

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**BÁO CÁO**

**THỰC TẬP CƠ SỞ**

***Đề tài:***

**TÌM HIỂU THUẬT TOÁN MÃ HÓA MD5 VÀ**

**ỨNG DỤNG NGÔN NGỮ PHP ĐỂ CÀI ĐẶT THEO MÔ HÌNH MVC**

**Sinh viên thực hiện: LY A SÚ**

**Lớp: CNTT – K14E**

**Giáo viên hướng dẫn:ThS. NGUYỄN QUANG HIỆP**

**Thái Nguyên, tháng 07 năm 2018**

# MỤC LỤC

[MỤC LỤC 1](#_Toc519594781)

[DANH MỤC HÌNH VẼ: 3](#_Toc519594782)

[PHẦN I CƠ SỞ LÝ THUYẾT 4](#_Toc519594783)

[1. GIỚI THIỆU VỀ NGÔN NGỮ CÀI ĐẶT 4](#_Toc519594784)

[1.1. KHÁI NIỆM 4](#_Toc519594785)

[1.2. ỨNG DỤNG 4](#_Toc519594786)

[2. GIỚI THIỆU VỀ ĐỀ TÀI 4](#_Toc519594787)

[2.1. MỤC ĐÍCH 4](#_Toc519594788)

[2.2. ĐỐI TƯỢNG NGHIÊN CỨU 4](#_Toc519594789)

[2.3. PHẠM VI NGHIÊN CỨU 5](#_Toc519594790)

[PHẦN II. NỘI DUNG 6](#_Toc519594792)

[CHƯƠNG 1 :](#_Toc519594796) [TỔNG QUAN VỀ HÀM BĂM VÀ THUẬT TOÁN HÀM BĂM MD5 10](#_Toc519594797)

[1.1. GIỚI THIỆU VỀ HÀM BĂM MẬT MÃ 10](#_Toc519594798)

[1.1.1. Tính chất 10](#_Toc519594799)

[1.1.2 Ứng dụng 12](#_Toc519594800)

[1.2 CẤU TRÚC MERKLE-DAMGARD 13](#_Toc519594801)

[1.3 BIRTHDAY ATTACK 13](#_Toc519594802)

[1.4 HÀM BĂM MẬT MÃ 14](#_Toc519594803)

[1.5 CẤU TRÚC HÀM BĂM 15](#_Toc519594804)

[1.6 TÍNH AN TOÀN HÀM BĂM VÀ HIỆN TƯỢNG ĐỤNG ĐỘ 16](#_Toc519594805)

[1.7 TÍNH MỘT CHIỀU 16](#_Toc519594806)

[1.8 GHÉP CÁC HÀM BĂM MẬT MÃ 16](#_Toc519594807)

[1.9 THUẬT TOÁN BĂM MẬT MÃ 17](#_Toc519594808)

[1.10 PHƯƠNG PHÁP SECURE HASH STANDARD (SHS) 18](#_Toc519594809)

[CHƯƠNG 2 : HÀM BĂM MD5 18](#_Toc519594810)

[2.1 GIỚI THIỆU 18](#_Toc519594811)

[2.2 KHÁI NIỆM 18](#_Toc519594812)

[2.3 ỨNG DỤNG 18](#_Toc519594813)

[2.4 GIẢI THUẬT 19](#_Toc519594814)

[2.5 MD5 (Message Digest) 20](#_Toc519594815)

[CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG THUẬT 26](#_Toc519594816)

[3.1 NHIỆM VỤ CỦA CHƯƠNG TRÌNH 26](#_Toc519594817)

[3.2 THUẬT TOÁN VÀ SƠ ĐỒ KHỐI 26](#_Toc519594818)

[3.2.1 Thuật toán: 26](#_Toc519594819)

[3.2.2. Sơ đồ khối thuật toán MD5 30](#_Toc519594820)

[3.3 KẾT QUẢ CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN BĂM MD5 33](#_Toc519594821)

[3.3.1. Giao diện chương trình mô phỏng 33](#_Toc519594822)

[3.3.2 Các bước thực hiện chương trình 33](#_Toc519594823)

[3.3.3 Các kết quả thử nghiệm 35](#_Toc519594824)

[KẾT LUẬN 37](#_Toc519594825)

[TÀI LIỆU THAM KHẢO 38](#_Toc519594826)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN 39](#_Toc519594827)

# DANH MỤC HÌNH VẼ:

[Hình 1.1: Cấu trúc băm Merkle-Damgård. 9](#_Toc519594324)

[Hình 1.2: Sơ đồ vòng lặp chính của MD5 18](file:///D:\download%20may%201\báo-cáo-thực-tập-cơ-sở.docx#_Toc519594325)

[Hình 1.3: Sơ đồ 1 vòng lặp của MD5 18](file:///D:\download%20may%201\báo-cáo-thực-tập-cơ-sở.docx#_Toc519594326)

# PHẦN I CƠ SỞ LÝ THUYẾT

# 1. GIỚI THIỆU VỀ NGÔN NGỮ CÀI ĐẶT

# 1.1. KHÁI NIỆM

PHP là gì? PHP chính là từ viết tắt của Hypertext Preprocessor) là một loại ngôn ngữ lập trình kịch bản hay một loại mã lệnh được dùng chủ yếu với mục đích phát triển các ứng dụng viết cho máy chủ, mở mã nguồn. PHP hiện nay rất phù hợp với các web bởi nó có tốc độ nhanh, nhỏ gọn và cú pháp giống với ngôn ngữ lập trình C và Java.

Hơn thế nữa, loại ngôn ngữ này còn khá dễ học và có thời gian xây dựng sản phẩm tương đối ngắn so với các ngôn ngữ khác. Vì thế, PHP nhanh chóng trở thành ngôn ngữ lập trình web phổ biến và được ưa chuộng sử dụng hàng đầu thế giới. Cũng từ đây, thuật ngữ PHP là gì cũng được tìm kiếm nhiều hơn.

# 1.2. ỨNG DỤNG

Ngày nay PHP được sử dụng với các mục đích sau:

* Phát triển ứng dụng phầm mền cho các thiết bị điện tử thông minh, các ứng dụng cho doanh nghiệp với quy mô lớn.
* Tạo các trang web có nội dung động (web applet), nâng cao chức năng của server.
* Phát triển nhiều loại ứng dụng khác nhau: Cơ sở dữ liệu, mạng, Internet, viễn thông, giải trí,...

# 2. GIỚI THIỆU VỀ ĐỀ TÀI

# 2.1. MỤC ĐÍCH

* Tìm hiểu hàm băm, đi vào thuật toán MD5

Hàm băm (Hash Function): Cho đầu vào là một thông báo có kích thước thay đổi, đầu ra là một mã băm có kích thước cố định.

Giải thuật băm MD5 (Message Digest 5): được sử dụng để kiểm tra tính toàn vẹn của khối dữ liệu lớn. Thuật toán nhận đầu vào là một đoạn tin có chiều dài bất kỳ, băm nó thành các khối 512 bit và tạo đầu ra là một đoạn tin 128 bit.

* Xây dựng chương trình mô phỏng hàm băm MD5

Đưa ra kết quả chương trình mô phỏng và các kết quả thu được đúng với các tài liệu tiêu chuẩn về thuật toán MD5

# 2.2. ĐỐI TƯỢNG NGHIÊN CỨU

- các phương pháp mật mã hóa

- Thuật toán hàm băm MD5

# 2.3. PHẠM VI NGHIÊN CỨU

Bài báo cáo này tập trung nghiên cứu các thuật toán băm MD5 và ứng dụng ngôn ngữ php để cài đặt

# PHẦN II. NỘI DUNG

# CHƯƠNG 1 :

# TỔNG QUAN VỀ HÀM BĂM VÀ THUẬT TOÁN HÀM BĂM MD5

# 1.1. GIỚI THIỆU VỀ HÀM BĂM MẬT MÃ

Hàm băm mật mã là một thủ tục tất định có đầu vào là khối dữ liệu bất kỳ và trả về một xâu bit có độ dài cố định, gọi là giá trị băm (mật mã), mà bất kỳ sự thay đổi vô tình hay các ý trên dữ liệu sẽ thay đổi giá trị băm. Dữ liệu đem mã hóa thường được gọi là thông điệp (message), và giá trị băm đôi khi còn được gọi là tóm lược thông điệp (message digest) hay giá trị tóm lược (digest).

Hàm băm mật mã lý tưởng có 4 tính chất chính sau:

• dễ dàng tính giá trị băm với bất kỳ thông điệp cho trước nào,

• không thể tìm được một thông điệp từ một giá trị băm cho trước,

• không thể sửa được một thông điệp mà không làm thay đổi giá trị băm của nó,

• không thể tìm ra 2 thông điệp khác nhau mà có cùng giá trị băm.

