**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**ĐẠI HỌC QUỐC GIA HÀ NỘI**

**KHOA VẬT LÝ**

**-----🙠🕮🙢-----**



**BÁO CÁO**

**MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**ĐỀ TÀI**

**NGHIÊN CỨU CÁC GIẢI THUẬT MÃ HÓA**

Cán bộ hướng dẫn: TS Phạm Huy Thông

Họ và tên các sinh viên thực hiện:

* Nguyễn Xuân An
* Võ Đình Đại
* Trần Ngọc Phúc
* Trịnh Trọng Phước
* Nguyễn Hữu Thắng

Lớp: K66 Kỹ Thuật Điện Tử và Tin Học

**Hà Nội, Năm 2025**

**ĐÓNG GÓP CỦA CÁC THÀNH VIÊN TRONG NHÓM**

|  |  |  |
| --- | --- | --- |
| **Tên thành viên** | **Công việc** | **Phần trăm đóng góp** |
| Nguyễn Xuân An |  | 20% |
| Võ Đình Đại | Viết báo cáo | 20% |
| Trần Ngọc Phúc |  | 20% |
| Trịnh Trọng Phước |  | 20% |
| Nguyễn Hữu Thắng |  | 20% |

**LỜI CẢM ƠN**

Nhóm chúng em xin chân thành cảm ơn TS. Phạm Huy Thông, thầy đã tận tình hướng dẫn và hỗ trợ chúng em trong suốt quá trình học tập để thực hiện đề tài ”Nghiên cứu các giải thuật mã hóa”. Những sự chỉ dẫn và giảng dạy của thầy trong môn học ”Cấu trúc dữ liệu và giải thuật” đã giúp chúng em có thêm kiến thức, kỹ năng và hoàn thành tốt đề tài cuối kỳ này.

Chúng em cũng xin gửi lời cảm ơn đến Khoa Vật Lý đã tạo điều kiện thuận lợi và cung cấp môi trường học tập lý tưởng để chúng em có thể thực hiện đề tài này.

Nhóm chúng em đã cố gắng làm tốt nhất có thể cho đề tài này, nhưng chắc chắn không tránh khỏi những thiếu sót. Chúng em rất mong nhận được sự đóng góp ý kiến từ thầy và các bạn để ngày càng hoàn thiện hơn.

Xin trân trọng cảm ơn!

Nhóm 6

**MỤC LỤC**

No table of contents entries found.

**DANH MỤC HÌNH VẼ**

**No table of figures entries found.**

# TỔNG QUAN VỀ MÃ HÓA

## Giới thiệu chung

### Lịch sử phát triển của mã hóa

Mã hóa (encryption) là một trong những lĩnh vực lâu đời nhất của khoa học và công nghệ, xuất hiện từ thời kỳ cổ đại như một cách để bảo vệ thông tin bí mật. Qua hàng nghìn năm, mã hóa đã phát triển không ngừng, từ những kỹ thuật thủ công đơn giản đến các hệ thống mã hóa phức tạp dựa trên toán học hiện đại và công nghệ điện tử. Sau đây là phát triển của mã hóa qua các thời kỳ.

**Giai đoạn cổ đại: Sự ra đời của mã hóa**

Mã hóa đã xuất hiện từ thời kỳ cổ đại như một công cụ để bảo vệ thông điệp quân sự hoặc thông tin quan trọng. Một trong những hệ thống mã hóa đầu tiên được biết đến là mã Caesar, do Julius Caesar phát minh vào thế kỷ I TCN. Hệ thống này dịch chuyển các ký tự trong bảng chữ cái theo một số vị trí cố định. Mặc dù đơn giản, mã Caesar đã đặt nền móng cho các hệ thống mã hóa thay thế (substitution cipher).

Tại Hy Lạp cổ đại, một công cụ khác được sử dụng để mã hóa là Scytale, một hình trụ gỗ được dùng để tạo mã bằng cách quấn giấy hoặc da xung quanh nó. Khi không có hình trụ đúng kích thước, thông điệp sẽ không thể giải mã.

