã cho thì thay thế khối dữ liệu M_k và M_{k+1} thành M'_k và M'_{k+1}, tức là ta đã tìm được bản tin mới, mà có giá trị hàm Hash bằng với giá trị hàm Hash của bản tin ban đầu. Nếu như thuật toán mã khối là vững chắc đối với phép tấn công trên cơ sở biết bản tin, thì phép tấn công đã cho trên hàm Hash có độ tính phức tạp cao.

Chúng ta xem cụ thể phép tấn công này. Giả sử cho bản tin M, giá trị hàm Hash của M là H. Mục đích của phép tấn công là tìm ra bản tin khác M' mà gía trị hàm Hash của M' cũng bằng H. Chúng ta chia bản tin $M = (M_1, M_2, ..., M_k, M_{k+1}, ..., M_n)$ thành hai phần $M^{(1)} = (M_1, M_2, ..., M_k)$ và $M^{(2)} = (M_{k+1}, ..., M_n)$. Phần đầu tiên của bản tin được biến dạng nhiều lần và mỗi sự biến dạng đó giá trị hàm Hash được tính bằng $H^{(1)}$. Giả sử nhận được N_1 gía trị $H^{(1)}$ từ N_1 phương án của phần thứ nhất (xem hình …). Phần thứ hai của bản tin $M^{(2)}$ cũng biến dạng nhiều lần, từ mỗi biến dạng đó hàm Hash được tính theo thuật toán

khác, ở đây chúng ta tính theo thứ tự ngược và sử dụng hàm giải mã D, tương ứng vơi hàm mã hóa E (xem hình…). Giả sử thu được N_2 giá trị $H^{(2)}$ từ N_2 phương án của phần hai.

Khi số lượng N_1 và N_2 đủ lớn thì có thể tìm được cặp giá trị bằng nhau trong số $H^{(1)}$ và $H^{(2)}$ vớ xác suất lớn. Giả sử rằng nó tương ứng với hai bản tin là $M'^{(1)}$ và $M'^{(2)}$. Rõ ràng rằng bản tin $M'=(M'^{(1)},M'^{(2)})\neq M$ mà H(M)=H(M'). Vậy đã tìm ra được sự va chạm. Giờ chúng ta xác định xem cần bộ nhớ và độ khó của phương pháp tấn công này.

Chúng ta giả sử rằng N_1 và N_2 tồn tại giá trị nhỏ hơn 2^m, m là kích thước của giá trị băm. Như thế, với giá trị xác suất gần đúng đủ chính xác có thể tiếp nhận, sao cho giá trị đầu tiên $H^{(1)}$ từ tập $\{H^{(1)}\}$ không trùng với một giá trị nào của tập $\{H^{(2)}\}$ bằng

$p_1\approx(1-2^{-m})^{N_2}$.

Xác suất để không một giá trị nào của tập $\{H^{(1)}\}$trùng với một giá trị của tập $\{H^{(2)}\}$ bằng

$p'\approx(1-2^{-m})^{N_2N_1}$

Như vậy xác suất để tìm ra ít nhất một cặp trùng nhau giữa tập $\{H^{(1)}\}$ và tập $\{H^{(2)}\}$ là