Hàm băm mật mã có rất nhiều ứng dụng trong an toàn thông tin, nhất là cho chữ ký điện tử (Digital Signatures), mã xác thực thông điệp (MACs – Message Authentication Codes), và một số dạng xác thực khác. Chúng cũng có thể sử dụng như các hàm băm thông thường, để đánh chỉ số dữ liệu trong bảng băm: như điểm chỉ, để nhận diện dữ liệu lặp hay xác định tệp dữ liệu duy nhất: hay như checksums để nhận biết sự thay đổi dữ liệu. Thật vậy, trong lĩnh vực an toàn thông tin, giá trị băm mật mã đôi khi được gọi điểm chỉ (số), checksums, hay giá trị băm, dù rằng tất cả các thuật ngữ này chỉ đại diện về mặt chức năng nhưng các tính

## 1.1.1. Tính chất

Hầu hết các hàm băm mật mã được thiết kế với đầu xâu đầu vào có độ dài tùy ý và kết quả đầu ra là một giá trị băm có độ dài cố định. Hàm băm mật mã phải có thể chống lại được tất cả các kiểu tấn công phân tích đã biết. Tối thiểu, nó phải có các tính chất sau:

Kháng tiền ảnh (Preimage resistance): cho trước một giá trị băm h, khó tìm ra thông điệp m thỏa mã h = hash(m). Khái niệm như là hàm một chiều (one way function). Các hàm thiếu tính chất này sẽ bị tổn thương bởi các tấn công tiền ảnh (preimage attacks).

Kháng tiền ảnh thứ 2 (Second preimage resistance): cho trước một đầu vào m1, khó có thể tìm ra đầu vào m2 khác (không bằng m1) thỏa mãn hash(m1) = hash(m2). Tính chất này đôi khi như là kháng va chạm yếu (weak collision resistance). Các hàm thiếu tính chất này sẽ bị tổn thương bởi các tấn công tiền ảnh thứ 2 (second preimage attacks).

Kháng va chạm (Collision resistance): khó có thể tìm ra 2 thông điệp m1 và m2 thỏa mãn hash(m1) = hash(m2). Một cặp như vậy được gọi là một va chạm băm (mật mã), và tính chất này đôi khi như là kháng va chạm mạnh (strong collision resistance). Tính chất này yêu cầu rằng một giá trị băm tối thiểu cũng mạnh hơn yêu cầu kháng tiền ảnh, hơn nữa các va chạm có thể tìm được bởi tấn công ngày sinh (birthday attack).

Các tính chất trên nói lên rằng đối phương ác ý không thể thay hoặc sửa dữ liệu đầu vào mà không làm thay đổi giá trị tóm lược của nó. Do đó, nếu 2 xâu có cùng một giá trị tóm lược, thì người ta tin tưởng rằng chúng là giống nhau.

Một hàm có các tiêu chí này vẫn có thể có các tính chất không mong muốn. Hiện tại các hàm băm mật mã thông thường vẫn bị tổn thương bởi các tấn công mở rộng độ dài (length-extension attacks): cho trước h(m) và len(m) nhưng không biết m, bằng cách chọn m’ hợp lý, kẻ tấn công có thể tính h(m || m’), với || ký hiệu là phép ghép xâu. Tính chất này có thể được sử dụng để phá vỡ các lược đồ xác thực đơn giản dựa vào hàm băm. Cấu trúc HMAC (Hash Message Authentication Code) gặp phải các vấn đề như vậy.

Về mặt lý tưởng, người ta có thể muốn các điều kiện mạnh hơn. Kẻ tấn công không thể tìm ra 2 thông điệp có các giá trị tóm lược gần giống nhau; hoặc luận ra bất kỳ thông tin có ích nào về dữ liệu, mà chỉ cho trước giá trị tóm lược. Do đó, hàm băm mật mã phải tiến gần tới hàm ngẫu nhiên (đến mức có thể) mà vẫn là tất định và tính toán hiệu quả.

Thuật toán checksum, như là CRC32 và các CRC (Cyclic Redundancy Check) khác, được thiết kế nhiều yêu cầu yếu hơn, và nói chung không giống như là các hàm băm mật mã. Ví dụ, có một CRC đã được sử dụng kiểm tra tính toàn vẹn trong chuẩn mã WEP (Wired Equivalent Privacy), nhưng đã có một tấn công khai thác tính tuyến tính của checksum.

## 1.1.2 Ứng dụng

Một ứng dụng tiêu biểu của hàm băm mật mã sẽ như sau: Alice đặt ra một bài toán khó cho Bob, và tuyên bố rằng cô ta đã giải được. Bob sẽ phải cố gắng tự thực hiện, nhưng chưa dám chắc rằng Alice không giải sai. Do đó, Alice viết ra lời giải của mình, gắn thêm một giá trị nonce ngẫu nhiên, tính giá trị băm của nó và cho Bob biết giá trị băm đó (giữ bí mật lời giải và giá trị nonce). Bằng cách này, khi Bob tìm ra lời giải của mình vài ngày sau đó, Alice có thể chứng minh rằng cô ta có lời giải sớm hơn bằng cách tiết lộ giá trị nonce cho Bob. (Đây là một ví dụ về một lược đồ cam kết đơn giản trong thực tế, vai trò của Alice và Bob thường sẽ là các chương trình máy tính, và bí mật sẽ là một cái gì đó dễ dàng giả mạo hơn là một bài toán đó theo yêu cầu).

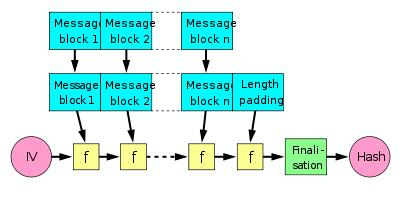
Một ứng dụng quan trọng của băm an toàn là xác minh tính toàn vẹn của thông điệp. Xác định liệu có thay đổi nào đã được thực hiện đối với thông điệp (hoặc một tập tin), ví dụ, có thể được thực hiện bằng cách so sánh các giá trị tóm lược thông điệp đã tính toán trước, và sau đó, truyền đi (hoặc sự kiện nào đó).

Một giá trị tóm lược thông điệp cũng có thể phục vụ như là một phương tiện nhận dạng một tập tin đáng tin cậy một số hệ thống quản lý mã nguồn, bao gồm Git, Mercurial và Monotone, sử dụng giá trị sha1sum của nhiều dạng nội dung khác nhau (nội dung tập tin, cây thư mục, vv) để nhận dạng chúng một cách duy nhất duy nhất.Một ứng dụng khác liên quan tới việc xác thực mật khẩu. Mật khẩu thường không được lưu trữ dạng văn bản rõ, với các lý do hiển nhiên, mà thay bằng dạng giá trị tóm lược. Để xác thực người dùng, mật khẩu đại diện cho người sử dụng được băm và so sánh với giá trị băm lưu trữ. Điều này đôi khi được gọi là phép mã hóa một chiều (one-way encryption). Đối với cả hai lý do bảo mật và hiệu suất, hầu hết các thuật toán chữ ký số chỉ định rằng chỉ giá trị tóm lược của thông báo được "ký", chứ không phải toàn bộ thông báo. Các hàm băm cũng có thể được sử dụng trong việc tạo các bit giả ngẫu nhiên (pseudorandom)

Các giá trị băm còn được sử dụng để nhận dạng tập tin trên mạng chia sẻ peer-to-peer. Ví dụ, trong một liên kết ed2k, một giá trị băm MD4-biến thể được kết hợp với kích thước tập tin, cung cấp đủ thông tin để định vị các nguồn tập tin, tải các tập tin và xác nhận nội dung của nó. (Trong máy tính, các liên kết ed2k là các hyperlinks được sử dụng để biểu thị các tập tin được lưu trữ trong mạng eDonkey P2P.) Các liên kết Magnet là một ví dụ khác. Các giá trị băm tập tin như vậy thường là băm đầu danh sách băm hoặc cây băm để có thêm nhiều tiện lợi.

# 1.2 CẤU TRÚC MERKLE-DAMGARD

Một hàm băm phải có thể xử lý thông điệp độ dài tùy ý cho kết quả đầu ra có độ dài cố định. Điều này có thể đạt được bằng cách chặt đầu vào thành chuỗi các khối có kích thước bằng nhau, và tác động vào chúng theo thứ tự bằng cách sử dụng một hàm nén một chiều. Hàm nén hoặc có thể được thiết kế đặc biệt cho băm hoặc được xây dựng từ một thuật toán mã khối. Một hàm băm được xây dựng với cấu trúc Merkle-Damgård có khả năng kháng va chạm dựa vào hàm nén của nó; bất kỳ va chạm nào đối với hàm băm đầy đủ có thể đều bắt nguồn từ một va chạm trong các hàm nén.

  
Hình 1.1: Cấu trúc băm Merkle-Damgård.

