<https://learncryptography.com/hash-functions/what-are-hash-functions>

**Hash (hàm băm) là gì?**

Một hàm băm F nhận một input có giá trị x và sinh ra một output có giá trị y tương ứng xác định.



X – có thể là bất kì kiểu dữ liệu nào, ví dụ như số, files, etc.

Y – gọi là một kết quả băm (hash); thường được biểu diễn dưới dạng hexa.

Ví dụ: hàm băm MD5, tạo ra một kết quả 32 kí tự dạng hexa (128 bit) cho bất kì input nào.



Đặc tính quan trọng của hàm băm là không thể đảo ngược (irreversible); tức là gần như không thể có công thức tính được input x khi biết trước output y. Cách duy nhất là sử dụng duyệt trâu ☺ brute-force attack, duyệt tất cả các khả năng có thể của input.

Một trường hợp sử dụng hàm băm như sau: Giả sử Alice nói với Bob rằng cô ấy biết kết quả của một bài toán trong lớp đại số. Bob muốn Alice chứng tỏ rằng cô ấy đang không khoác loác mà không để lộ kết quả ra cho Bob. Alice sử dụng hàm băm để băm kết quả của cô ấy (giả sử đáp số là 42):



Alice đưa kết quả băm này cho Bob. Như vậy, Bob không thể biết được đáp số thật sự là gì; tuy nhiên khi Bob giải được đáp số thì anh ấy có thể băm kết quả ra và so sánh với kết quả băm của Alice.

Như vậy các hàm băm thường được sử dụng để xác nhận thông tin (verify) mà không tiết lộ thông tin chính xác đằng sau là gì.

<https://en.wikipedia.org/wiki/Secure_Hash_Algorithms>

Một số họ thuật toán băm và biến thể là MD5 (128 bit); SHA-0, SHA-1 (160 bit); SHA-2, trong đó có SHA-256 (được dùng trong bitcoin), SHA-3.

**Tại sao các hàm băm lại không thể đảo ngược?**

Mặc dù cách áp dụng cụ thể là khác nhau nhưng nhìn chung tất cả các hàm băm đều sử dụng phép toán lấy phần dư (modulo operator). Ví dụ:



Hoặc



Có thể thấy rằng phép toán mod là không thể đảo ngược. Giả sử bạn biết kết quả lấy mod là 4, nhưng vấn đề là có vô số các cặp số có thể sử dụng để lấy được kết quả này.

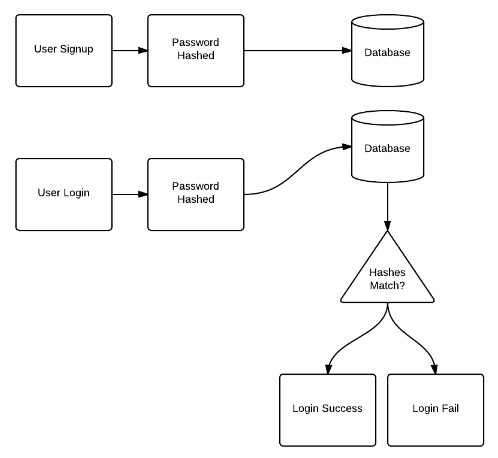
Hơn nữa, rất nhiều thông tin đã bị lược bỏ trong quá trình băm vì chuỗi input có thể rất dài nhưng chuỗi output thì luôn là một tập hợp có độ dài cố định tùy theo thuật toán.

Do đó hầu như không thể lấy được dữ liệu gốc chỉ bằng kết quả băm – cách duy nhất là duyệt vét cạn các cặp dữ liệu. Nếu chúng ta có thể đảo ngược lại một hàm băm, chúng ta có thể nén dữ liệu bất kì xuống kích cỡ chỉ còn một vài byte.

**Lưu trữ mật khẩu với các hàm băm**

Các website lưu trữ thông tin người dùng vào cơ sở dữ liệu (database) bao gồm username, mật khẩu, và các thông tin cá nhân khác. Các hacker có thể đột nhập vào database để lấy các thông tin và truy cập trái phép vào các tài khoản một khi đã biết mật khẩu. Để hạn chế khả năng lộ mật khẩu người dùng, **password hashing** ra đời.

Password hashing: thay vì lưu trữ nguyên định dạng text của mật khẩu, người ta sẽ đưa mật khẩu đó qua một hàm băm và lưu trữ kết quả băm của nó. Lần đăng nhập tiếp theo, mật khẩu được người dùng nhập vào sẽ được đưa qua cùng một hàm băm và so sánh hai kết quả băm. Như vậy, kể cả khi thông tin trong database bị lộ - không ai có thể đăng nhập vào tài khoản vì họ chỉ biết được kết quả băm chứ không biết mật khẩu thực tế là gì.

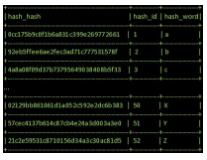


Hiệu ứng avalanche: ngay cả một thay đổi rất nhỏ trong input cũng gây nên sự thay đổi rất lớn ở output.

