1. Lịch sử phát triển của argon2

-Argon2 được phát triển bởi các nhà nghiên cứu bảo mật Alex Biryukov, Daniel Dinu và Dmitry Khovratovich vào năm 2015. Họ đã thiết kế Argon2 nhằm cải tiến và thay thế các hàm băm cũ như bcrypt và scrypt, những hàm băm này đã trở nên lỗi thời và dễ bị tấn công.

Argon2 đã được chấp nhận làm một trong những hàm băm mạnh nhất và được khuyến khích sử dụng trong các ứng dụng bảo mật. Nó đã được sử dụng trong các thư viện bảo mật như libsodium và bcrypt để bảo mật mật khẩu.

Argon2 đã giành được nhiều giải thưởng trong các cuộc thi bảo mật quốc tế, bao gồm giải thưởng "Best Password Hashing Competition" tại hội nghị PasswordsCon 2015 và giải thưởng "Best Paper Award" tại hội nghị EUROCRYPT 2016.

Argon2 cũng đã được sử dụng trong các ứng dụng thực tế như 1Password, KeePassXC và Bitwarden để bảo mật mật khẩu của người dùng.

Tóm lại, Argon2 là một trong những hàm băm mạnh nhất và được khuyến khích sử dụng trong các ứng dụng bảo mật. Nó đã được phát triển bởi các nhà nghiên cứu bảo mật vào năm 2015 và đã giành được nhiều giải thưởng trong các cuộc thi bảo mật quốc tế.

2. Giới thiệu hàm băm Argon2

-Argon2 là một hàm băm (hash function) được thiết kế để sử dụng trong các ứng dụng bảo mật như mật khẩu (password) và chứng chỉ số (digital certificate). Hàm băm này được chấp nhận là một trong những hàm băm mạnh nhất hiện nay.

-Argon2 được thiết kế để chống lại các cuộc tấn công bằng phương pháp lưu trữ trước (pre-computed hash) và các cuộc tấn công bằng phương pháp sử dụng đa nhân (parallel computation). Nó sử dụng một số tham số đầu vào như kích thước bộ nhớ (memory size), số vòng lặp (number of iterations), và kích thước salt để tăng độ khó của quá trình băm.

-Argon2 có hai phiên bản chính là Argon2d và Argon2i. Argon2d được thiết kế để chống lại các cuộc tấn công bằng phương pháp đa nhân, trong khi Argon2i được thiết kế để chống lại các cuộc tấn công bằng phương pháp lưu trữ trước.

-Argon2 được sử dụng rộng rãi trong các thư viện bảo mật như libsodium và bcrypt để bảo mật mật khẩu.

3. Các hàm băm Argon2

-Có hai loại chính của Argon2 là Argon2d và Argon2i:

-Argon2d được thiết kế để chống lại các cuộc tấn công bằng phương pháp đa nhân (parallel computation). Nó sử dụng một hàm băm được gọi là BLAKE2b để tính toán kết quả băm. Tuy nhiên, Argon2d không được khuyến khích sử dụng trong các ứng dụng bảo mật vì nó có thể dễ dàng bị tấn công bằng phương pháp lưu trữ trước (pre-computed hash).

-Argon2i được thiết kế để chống lại các cuộc tấn công bằng phương pháp lưu trữ trước (pre-computed hash). Nó sử dụng một hàm băm được gọi là BLAKE2b để tính toán kết quả băm, nhưng kết quả này được sử dụng để tạo ra một ma trận (matrix) kết hợp với các giá trị salt và mật khẩu. Argon2i được khuyến khích sử dụng trong các ứng dụng bảo mật vì nó đảm bảo tính chống lại các cuộc tấn công bằng phương pháp lưu trữ trước.

-Ngoài ra, Argon2 còn có một phiên bản khác được gọi là Argon2id.

-Argon2id là sự kết hợp của Argon2i và Argon2d, nó kết hợp đặc tính của cả hai loại Argon2 để đảm bảo tính chống lại các cuộc tấn công bằng cả phương pháp lưu trữ trước và đa nhân. Argon2id là phiên bản được khuyến khích sử dụng trong các ứng dụng bảo mật.