Khối cuối xử lý cũng phải được thêm cho đủ độ dài khối; điều này rất quan trọng đối với tính an toàn của cấu trúc Merkle-Damgård. Hầu hết các hàm băm phổ dụng, bao gồm SHA-1 và MD5, thực hiện theo cấu trúc này.

Cấu trúc trên vẫn tiềm ẩn lỗi cố hữu, bao gồm các tấn công mở rộng độ dài và các tấn công tạo-và-dán, và không thể song song được. Kết quả là, có nhiều ứng cử viên trong cuộc thi về hàm băm mật mã của NIST được xây dựng dựa trên các kiến trúc khác.

# 1.3 BIRTHDAY ATTACK

Như đã biết, một dạng tấn công có khả năng đối với các hệ chữ ký điện tử có dùng hàm Băm là tìm cách tạo ra những văn bản x và x’ có nội dung khác nhau (một có lợi và một là bất lợi cho bên ký) mà giá trị Băm giống nhau. Kẻ địch có thể tìm cách tạo ra một số lượng rất lớn các văn bản có nội dung không thay đổi nhưng khác nhau về biểu diễn nhị phân (đơn giản là việc thêm bớt khoảng trắng hay dùng nhiều từ đồng nghĩa để thay thế ...), sau đó sử dụng một chương trình máy tính để tính giá trị Băm của các văn bản đó và đem so sánh với nhau để hi vọng tìm ra một cặp văn bản đụng độ (sử dụng phương pháp thống kê).

Nhưng việc này đòi hỏi số văn bản cần được tính giá trị Băm phải lớn hơn kích thước không gian Băm rất nhiều. Chẳng hạn như nếu hàm Băm có không gian Băm 64-bit thì số lượng văn bản cần được đem ra nạp vào chương trình phải ít nhất 264 (với một máy tính có thể thực hiện việc Băm 1 triệu bức điện trong 1 giây, thì phải mất 6000.000 năm tính toán.

Tuy nhiên nếu kẻ địch thử với lượng văn bản ít hơn nhiều, trong phạm vi có thể tính được thì xác suất để tìm được đụng độ sẽ như thế nào? Câu trả lời là “có thể thực hiện được”. Bản chất của hiện tượng này được minh hoạ rõ thông qua phát biểu sau, thường được gọi là nghịch lý ngày sinh (birthday paradox)

# 1.4 HÀM BĂM MẬT MÃ

Hàm băm mật mã là hàm toán học chuyển đổi một thông điệp có độ dài bất kỳ thành một dãy bit có độ dài cố định (tùy thuộc vào thuật toán băm). Dãy bit này được gọi là thông điệp rút gọn (message digest) hay giá trị băm (hash value), đại diện cho thông điệp ban đầu. Dễ dàng nhận thấy rằng hàm băm h không phải là một song ánh. Do đó, với thông điệp x bất kỳ, tồn tại thông điệp x’ ≠ x sao cho h(x)= h(x’). Lúc này, ta nói rằng “có sự đụng độ xảy ra”. Một hàm băm h được gọi là an toàn (hay “ít bị đụng độ”) khi không thể xác định được (bằng cách tính toán) cặp thông điệp x và x’ thỏa mãn x≠x’ và h(x) = h(x’). Trên thực tế, các thuật toán băm là hàm một chiều, do đó, rất khó để xây dựng lại thông điệp ban đầu từ thông điệp rút gọn. Hàm băm giúp xác định được tính toàn vẹn dữ liệu của thông tin: mọi thay đổi, dù là rất nhỏ, trên thông điệp cho trước, ví dụ như đổi giá trị 1 bit, đều làm thay đổi thông điệp rút gọn tương ứng. Tính chất này hữu ích trong việc phát sinh, kiểm tra chữ ký điện tử, các đoạn mã chứng nhận thông điệp, phát sinh số ngẫu nhiên, tạo ra khóa cho quá trình mã hóa…

Hàm băm là nền tảng cho nhiều ứng dụng mã hóa. Có nhiều thuật toán để thực hiện hàm băm, trong số đó, phương pháp SHA-1 và MD5 thường được sử dụng khá phổ biến từ thập niên 1990 đến nay.

* Hàm băm MD4 (Message Digest 4) và MD5 (Message Digest 5):

• Hàm băm MD4 được Giáo sư Ron Rivest đề nghị vào năm 1990. Vào năm 1992, phiên bản cải tiến MD5 của thuật toán này ra đời.

• Thông điệp rút gọn có độ dài 128 bit.

• Năm 1995, Hans Dobbertin đã chỉ ra sự đụng độ ngay chính trong bản thân hàm nén của giải thuật (mặc dù chưa thật sự phá vỡ được giải thuật). Năm 2004, nhóm tác giả Xiaoyun Wang, Dengguo Feng, Xuejia Lai và Hongbo Yu đã công bố kết quả về việc phá vỡ thuật toán MD4 và MD5 bằng phương pháp tấn công đụng độ

* Phương pháp Secure Hash Standard (SHS):

• Phương pháp Secure Hash Standard (SHS) do NIST và NSA xây dựng được công bố trên Federal Register vào ngày 31 tháng 1 năm 1992 và sau đó chính thức trở thành phương pháp chuẩn từ ngày 13 tháng 5 năm 1993.

• Thông điệp rút gọn có độ dài 160 bit. Ngày 26/08/2002, Viện Tiêu chuẩn và Công nghệ quốc gia của Hoa Kỳ (National Institute of Standard and Technology - NIST) đã đề xuất hệ thống chuẩn hàm băm an toàn (Secure Hash Standard) gồm 4 thuật toán hàm băm SHA-1, SHA-256, SHA-384, SHA-512. Đến 25/03/2004, NIST đã chấp nhận thêm thuật toán hàm băm SHA-224 vào hệ thống chuẩn hàm băm. Các thuật toán hàm băm do NIST đề xuất được đặc tả trong tài liệu FIPS180-2

# 1.5 CẤU TRÚC HÀM BĂM

Hầu hết các hàm băm mật mã đều có cấu trúc giải thuật như sau:

Cho trước một thông điệp M có độ dài bất kỳ. Tùy theo thuật toán được sử dụng, chúng ta có thể cần bổ sung một số bit vào thông điệp này để nhận được thông điệp có độ dài là bội số của một hằng số cho trước. Chia nhỏ thông điệp thành từng khối có kích thước bằng nhau: M1, M2, …Ms

Gọi H là trạng thái có kích thước n bit, f là “hàm nén” thực hiện thao tác trộn khối dữ liệu với trạng thái hiện hành. Khởi gán H0 bằng một vector khởi tạo nào đó Hi = f(Hi-1,M) với i = 1, 2, 3, …, s Hs chính là thông điệp rút gọn của thông điệp M ban đầu

# 1.6 TÍNH AN TOÀN HÀM BĂM VÀ HIỆN TƯỢNG ĐỤNG ĐỘ

Hàm băm được xem là an toàn đối với hiện tượng đụng độ khi rất khó tìm được hai thông điệp có cùng giá trị băm.

Nhận xét: Trong một tập hợp mà các phần tử mang một trong N giá trị cho trước với xác suất bằng nhau, chúng ta cần khoảng N phép thử ngẫu nhiên để tìm ra một cặp phần tử có cùng giá trị.

Như vậy, phương pháp hàm băm được xem là an toàn đối với hiện tượng đụng độ nếu chưa có phương pháp tấn công nào có thể tìm ra cặp thông điệp có cùng giá trị hàm băm với số lượng tính toán ít hơn đáng kể so với ngưỡng 2n/2, với n là kích thước (tính bằng bit) của giá trị băm.

Phương pháp tấn công dựa vào đụng độ:

• Tìm ra 2 thông điệp có nội dung khác nhau nhưng cùng giá trị băm.

• Ký trên một thông điệp, sau đó, người ký sẽ không thừa nhận đây là chữ

ký của mình mà nói rằng mình đã ký trên một thông điệp khác.

• Như vậy, cần phải chọn 2 thông điệp “đụng độ” với nhau trước khi ký.

# 1.7 TÍNH MỘT CHIỀU

Hàm băm được xem là hàm một chiều khi cho trước giá trị băm, không thể tái tạo lại thông điệp ban đầu, hay còn gọi là “tiền ảnh” (“pre-image”). Như vậy, trong trường hợp lý tưởng, cần phải thực hiện hàm băm cho khoảng 2n thông điệp để tìm ra được “tiền ảnh” tương ứng với một giá trị băm.

Nếu tìm ra được một phương pháp tấn công cho phép xác định được “tiền ảnh” tương ứng với một giá trị băm cho trước thì thuật toán băm sẽ không còn an toàn nữa. Cách tấn công nhằm tạo ra một thông điệp khác với thông điệp ban đầu nhưng có cùng giá trị băm gọi là tấn công “tiền ảnh thứ hai” (second pre-image attack).