Ví dụ, Bob đăng kí mật khẩu là “hunter2”, server sử dụng hàm băm MD5 và lưu trữ kết quả băm là “2ab96390c7dbe3439de74d0c9b0b1767”. Lần đăng nhập tiếp theo, Bob nhập mật khẩu là “hunter1”, server đưa chuỗi này qua xử lý hàm băm MD5 và nhận được “726ad07bc398372b56a52e3de8693679”. Hai kết quả băm của “hunter1” và “hunter2” sai khác nhau rất lớn mặc dù hai chuỗi gốc chỉ khác nhau một kí tự (hiệu ứng avalanche) nên Bob không thể đăng nhập.

**Rainbow Tables**

Một rainbow table là một cơ sở dữ liệu chứa rất nhiều input và output tương ứng của các hàm băm.



Rainbow table được tạo ra bằng cách tính toán trước các kết quả của hàm băm với một tập các input cho trước (như vậy phần tính toán sẽ được lược bỏ). Khi đó trong quá trình duyệt vét cạn, chúng ta chỉ cần so sánh kết quả băm với giá trị lưu trong bảng và “nếu may mắn” trùng hợp, chúng ta sẽ thu được dữ liệu gốc.

Tất nhiên nhược điểm của rainbow table là chỉ hiệu quả đối với một tập giới hạn các kết quả băm nhất định. Nếu kết quả cần tìm không xuất hiện trong bảng thì nó là vô nghĩa dù bạn có lưu trữ nhiều cặp input – output như thế nào đi chăng nữa.

**Password Salting**

Rainbow Table attack: hacker có thể xây dựng một bảng tính toán trước kết quả băm của các mật khẩu thường được sử dụng, ví dụ như “123456”, “abc123”, etc. Nhờ đó, hacker có thể so sánh trực tiếp các giá trị trong bảng với kết quả băm có trong database và tìm ra mật khẩu của user tương ứng.

Một biện pháp phòng tránh tình trạng này là sử dụng **password salting**.



Một giá trị salt được cộng thêm vào input của hàm băm. Nội dung của salt là ngẫu nhiên và độc nhất với mỗi user. Salt được lưu cùng mật khẩu, sử dụng trong quá trình lưu trữ và xác nhận mật khẩu.

Rõ ràng, nếu hacker cố so sánh kết quả băm trong database với danh sách tính toán trước các kết quả băm của các mật khẩu thông dụng thì không chính xác. Vì giá trị salt đã làm thay đổi hoàn toàn kết quả của hàm băm.

**Hash Collision Attack**

Tấn công Hash Collision là cố gắng tìm ra hai chuỗi input có kết quả băm giống nhau khi sử dụng cùng một hàm băm. Trên lý thuyết, khả năng này có thể xảy ra vì độ dài chuỗi input là không giới hạn, trong khi độ dài chuỗi output là cố định. Đây được gọi là một sự trùng hợp hoặc va chạm (collision).

Tuy xác suất để xảy ra va chạm là rất nhỏ, đặc biệt là với các hàm băm có độ dài chuỗi output lớn. Tuy nhiên với khả năng tính toán của máy tính ngày càng cao thì việc duyệt vét cạn để tìm ra hash collision là có thể thực hiện được, nhờ đó lợi dụng sơ hở của các ứng dụng so sánh kết quả băm – ví dụ băm mật khẩu hay kiểm tra độ toàn vẹn của file.

Ví dụ, giả sử chúng ta có một hàm băm có kết quả băm như sau:

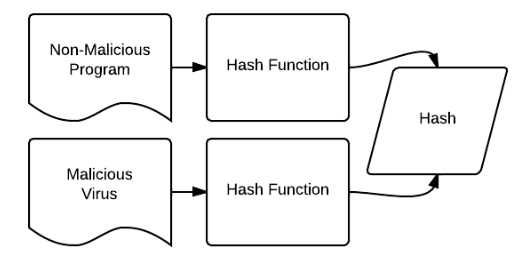


Bây giờ hacker cần phải tìm ra một sự va chạm – bằng phương pháp duyệt vét cạn, giả sử hacker tìm được chuỗi:



Kết quả băm của hai chuỗi input trên là như nhau.

Nhờ vậy, hacker có thể lợi dụng trong trường hợp người dùng tải một file và nó sẽ được băm để xác nhận rằng file tải xuống là toàn vẹn. Hacker thay đổi file gốc bằng một file chứa virus mà vẫn có cùng kết quả băm. Như vậy người dùng không thể biết được sự khác nhau vì file virus vẫn hoàn toàn hợp lệ.

****

Chúng ta có cần lo lắng về hash collision? Câu trả lời phụ thuộc vào hàm băm. MD5 và SHA-1 không hoàn toàn miễn nhiễm trước tấn công hash collision. Tuy nhiên các hàm mạnh hơn như SHA-256 là an toàn trong khả năng tính toán hiện nay.