3. Thuật toán của Argon2

Argon2 là một thuật toán băm mật khẩu hiện đại, được thiết kế để cung cấp độ bảo mật cao và khả năng chống lại các cuộc tấn công bằng lực brute. Argon2 sử dụng nhiều kỹ thuật bảo mật khác nhau để tăng cường tính bảo mật của nó.

Thuật toán Argon2 được chia thành hai pha chính:

1. Pha chuẩn bị (Setup Phase): Trong pha này, Argon2 sẽ xác định các tham số đầu vào và tạo ra các khối dữ liệu để sử dụng trong quá trình băm mật khẩu.

a. Xác định các tham số đầu vào: Argon2 nhận vào các tham số đầu vào, bao gồm mật khẩu (password), chuỗi ngẫu nhiên (salt), kích thước bộ nhớ (memory size), số lần lặp lại (iterations), số lượng luồng (parallelism).

b. Tạo các khối dữ liệu: Argon2 sử dụng các tham số đầu vào để tạo ra các khối dữ liệu để sử dụng trong quá trình băm mật khẩu.

1. Pha băm (Hashing Phase): Trong pha này, Argon2 sử dụng các khối dữ liệu đã được tạo ra trong pha chuẩn bị để băm mật khẩu. Quá trình băm được thực hiện bằng cách sử dụng một chuỗi các hàm băm (hashing function) và một số vòng lặp để tăng độ bảo mật và khó bị tấn công bằng lực brute.

a. Tạo khối mật khẩu (Password Block): Argon2 sử dụng mật khẩu và salt để tạo ra một khối dữ liệu đầu tiên, gọi là khối mật khẩu.

b. Tạo khối phân tán (S-Box Block): Argon2 sử dụng khối mật khẩu để tạo ra một khối phân tán đầu tiên, gọi là khối S-Box.

c. Tạo khối bộ nhớ (Memory Block): Argon2 sử dụng khối S-Box để tạo ra một khối dữ liệu mới, gọi là khối bộ nhớ, sử dụng kỹ thuật phân bổ bộ nhớ (memory allocation).

d. Thực hiện vòng lặp (Iteration): Argon2 thực hiện một chuỗi các vòng lặp để tăng độ bảo mật và khó bị tấn công bằng lực brute.

e. Tạo mã băm (Hash): Sau khi hoàn thành quá trình lặp lại, Argon2 sử dụng khối đầu ra cuối cùng để tạo ra mã băm (hash).

Argon2 có nhiều phiên bản với các tham số khác nhau để đáp ứng các yêu cầu khác nhau về độ bảo mật và hiệu suất.

Tổng quan, Argon2 là một thuật toán băm mật khẩu hiện đại và được coi là một trong những thuật toán băm mật khẩu an toàn nhất hiện nay. Argon2 sử dụng các kỹ thuật bảo mật khác nhau để tăng cường tính bảo mật của nó, bao gồm sử dụng bộ nhớ và phân tán. Argon2 cũng cung cấp nhiều phiên bản với các tham số khác nhau để đáp ứng các yêu cầu khác nhau về độ bảo mật và hiệu suất.

Argon2 là một thuật toán băm mật khẩu hiện đại và có thể được sử dụng trong nhiều ứng dụng khác nhau. Sau đây là một số ví dụ về các ứng dụng mà Argon2 có thể được sử dụng:

1. Quản lý tài khoản: Argon2 có thể được sử dụng để bảo vệ mật khẩu của người dùng trong các hệ thống quản lý tài khoản, chẳng hạn như hệ thống quản lý tài khoản ngân hàng trực tuyến.
2. Ứng dụng di động: Argon2 có thể được sử dụng để bảo vệ mật khẩu của người dùng trong các ứng dụng di động, bao gồm ứng dụng được sử dụng để quản lý thông tin cá nhân, ứng dụng giao dịch và các ứng dụng khác.
3. Hệ thống quản lý cơ sở dữ liệu: Argon2 có thể được sử dụng để bảo vệ mật khẩu của người dùng trong các hệ thống quản lý cơ sở dữ liệu, chẳng hạn như các hệ thống quản lý cơ sở dữ liệu của các doanh nghiệp và tổ chức.
4. Hệ thống đăng nhập: Argon2 có thể được sử dụng để bảo vệ mật khẩu của người dùng trong các hệ thống đăng nhập, chẳng hạn như hệ thống đăng nhập của các trang web và ứng dụng.
5. Blockchain: Argon2 có thể được sử dụng trong các hệ thống blockchain để bảo vệ mật khẩu của người dùng và tăng tính bảo mật của các giao dịch.

Argon2 là một trong những hàm băm mạnh nhất hiện nay và có nhiều ưu điểm, tuy nhiên cũng có một số nhược điểm nhất định.

Ưu điểm của Argon2:

* Tính chống lại các cuộc tấn công bằng phương pháp lưu trữ trước (pre-computed hash) và đa nhân (parallel computation).
* Dễ dàng sử dụng và tích hợp vào các ứng dụng bảo mật.
* Có thể điều chỉnh độ khó của quá trình băm bằng cách sử dụng các tham số đầu vào như kích thước bộ nhớ (memory size), số vòng lặp (number of iterations), và kích thước salt.
* Được sử dụng rộng rãi trong các thư viện bảo mật như libsodium và bcrypt để bảo mật mật khẩu.

Nhược điểm của Argon2:

* Tốn nhiều tài nguyên hệ thống, đặc biệt là bộ nhớ, do đó có thể làm chậm quá trình xử lý.
* Không thể sử dụng trên các hệ thống cũ hoặc thiết bị có tài nguyên hạn chế.
* Đôi khi khó để điều chỉnh độ khó của quá trình băm để đảm bảo tính bảo mật và hiệu suất tối ưu.
* Chưa được sử dụng rộng rãi trong các ứng dụng bảo mật, do đó có thể có một số rủi ro về tính ổn định và độ tin cậy.

Tuy nhiên, với các ưu điểm vượt trội của mình, Argon2 vẫn là một lựa chọn tốt cho các ứng dụng bảo mật và được sử dụng rộng rãi trong các thư viện bảo mật.

\*Argon2 là một hàm băm mật mã được thiết kế để tăng cường bảo mật của các ứng dụng lưu trữ mật khẩu bằng cách sử dụng một số kỹ thuật khác nhau để khó khăn hơn cho kẻ tấn công tìm ra mật khẩu gốc.

Trong Argon2, "salt" là một chuỗi ngẫu nhiên được sử dụng để kết hợp với mật khẩu người dùng trước khi băm. Salt được sử dụng để đảm bảo rằng các mật khẩu giống nhau sẽ được băm thành các giá trị khác nhau, điều này làm cho việc tấn công bằng cách sử dụng bảng băm được lưu trữ trước đó trở nên khó khăn hơn đáng kể.

Việc sử dụng salt trong Argon2 là một trong những kỹ thuật cơ bản để tăng cường bảo mật và đảm bảo rằng các mật khẩu được lưu trữ an toàn hơn.