# 1.8 GHÉP CÁC HÀM BĂM MẬT MÃ

Ghép các kết quả đầu ra từ nhiều hàm băm tạo ra tính kháng va chạm tối thiểu cũng tốt như là độ mạnh nhất của thuật toán trong các kết nối. Ví dụ, SSL sử dụng ghép nối MD5 và SHA-1 để đảm bảo giao thức đó sẽ vẫn an toàn thậm chí nếu một hàm băm bị phá vỡ.

Tuy nhiên, với các hàm băm Merkle-Damgård, hàm ghép nối chỉ mạnh như các thành phần tốt nhất, chứ không mạnh hơn. Joux chỉ ra rằng 2-va chạm dẫn đến n-va chạm nếu dễ dàng tìm ra 2 thông điệp có cùng giá trị băm MD5, thì không khó để tìm thấy nhiều thông điệp khi kẻ tấn công muốn lấy các giá trị băm MD5 giống nhau. Trong số n thông điệp với cùng giá trị băm MD5, có khả năng có được một va chạm SHA-1. Nhưng công việc cần thực hiện thêm là tìm va chạm SHA-1 (làm sao để tốt hơn tìm kiếm ngày sinh) là đa thức. Lập luận này được tổng kết bởi Finney.

# 1.9 THUẬT TOÁN BĂM MẬT MÃ

Có một danh sách dài các hàm băm mật mã, mặc dù trong đó có nhiều hàm băm được cho là dễ bị tổn thương và không nên sử dụng. Ngay cả khi một hàm băm chưa bị phá vỡ, một tấn công thành công đối với một biến thể yếu đó có thể làm giảm sự tự tin của các chuyên gia và dẫn đến loại bỏ nó. Ví dụ, vào tháng 8 năm 2004 người ta đã tìm ra những điểm yếu của một vài hàm băm phổ biến vào thời đó, bao gồm SHA-0, RIPEMD, và MD5. Điều này đã đặt ra câu hỏi an ninh lâu dài của các thuật toán sau này được bắt nguồn từ những hàm băm này - đặc biệt, SHA-1 (một phiên bản mạnh của SHA-0), RIPEMD-128, và RIPEMD-160 (cả hai phiên bản mạnh của RIPEMD). Vì vậy, cả SHA-0 và RIPEMD đều không được sử dụng rộng rãi kể từ khi chúng được thay thế bởi các phiên bản mạnh.

Đến năm 2009, hai hàm băm mật mã được sử dụng thông dụng nhất vẫn là MD5 và SHA-1. Tuy nhiên, MD5 đã bị phá vỡ do có một tấn công lên nó để phá vỡ SSL trong năm 2008 SHA-0 và SHA-1 là các thành viên của họ hàm băm SHA được phát triển bởi NSA. Vào tháng 2 năm 2005, đã tấn công thành công trên SHA-1, việc tìm kiếm va chạm trong khoảng 269 phép toán băm, thay vì 280 theo dự kiến cho hàm băm 160-bit. Vào tháng 8 năm 2005, có một tấn công thành công trên SHA-1 trong đó việc tìm kiếm va chạm chỉ cần 263 phép toán băm. Điểm yếu lý thuyết của SHA-1 tồn tại vốn có, nhưng gợi ý rằng có thể thực hiện về mặt thực tế để phá vỡ nó cũng phải mất vài năm. Các ứng dụng mới có thể tránh được những vấn đề này bằng cách sử dụng thêm các thành viên tiên tiến của họ SHA, như SHA-2, hoặc sử dụng các kỹ thuật như băm ngẫu nhiên hóa sẽ công quan tâm đến kháng va chạm.

Tuy nhiên, để đảm bảo tính chất mạnh lâu dài của các ứng dụng có sử dụng hàm băm, hiện có một cuộc thi nhằm thiết kế hàm băm thay thế cho SHA-2, nó sẽ có tên là SHA-3 và trở thành một tiêu chuẩn FIPS vào năm 2012

# 1.10 PHƯƠNG PHÁP SECURE HASH STANDARD (SHS)

Phương pháp Secure Hash Standard (SHS) do NIST và NSA xây dựng được

công bố trên Federal Register vào ngày 31 tháng 1 năm 1992 và sau đó chính thức trở thành phương pháp chuẩn từ ngày 13 tháng 5 năm 1993. Nhìn chung, SHS được xây dựng trên cùng cơ sở với phương pháp MD4 và MD5. Tuy nhiên, phương pháp SHS lại áp dụng trên hệ thống big-endian thay vì little-endian như phương pháp MD4 và MD5. Ngoài ra, thông điệp rút gọn kết quả của hàm băm SHS có độ dài 160 bit (nên phương pháp này thường được sử dụng kết hợp với thuật toán DSS).

# CHƯƠNG 2 : HÀM BĂM MD5

# 2.1 GIỚI THIỆU

Hàm băm MD4 (Message Digest 4) được Giáo sư Rivest đề nghị vào năm 1990. Vào năm sau, phiên bản cải tiến MD5 của thuật toán này ra đời. Cùng với phương pháp SHS, đây là ba phương pháp có ưu điểm tốc độ xử lý rất nhanh nên thích hợp áp dụng trong thực tế đối với các thông điệp dài.

Thông điệp ban đầu x sẽ được mở rộng thành dãy bit X có độ dài là bội số của 512. Một bit 1 được thêm vào sau dãy bit x, tiếp đến là dãy gồm d bit 0 và cuối cùng là dãy 64 bit l biểu diễn độ dài của thông điệp x. Dãy gồm d bit 0 được thêm vào sao cho dãy X có độ dài là bội số 512.

# 2.2 KHÁI NIỆM

MD5 (Message-Digest algorithm 5) là một hàm băm để mã hóa với giá trị băm là 128bit. Từng được xem là một chuẩn trên Internet, MD5 đã được sữ dụng rộng rải trong các chương trình an ninh mạng, và cũng thường được dùng để kiểm tra tính nguyên vẹn của tập tin.

MD5 được thiết kế bởi Ronald Rivest vào năm 1991 để thay thế cho hàm băm trước đó, MD4 (cũng do ông thiết kế, trước đó nữa là MD2).

# 2.3 ỨNG DỤNG

Md5 có 2 ứng dụng quan trọng:

* MD5 được sử dụng rộng rải trong thế giới phần mềm để đảm bảo rằng tập tin tải về không bị hỏng. Người sử dụng có thể so sánh giữa thông số kiểm tra phần mềm bằng MD5 được công bố với thông số kiểm tra phần mềm tải về bằng MD5. Hệ điều hành Unix sử dụng MD5 để kiểm tra các gói mà nó phân phối, trong khi hệ điều hành Windows sử dụng phần mềm của hãng thứ ba.
* MD5 được dùng để mã hóa mật khẩu. Mục đích của việc mã hóa này là biến đổi một chuổi mật khẩu thành một đoạn mã khác, sao cho từ đoạn mã đó không thể nào lần trở lại mật khẩu. Có nghĩa là việc giải mã là không thể hoặc phải mất một khoãng thời gian vô tận (đủ để làm nản lòng các hacker).

# 2.4 GIẢI THUẬT

MD5 biến đổi một thông điệp có chiều dài bất kì thành một khối có kích thước cố định 128 bits. Thông điệp đưa vào sẻ được cắt thành các khối 512 bits. Thông điệp được đưa vào bộ đệm để chiều dài của nó sẻ chia hết cho 512. Bộ đệm hoạt động như sau:

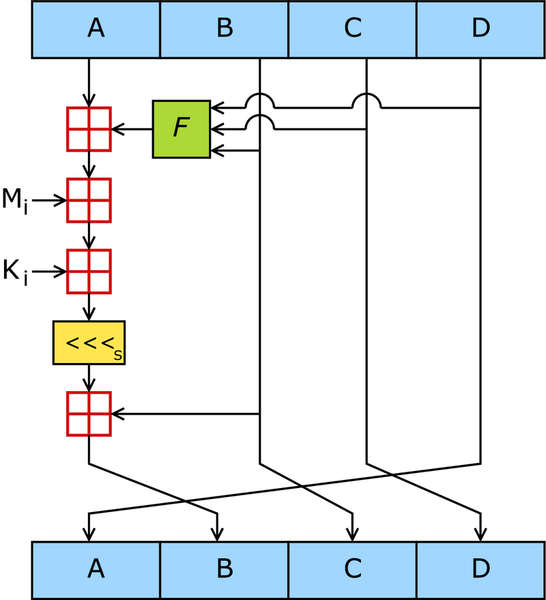
- Trước tiên chèn bit 1 vào cuối thông điệp.

- Tiếp đó là hàng loạt bit Zero cho tới khi chiều dài của nó nhỏ hơn bội số của 512

một khoảng 64 bit .