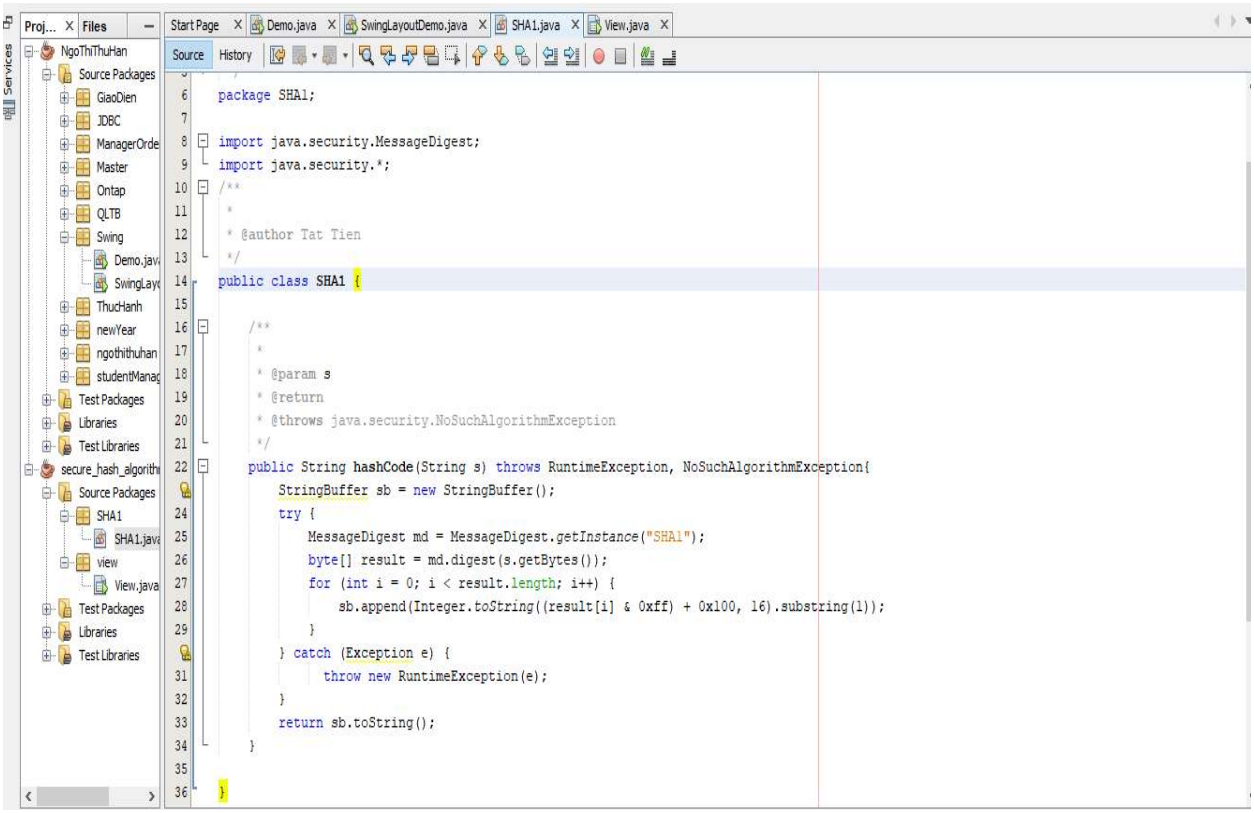
$p\approx 1-p'=1-(1-2^{-m})^{N_1N_2}\approx 1-(1-N_1N_2\cdot 2^{-m})\approx N_1N_2\cdot 2^{-m}$.

Bây giờ với điều kiện $N_1N_2\cdot 2^{-m}=1/2$ đốiv với trường hợp $N_1=N_2=N$ dễ dàng xác định giá trị $N_{1/2}$, với xác suất va chạm là ½:

$N_{1/2}\approx\sqrt{2^{m-1}}$.