1. Tạo salt: Trong quá trình tạo mật khẩu, Argon2 sẽ tạo ra một chuỗi ngẫu nhiên gọi là "salt". Salt thường có kích thước lớn hơn hoặc bằng với kích thước của block trong hàm băm Blake2b. Salt được tạo ra bằng cách sử dụng các nguồn ngẫu nhiên, chẳng hạn như thời gian hệ thống, số lượng bộ nhớ trống, hoặc hỗn hợp của các giá trị này.
2. Kết hợp salt và mật khẩu: Sau khi tạo salt, Argon2 sẽ kết hợp salt với mật khẩu người dùng bằng cách thực hiện phép XOR (hoặc các phép toán tương tự). Kết quả của phép XOR là một giá trị mới được gọi là "key". Key có cùng kích thước với salt và được sử dụng để tạo ra bản mã từ mật khẩu người dùng.
3. Tạo bản mã: Argon2 sử dụng key để tạo ra bản mã từ mật khẩu người dùng. Quá trình này được thực hiện bằng cách sử dụng một hàm băm đặc biệt gọi là Blake2b. Hàm băm này chuyển đổi các khối dữ liệu lớn thành các giá trị băm ngắn hơn. Quá trình chuyển đổi này được thực hiện nhiều lần để tạo ra một bản mã với độ dài bằng với kích thước của block trong hàm băm.
4. Lưu trữ salt và bản mã: Salt và bản mã được lưu trữ trong cơ sở dữ liệu. Salt được lưu trữ cùng với bản mã để đảm bảo rằng các mật khẩu giống nhau của người dùng sẽ được băm thành các giá trị khác nhau. Khi người dùng đăng nhập, mật khẩu của họ sẽ được kết hợp với salt đã lưu trữ trong cơ sở dữ liệu và băm để so sánh với bản mã đã lưu trữ.
5. Quá trình kết hợp salt và mật khẩu được thực hiện như sau: salt được tạo ngẫu nhiên với kích thước tối thiểu là 8 byte. Sau đó, salt được kết hợp với mật khẩu bằng phép XOR để tạo ra một khóa (key) mới. Khóa này có cùng kích thước với salt và được sử dụng để tạo ra bản mã từ mật khẩu người dùng.
6. Phép XOR được sử dụng để kết hợp salt và mật khẩu vì nó là một phép toán ngược lại, có thể được dùng để khôi phục lại giá trị ban đầu khi biết kết quả và một trong hai giá trị ban đầu. Việc sử dụng phép XOR cũng đảm bảo rằng giá trị key được tạo ra sẽ không bị lặp lại nếu cùng mật khẩu được sử dụng với các salt khác nhau.

\*Bộ nhớ của Argon2 được chia thành hai phần: bộ nhớ tạm và bộ nhớ sử dụng để lưu trữ bản mã (hash). Cả hai phần đều được sử dụng để khó khăn hơn cho kẻ tấn công tìm ra mật khẩu gốc.

1. Bộ nhớ tạm (memory): Bộ nhớ tạm được sử dụng để tạo ra bản mã từ mật khẩu người dùng. Kích thước của bộ nhớ tạm được đặt bởi tham số được gọi là "-m", và thường nằm trong khoảng từ 1MB đến 2GB. Bộ nhớ tạm được chia thành các khối bằng nhau được gọi là "lanes". Mỗi lane có thể được xử lý độc lập và được sử dụng để tính toán các giá trị trung gian trong quá trình tạo bản mã.
2. Bộ nhớ hash: Sau khi tạo bản mã từ mật khẩu người dùng, bản mã được lưu trữ trong bộ nhớ hash. Kích thước của bộ nhớ hash được đặt bởi tham số được gọi là "-t", và thường nằm trong khoảng từ 1 đến 4. Bộ nhớ hash được sử dụng để lưu trữ bản mã và đảm bảo rằng các giá trị băm khác nhau sẽ được lưu trữ trong các vị trí khác nhau trong bộ nhớ.

\*Để tính toán bộ nhớ được sử dụng bởi Argon2, bạn cần biết các thông số khối lượng công việc và bộ nhớ của thuật toán.

Argon2 có hai đối số liên quan đến bộ nhớ:

* m\_cost: đây là số lượng bộ nhớ được sử dụng bởi Argon2 để tính toán hàm băm. Được đo bằng byte.
* t\_cost: đây là số lượng vòng lặp được thực hiện bởi Argon2. Điều này ảnh hưởng đến thời gian tính toán và bộ nhớ được sử dụng.

Công thức tính toán bộ nhớ sử dụng của Argon2 như sau:

memory = m\_cost \* 1024 \* t\_cost / 4

Trong đó:

* m\_cost được đo bằng kilobyte, do đó chúng ta nhân với 1024 để chuyển đổi thành byte.
* t\_cost được chia cho 4, bởi vì Argon2 sử dụng bộ lưu trữ 4-đường (4-lane memory) để tối đa hóa hiệu suất.