- Phần còn lại sẽ được lấp đầy bởi một số nguyên 64 bit biểu diển chiều dài ban đầu của thông điệp. Thuật toán chính của MD5 hoạt động trên một bộ 128 bit. Chia nhỏ nó ra thành 4 từ 32 bit, kí hiệu là A,B,C và D. Các giá trị này là các hằng số cố định. Sau đó thuật toán chính sẻ luân phiên hoạt động trên các khối 512 bit. Mỗi khối sẻ phối hợp với một bộ. Quá trình xữ lý một khối thông điệp bao gồm 4 bước tương tự nhau, gọi là vòng (“round”). Mỗi vòng lại gồm 16 quá trình tương tự nhau dựa trên hàm một chiều F, phép cộng module và phép xoay trái…

Hình bên dưới mô tả một quá trình trong một vòng. Có 4 hàm một chiều F có thể sử dụng. Mỗi vòng sử dụng một hàm khác nhau.



Hàm băm MD5 (còn được gọi là hàm tóm tắt thông điệp - message degests) sẻ trả về một chuổi số thập lục phân gồm 32 số liên tiếp.

Dưới đây là các ví dụ mô tả các kết quả thu được sau khi băm.

* *MD5("The quick brown fox jumps over the lazy dog")*

= 9e107d9d372bb6826bd81d3542a419d6

Thậm chỉ chỉ cần một thay đổi nhỏ cũng làm thay đổi hoàn toàn kết quả trả về:

* *MD5("The quick brown fox jumps over the lazy cog")*

= 1055d3e698d289f2af8663725127bd4b

Ngay cả một chuổi rổng cũng cho ra một kết quả phức tạp:

* MD5(" ") = d41d8cd98f00b204e9800998ecf8427e

# 2.5 MD5 (Message Digest)

Ronald Rivest là người đã phát minh ra các hàm Băm MD2, MD4 (1990) và MD5 (1991). Do tính chất tương tự của các hàm Băm này, sau đây chúng ta sẽ xem xét hàm Băm MD5, đây là một cải tiến của MD4 và là hàm Băm được sử dung rộng rãi nhất, nguyên tắc thiết kế của hàm băm này cũng là nguyên tắc chung cho rất nhiều các hàm băm khác

2.11.5.1 Miêu tả MD5:

Đầu vào là những khối 512-bit, được chia cho 16 khối con 32-bit. Đầu ra của thuật toán là một thiết lập của 4 khối 32-bit để tạo thành một hàm Băm 128-bit duy nhất. Đầu tiên, ta chia bức điện thành các khối 512-bit, với khối cuối cùng (đặt là x và x <512-bit) của bức điện, chúng ta cộng thêm một bit 1 vào cuối của x, theo sau đó là các bit 0 để được độ dài cần thiết (512 bit). Kết quả là bức điện vào là một chuỗi M có độ dài chia hết cho 512, vì vậy ta có thể chia M ra thành các N word 32-bit (N word này sẽ chia hết cho 16).

Bây giờ, ta bắt đầu tìm cốt của bức điện với 4 khối 32-bit A, B, C và D (được xem như thanh ghi) :

A = 0x01234567

B = 0x89abcdef

C = 0xfedcba98

D = 0x76543210.

Người ta thường gọi A, B, C, D là các chuỗi biến số (chaining variables).

Bức điện được chia ra thành nhiều khối 512-bit, mỗi khối 512-bit lại được chia ra 16 khối 32-bit đi vào bốn vòng lặp của MD5. Giả sử ta đặt a, b, c và d thay cho A, B, C và D đối với khối 512-bit đầu tiên của bức điện. Bốn vòng lặp trong MD5 đều có cấu trúc giống nhau. Mỗi vòng thực hiện 16 lần biến đổi: thực hiện với một hàm phi tuyến của 3 trong 4 giá trị a, b, c và d sau đó nó cộng kết quả đến giá trị thứ 4, tiếp đó cộng với một khối con 32-bit và một hằng số. Sau đó, nó dịch trái một lượng bit thay đổi và cộng kết quả vào một trong 4 giá trị a, b, c hay d. Kết quả cuối cùng là một giá trị mới được thay thế một trong 4 giá trị a, b, c hay d.

Vòng 1

Vòng 2

Vòng 3

Vòng 4

**A**

**B  
C  
D**

**A**

**B  
C  
D**

Hình 1.2: Sơ đồ vòng lặp chính của MD5

Mj

ti

**a**

**b**

**c**

**<<<S**

**d**

Hình 1.3: Sơ đồ 1 vòng lặp của MD5

Có bốn hàm phi tuyến, mỗi hàm này được sử dụng cho mỗi vòng:

F (X, Y, Z) = (X **∧** Y) **∨** ((**¬** X) **∧** Z)

G(X, Y, Z) = (X **∧** Z) **∨** ((**¬** Z) **∧** Y)

H (X, Y, Z) = X **⊕** Y **⊕** Z

I (X, Y, Z) = Y **⊕** (X **∨** (¬Z))

**quy ước:**

X **∧** Y Phép toán AND trên bit giữa X và Y

X **∨** Y Phép toán OR trên bit giữa X và Y

X **⊕** Y Phép toán XOR trên bit giữa X và Y

¬X Phép toán NOT trên bit của X

X + Y Phép cộng (modulo )

X <<< s Các bit của X được dịch chuyển xoay vòng sang trái s vị trí (0 ≤ s < 32)

Những hàm này được thiết kế sao cho các bit tương ứng của X, Y và Z là độc lập và không ưu tiên, và mỗi bit của kết quả cũng độc lập và ngang bằng nhau.

Nếu Mj là một biểu diễn của khối con thứ j (j = 16) và <<<s là phép dịch trái của s bit, thì các vòng lặp có thể biểu diễn như sau:

FF(a,b,c,d,Mj,s,ti) được biểu diễn a = b + ((a + F(b,c,d) + Mj + ti)<<<s)

GG(a,b,c,d,Mj,s,ti)được biểu diễn a=b + ((a + G(b,c,d) + Mj + ti) <<<s)

HH(a,b,c,d,Mj,s,ti) được biểu diễn a=b+ ((a+H(b,c,d) + Mj + ti) <<< s)

II(a,b,c,d,Mj,s,ti) được biểu diễn a = b + ((a + I(b,c,d) + Mj + ti) <<< s).

2.11.5.2 Cách thực hiện:

Bốn vòng (64 bước) sẽ thực hiện như sau:

*Vòng 1:*

FF (a, b, c, d, M0, 7, 0x76aa478)

FF (d, a, b, c, M1, 12, 0xe8c7b756)

FF (c, d, a, b, M2, 17, 0x242070db)

FF (b, c, d, a, M3, 22, 0xc1bdceee)

FF (a, b, c, d, M4, 7, 0xf57c0faf)

FF (d, a, b, c, M5, 12, 0x4787c62a)

FF (c, d, a, b, M6, 17, 0xa8304613)

FF (b, c, d, a, M7, 22, 0xfd469501)

FF (a, b, c, d, M8, 7, 0x698098d8)

FF (d, a, b, c, M9, 12, 0x8b44f7af)

FF (c, d, a, b, M10, 17, 0xffff5bb1)

FF (b, c, d, a, M11, 22, 0x895cd7be)

FF (a, b, c, d, M12, 7, 0x6b901122)

FF (d, a, b, c, M13, 12, 0xfd987193)

FF (c, d, a, b, M14, 17, 0xa679438e)

FF (b, c, d, a, M15, 22, 0x49b40821).

*Vòng 2:*

GG (a, b, c, d, M1, 5, 0x61e2562)

GG (d, a, b, c, M6, 9, 0xc040b340)

GG (c, d, a, b, M11, 14, 0x265e5a51)

GG (b, c, d, a, M0, 20, 0xe9b6c7aa)

GG (a, b, c, d, M5, 5, 0xd62f105d)

GG (d, a, b, c, M10, 9, 0x02441453)

GG (c, d, a, b, M15, 14, 0xd8a1e681)

GG (b, c, d, a, M4, 20, 0xe7d3fbc8)

GG (a, b, c, d, M9, 5, 0x21e1cde6)

GG (d, a, b, c, M14, 9, 0xc33707d6)

GG (c, d, a, b, M3, 14, 0xf4d50d87)

GG (b, c, d, a, M8, 20, 0x455a14ed)

GG (a, b, c, d, M13, 5, 0xa9e3e905)

GG (d, a, b, c, M2, 9, 0xfcefa3f8)

GG (c, d, a, b, M7, 14, 0x676f02d9)

GG (b, c, d, a, M12, 20, 0x8d2a4c8a).