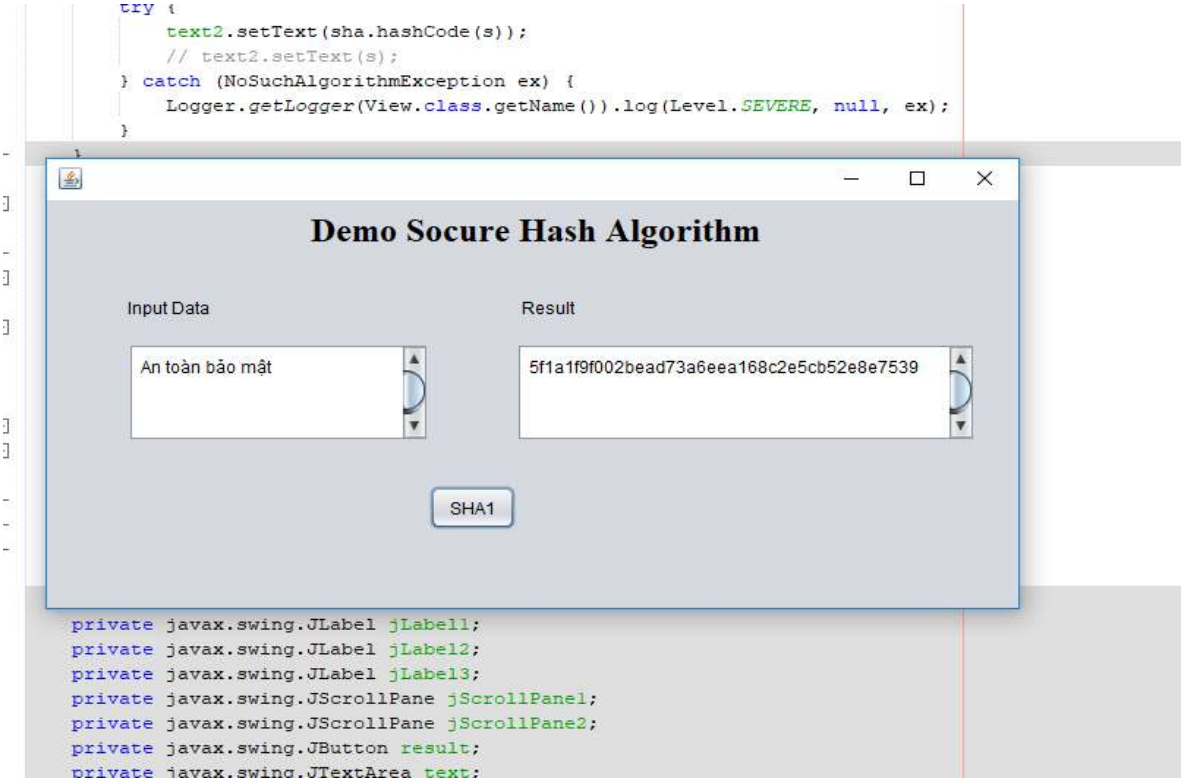
Để nhận được đánh giá chính xác hơn khi $N_1=N_2=2^{-m/2}$ có thể nhận gía trị p=0.63. Như vậy cần bộ nhớ cần thiết cho phép tấn công là $2m\sqrt{2^{m-1}}$ bít, độ khó tấn công là $N_1+N_2\approx\sqrt{2^{m+1}}$.

Chương 4. Demo



Hình 4. 1 Chương trình SHA-1

Hình 4.1 Thể hiện một đoạn chương trình sử dụng thuật toán SHA-1 trong java để cài đặt thử nghiệm.



Hình 4. 2 Demo SHA-1

Hình 4. 3 SHA-1 online

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.

Để đảm bảo an toàn Internet, đã đến lúc loại bỏ SHA-1 và sử dụng các hàm băm mật mã an toàn hơn như SHA-256 và SHA-3.

Tài liệu tham khảo

[1] H. X. Dậu, “An toàn toàn bảo mật hệ thố ng thông tin,” 2017.

[2] Wikipedia, “SHA – Wikipedia tiếng Việt.” [Online]. Available: https://vi.wikipedia.org/wiki/SHA. [Accessed: 13-Mar-2018].

[3] B. Schneier, “Cryptanalysis of SHA-1 - Schneier on Security.” [Online]. Available: https://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html. [Accessed: 13-Mar-2018].

[4] S. Manuel and T. Peyrin, “Collisions on SHA-0 in one hour,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 5086 LNCS, pp. 16–35, 2008.

[5] X. Wang, Y. L. Yin, and H. Yu, “Finding Collisions in the Full SHA-1,” no. 90304009, pp. 17–36, 2005.

[6] W. Penard and T. van Werkhoven, “On the Secure Hash Algorithm family,” p. 17, 2008.

[7] N. H. Function, “Description of SHA-256, SHA-384 AND SHA-512,” *Www.Iwar.Org.Uk/Comsec/Resources/Cipher/Sha256-384-512.Pdf*, pp. 1–50, 2004.

[8] J. Luis, G. Pardo, and C. Gómez-rodríguez, “The SHA-3 Family of Cryptographic Hash Functions and Extendable-Output Functions,” 2015.

[9] W. Stallings, *Cryptography and network security principles and practice*, 5th ed., vol. 139, no. 3. 2011.

[10] Xiaoyun Wang and Dengguo Feng and Xuejia Lai and Hongbo Yu, “Cryptology ePrint Archive: Report 2004/199,” 2004. [Online]. Available: https://eprint.iacr.org/2004/199. [Accessed: 13-Mar-2018].

[11] M. Stevens, “On collisions for MD5,” *Http://Www.Win.Tue.Nl/*, no. June, p. 89, 2007.

[12] M. Gebhardt, G. Illies, and W. Schindler, “A Note on the Practical Value of Single Hash Collisions for Special File Formats,” *October*, no. October, 2005.