*Vòng 3:*

HH (a, b, c, d, M5, 4, 0xfffa3942)

HH (d, a, b, c, M8, 11, 0x8771f681)

HH (c, d, a, b, M11, 16, 0x6d9d6122)

HH (b, c, d, a, M14, 23, 0xfde5380c)

HH (a, b, c, d, M1, 4, 0xa4beea44)

HH (d, a, b, c, M4, 11, 0x4bdecfa9)

HH (c, d, a, b, M7, 16, 0xf6bb4b60)

HH (b, c, d, a, M10, 23, 0xbebfbc70)

HH (a, b, c, d, M13, 4, 0x289b7ec6)

HH (d, a, b, c, M0, 11, 0xeaa127fa)

HH (c, d, a, b, M3, 16, 0xd4ef3085)

HH (b, c, d, a, M6, 23, 0x04881d05)

HH (a, b, c, d, M9, 4, 0xd9d4d039)

HH (d, a, b, c, M12, 11, 0xe6db99e5)

HH (c, d, a, b, M15, 16, 0x1fa27cf8)

HH (b, c, d, a, M2, 23, 0xc4ac5665).

*Vòng 4:*

II (a, b, c, d, M0, 6, 0xf4292244)

II (d, a, b, c, M7, 10, 0x432aff97)

II (c, d, a, b, M14, 15, 0xab9423a7)

II (b, c, d, a, M5, 21, 0xfc93a039)

II (a, b, c, d, M12, 6, 0x655b59c3)

II (d, a, b, c, M3, 10, 0x8f0ccc92)

II (c, d, a, b, M10, 15, 0xffeff47d)

II (b, c, d, a, M1, 21, 0x85845dd1)

II (a, b, c, d, M8, 6, 0x6fa87e4f)

II (d, a, b, c, M15, 10, 0xfe2ce6e0)

II (c, d, a, b, M6, 15, 0xa3013414)

II (b, c, d, a, M13, 21, 0x4e0811a1)

II (a, b, c, d, M4, 6, 0xf7537e82)

II (d, a, b, c, M11, 10, 0xbd3af235)

II (c, d, a, b, M2, 15, 0x2ad7d2bb)

II (b, c, d, a, M9, 21, 0xeb86d391).

Những hằng số ti được chọn theo quy luật sau: ở bước thứ i giá trị ti là phần nguyên của 232\*abs(sin(i)), trong đó i = [0..63] được tính theo radian.

Sau tất cả những bước này a, b, c và d lần lượt được cộng với A, B, C và D để cho kết quả đầu ra và thuật toán tiếp tục với khối dữ liệu 512-bit tiếp theo cho đến hết bức điện. Đầu ra cuối cùng là một khối 128-bit của A, B, C và D, đây chính là hàm Băm nhận được.

# CHƯƠNG 3: XÂY DỰNG CHƯƠNG TRÌNH MÔ PHỎNG THUẬT

# 3.1 NHIỆM VỤ CỦA CHƯƠNG TRÌNH

- Đưa ra thuật toán MD5

- Sơ đồ khối thuật toán MD5

- Kết quả mô phỏng của chương trình

- Kết quả thực nghiệm

# 3.2 THUẬT TOÁN VÀ SƠ ĐỒ KHỐI

## 3.2.1 Thuật toán:

Khởi gán các biến:

h0 := 0x67452301.

h1 := 0xEFCDAB89.

h2 := 0x98BADCFE.

h3 := 0x10325476.

Hệ số quay trái *R*[*i*]của mỗi chu kỳ:

*R*[ 0..15] := { 7, 12, 17, 22, 7, 12, 17, 22,

7, 12, 17, 22, 7, 12, 17, 22}

*R*[16..31] := { 5, 9, 14, 20, 5, 9, 14, 20,

5, 9, 14, 20, 5, 9, 14, 20}

*R*[32..47] := { 4, 11, 16, 23, 4, 11, 16, 23,

4, 11, 16, 23, 4, 11, 16, 23}

*R*[48..63] := { 6, 10, 15, 21, 6, 10, 15, 21,

6, 10, 15, 21, 6, 10, 15, 21}

Hằng số K[*i*]

**for** *i* **from** 0 **to** 63

*K*[*i*] := floor(abs(sin(*i* + 1)) × (2 **pow** 32))

* **Tiền xử lý:**
* Thêm bit 1 vào cuối thông điệp.
* Thêm vào k bit 0 sao cho độ dài thông điệp nhận được đồng dư 448

(mod 512).

* Thêm 64 bit biểu diễn độ dài của thông điệp gốc(giá trị lưu dạng big-endian).

***M***

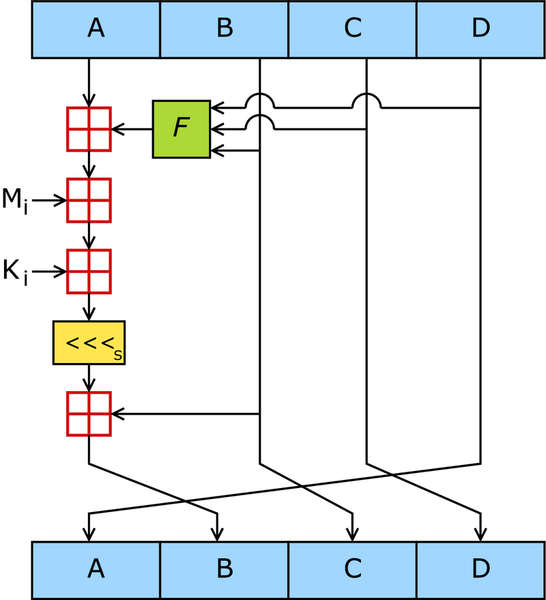
**1**

**0…0**

***m***

1 bit

* **Quá trình xử lý:**
* Chia thông điệp (đã padding) thành các khối 512 bit
* Với mỗi khối 512-bit:
  + Chia thành 16 word (32 bit, little-endian) w[0..15]
  + A= h0, B= h1, C= h2, D= h3
  + 64 chu kỳ xử lý
  + h0+=A, h1+=B, h2+=C, h3+=D
* Kết quả:= h0 | h1 | h2 | h3
* **Chu kỳ xử lý trong MD5**



Trong đó:

* *t* là số thứ tự của chu kỳ.
* A, B, C, D, là 4 word (32 bit) của trạng thái.
* *F* là hàm phi tuyến (thay đổi tùy theo chu kỳ).
* Đầu tiên, bốn biến A, B, C, D được khởi tạo. Những biến này được gọi là chaining variables.

Bốn chu kỳ biến đổi trong MD5 hoàn toàn khác nhau và lần lượt sử dụng các hàm F, G, H và I. Mỗi hàm có tham số X, Y, Z là các từ 32 bit và kết quả là một từ 32 bit.

* *<<< s* là phép quay trái *s* vị trí.
* ⊞ phép cộng modulo 232.

Mã giả thuật toán md5

H0 = 0x67452301;

H1 = 0xEFCDAB89;

H2 = 0x98BADCFE;

H3 = 0x10325476;

**for** *i* **from** 0 to 63

*f* = *F*[*i*] (B, C, D)

*g* = G[*i*] (*i*)

temp = D

D = C

C = B

B = ((A + *f* + *K*[*i*] + w[*g*])

**<<<** R[*i*]) + B

A = temp

0 ≤ *i* ≤ 15

*f* := (B **∧** C) **∨** ((**¬** B) **∧** D)

*g* := *i*

16 ≤ *i* ≤ 31

*f* := (D **∧** B) **∨** ((**¬** D) **∧** C)

*g* := (5×*i* + 1) **mod** 16

32 ≤ *i* ≤ 47

*f* := B **⊕** C **⊕** D

*g* := (3×*i* + 5) **mod** 16

48 ≤ *i* ≤ 63

*f* := C **⊕** (B **∨** (**¬** D))

*g* := (7×*i*) **mod** 16

## 3.2.2. Sơ đồ khối thuật toán MD5

**Sơ đồ khối tổng quát**

F

T

H0:=0x67452301

H1:=0xEFCDAB89

H2:=0x98BADCFE

H3:=0x10325476

Chia thông điệp thành M khối 512 bit

i<=M

i=1

i++

A=H0; B=H1; C=H2; D=H3;

Xử lý trên mỗi khối 512 bit

MD: H0 H1 H2 H3

H0=H0+A; H1=H1+B;

H2=H2+C; H3=H3+D

Tiền xử lý

**Sơ đồ khối Tiền xử lý:**

**F**

**T**

m=chiều dài thông điệp

Str[m+1]=1

k=1

(m+1+k)%512 !=448

Str[m+1+k]=0

k++

i=64

i!=0

Str[m+1+k+i]=m%2

m=m/2

**F**

**T**

i--

**Sơ đồ khối quá trình xử lý trên mỗi khối 512 bit:**

S

Đ

Đ

Đ

S

Đ

S

i =1

i <=15

i <=31

i <=47

*f* := (B **∧** C) **∨** ((**¬** B) **∧** D)

*g* := *i*

*f* := (D **∧** B) **∨** ((**¬** D) **∧** C)

*g* := (5×*i* + 1) **mod** 16

*f* := B **⊕** C **⊕** D

*g* := (3×*i* + 5) **mod** 16

*f* := C **⊕** (B **∨** (**¬** D))

*g* := (7×*i*) **mod** 16

temp = D , D = C , C = B

B = ((A + *f* + *K*[*i*] + w[*g*])

**<<<** R[*i*]) + B

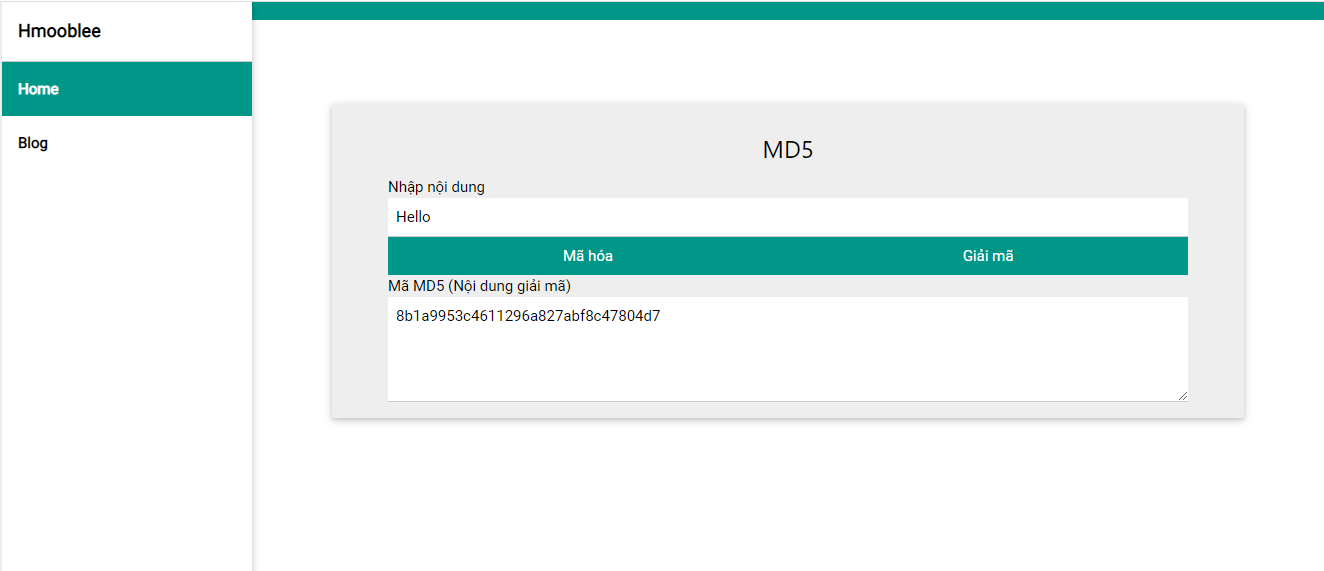
i <=64

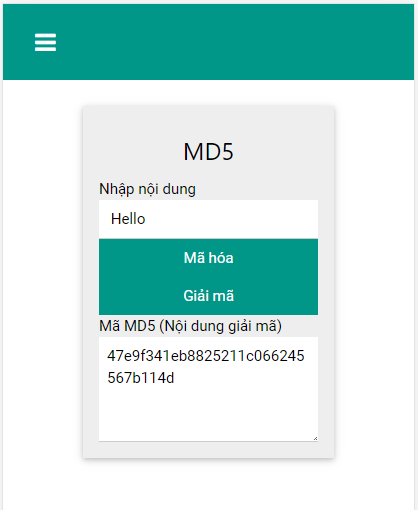
i =i+1

S

# 3.3 KẾT QUẢ CHƯƠNG TRÌNH MÔ PHỎNG THUẬT TOÁN BĂM MD5

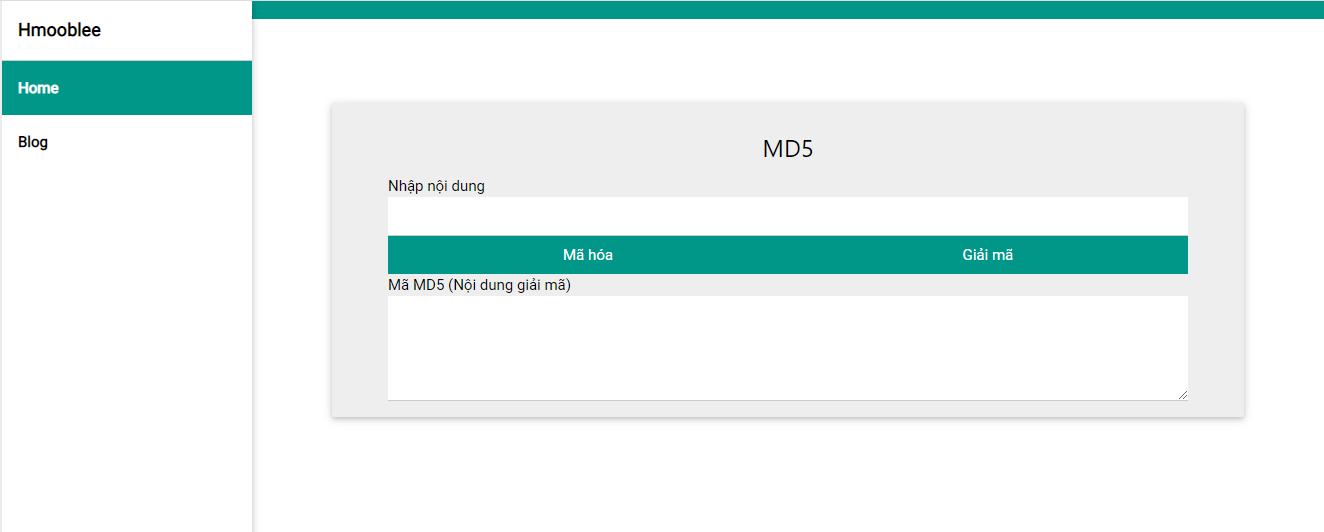
# 3.3.1. Giao diện chương trình mô phỏng

****

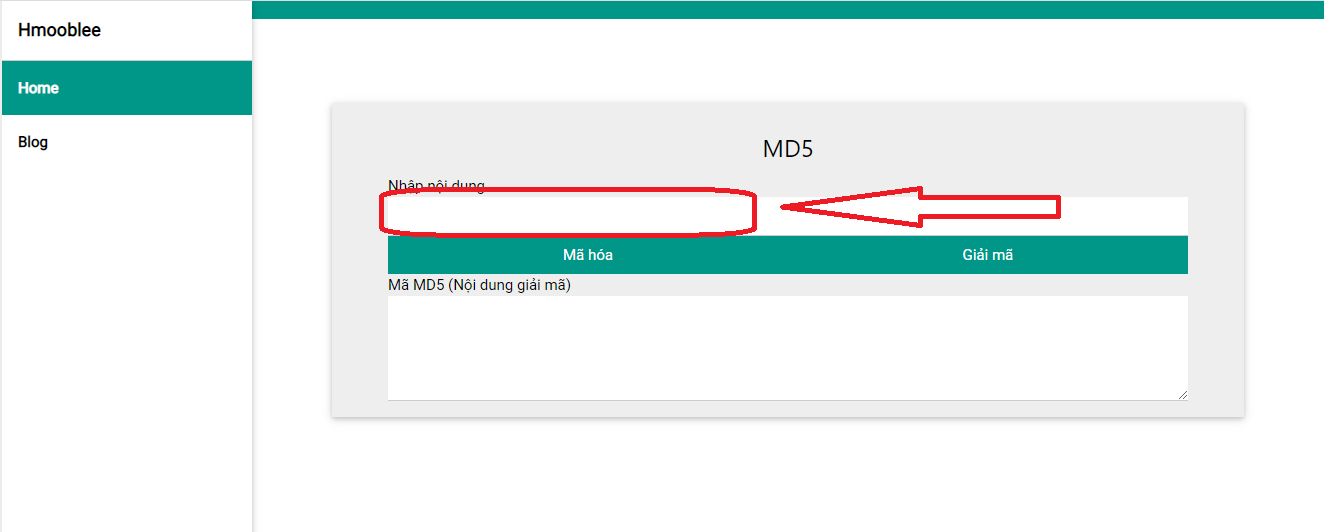
****

# 3.3.2 Các bước thực hiện chương trình

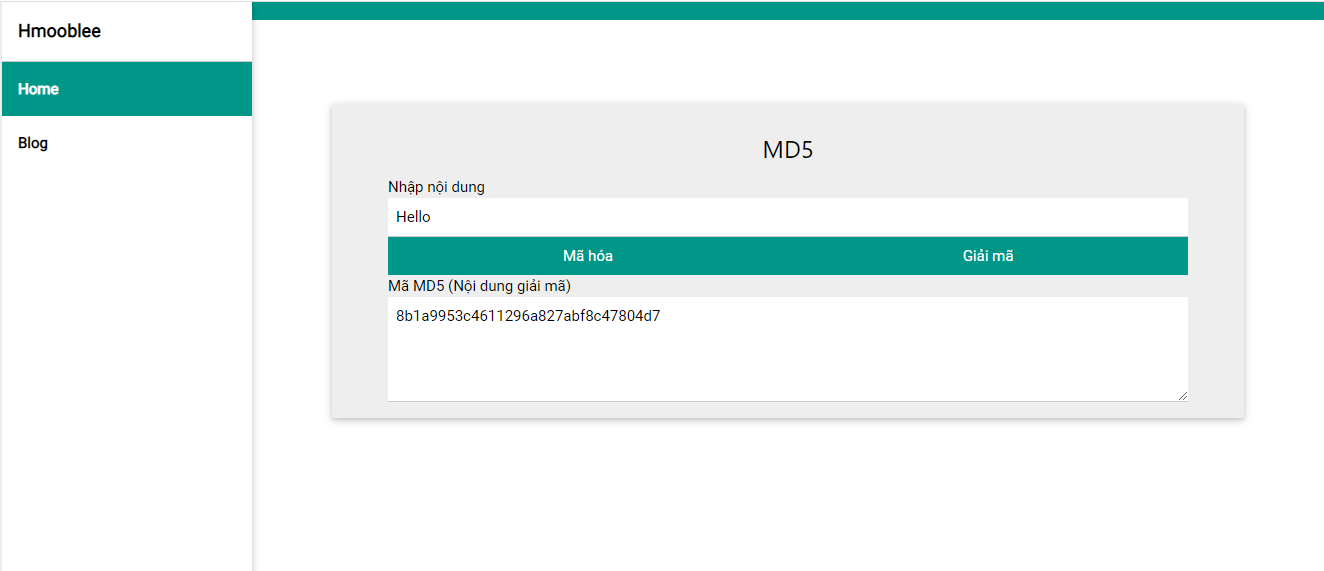
**Bước 1:** Chạy Project md5 sẽ xuất hiện giao diện chương trình mô phỏng



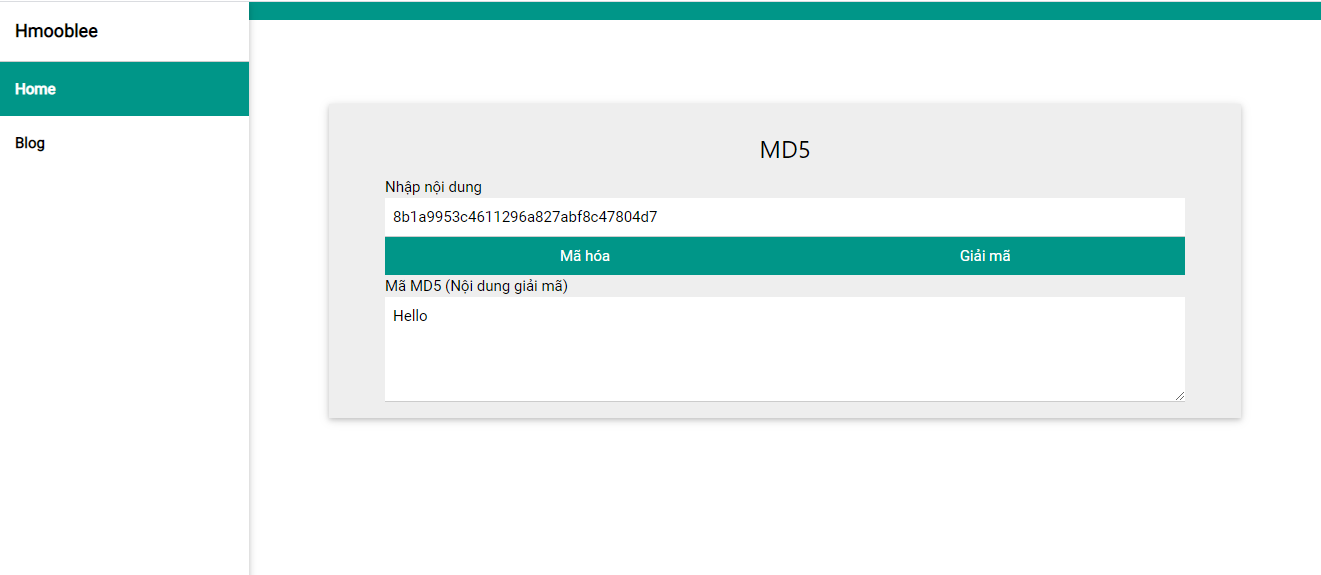
**Bước 2 :** Nhập thông điệp muốn mã hóa vào chỗ Thông điệp vào

****

**Bước 3 :** Sau đó chọn nút mã hóa . Ta sẽ có kết quả tại khung kết quả

****

**Bước 4 :** Giải mã. Hãy nhập mã MD5 vào ô nhập nội dung rồi chọn nút giải mã, Kết quả sẽ hiện thị ở khung Mã MD5(Nội dung giải mã). Trường hợp không hiển thị là do mã MD5 sai hoặc chưa có mã trong cơ sở dữ liệu.

****

# *3.3.3 Các kết quả thử nghiệm*

**Kết quả 1 :** MD5("Hello")

= 8b1a9953c4611296a827abf8c47804d7

**Kết quả 2 :** MD5("12345")

= 827ccb0eea8a706c4c34a16891f84e7b

**Kết quả 1 :** MD5(" ")

= 7215ee9c7d9dc229d2921a40e899ec5f

**Kết luận:** Trong chương này, chúng ta đã thực hiện được việc tìm hiểu về thuật toán băm MD5 trên các khía cạnh:

- Tìm hiểu, nắm rõ thuật toán, các bước thực hiện thuật toán.

- Biểu diễn sơ đồ khối của thuật toán.

- Thực hiện mô phỏng thuật toán băm MD5 và các kết quả thu được đúng với các tài liệu tiêu chuẩn về thuật toán MD5, chứng tỏ chương trình mô phỏng đã được thực hiện chính xác theo thuật toán.

# KẾT LUẬN

Sau một thời gian tập trung nghiên cứu, Em đã tìm hiểu được một số vấn đề về thuật toán MD5. Qua đó cho thấy rằng hàm băm là nền tảng để bảo đảm an ninh trong lĩnh vực thương mại điện tử, các phần mềm quản lý có kiến trúc kiểu Client/Server. Em đã nêu được quy trình ứng dụng thuật toán băm MD5. Từ đó, đề ra giải pháp ứng dụng hàm băm trong phần mềm quản lý, cụ thể là quá trình mã hóa văn bản. nhằm hướng tới việc xây dựng được hệ mật mã có độ an toàn cao hơn và chữ ký số có tính xác thực cao hơn trong tương lai.

Đề tài cũng chọn ra một giải pháp tạo chữ ký số tương đối mạnh là sử dụng thuật toán băm MD5 để xây dựng thử nghiệm chương trình ứng dụng.

Tuy nhiên đây là một lĩnh vực khó và thời gian hạn chế nên em mới chỉ dừng lại ở việc xây dựng chương trình demo. Hướng tiếp là xây dựng thành một chương trình có thể ứng dụng rộng rải hơn.

Với thời lượng và kiến thức em còn hạn chế nên chắc chắn em sẽ không tránh khỏi thiếu sót, rất mong nhận được sự đóng góp ý kiến của các thầy cô và bạn bè để hoàn thiện hơn.

# TÀI LIỆU THAM KHẢO

[1].https://vi.wikipedia.org/wiki/Java\_(ng%C3%B4n\_ng%E1%BB%AF\_l%E1%BA%ADp\_tr%C3%ACnh)

[2]. https://en.wikipedia.org/wiki/Digital\_signature

[3]. https://vi.wikipedia.org/wiki/MD5

[4].https://vi.wikipedia.org/wiki/H%C3%A0m\_b%C4%83m\_m%E1%BA%ADt\_m%C3%A3\_h%E1%BB%8Dc

[5]. https://vi.wikipedia.org/wiki/MD5#Các\_bảng\_băm\_MD5

[6]. https://en.wikipedia.org/wiki/MD5#Preimage\_vulnerability

[7]. https://vi.wikipedia.org/wiki/MD5#Mã\_giả

[8].

[https://en.wikipedia.org/wiki/Birthday\_attack](https://en.wikipedia.org/wiki/Birthday_attack 9)

[9]. Internet

# NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

…………………………………………………………………………………………...

*Thái Nguyên, ngày tháng năm 2018*

**Giáo viên hướng dẫn**