Tương tự, tại Ấn Độ và Trung Quốc, các kỹ thuật mã hóa thô sơ cũng được sử dụng để bảo vệ thư tín và thông điệp quan trọng.

**Thời kỳ trung cổ: Sự phát triển của mã hóa thủ công**

Trong thời kỳ trung cổ, mã hóa tiếp tục được phát triển với các kỹ thuật thay thế và chuyển vị phức tạp hơn. Một ví dụ tiêu biểu là mã Vigenère, được phát minh vào thế kỷ XVI bởi nhà ngoại giao người Pháp Blaise de Vigenère. Đây là một hệ thống mã hóa dựa trên bảng chữ cái lặp lại, sử dụng một khóa dài hơn để mã hóa, giúp tăng cường tính bảo mật so với mã Caesar.

Ngoài ra, trong thời kỳ này, mã hóa không chỉ được sử dụng cho quân sự mà còn xuất hiện trong các lĩnh vực khác như thương mại và ngoại giao. Các quốc gia châu Âu thường sử dụng mã hóa để bảo vệ thông tin liên quan đến giao dịch và các hiệp ước quốc tế.

**Thời kỳ hiện đại sớm: Sự xuất hiện của mật mã học toán học**

Sự ra đời của toán học hiện đại đã đưa mã hóa lên một tầm cao mới. Trong thế kỷ XIX, nhà toán học người Anh Charles Babbage đã nghiên cứu và phá vỡ mã Vigenère, đặt nền móng cho việc phân tích tần suất trong mật mã học.

Đầu thế kỷ XX, máy mã hóa cơ học bắt đầu xuất hiện, tiêu biểu là máy Enigma của Đức trong Thế chiến II. Máy Enigma sử dụng các rotor cơ học để mã hóa thông điệp và được coi là một trong những hệ thống mã hóa phức tạp nhất vào thời điểm đó. Việc giải mã Enigma bởi nhóm của Alan Turing tại Bletchley Park không chỉ thay đổi cục diện chiến tranh mà còn mở đường cho sự phát triển của khoa học máy tính và mật mã học hiện đại.

**Thời kỳ hiện đại: Mã hóa dựa trên máy tính**

Với sự xuất hiện của máy tính điện tử trong nửa sau thế kỷ XX, mã hóa chuyển từ các hệ thống cơ học sang các thuật toán dựa trên toán học phức tạp. Năm 1976, hai nhà khoa học Whitfield Diffie và Martin Hellman đã giới thiệu khái niệm mã hóa khóa công khai (Public Key Cryptography). Phát minh này đã thay đổi căn bản cách thức bảo mật thông tin, cho phép trao đổi dữ liệu an toàn mà không cần chia sẻ trước khóa bí mật.

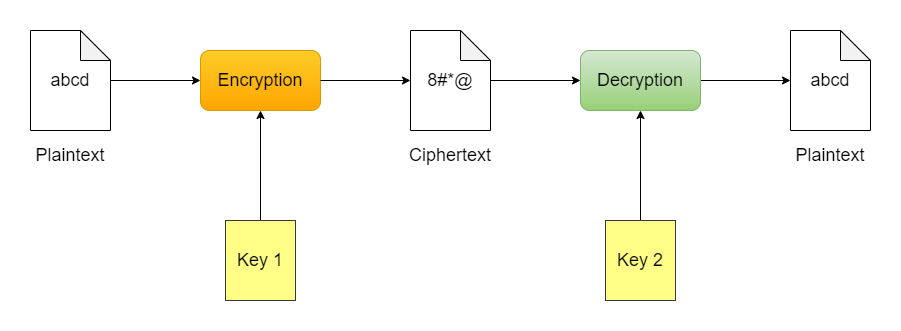
Một năm sau, vào năm 1977, thuật toán RSA (Rivest-Shamir-Adleman) được công bố, trở thành một trong những thuật toán mã hóa bất đối xứng phổ biến nhất. Đồng thời, các hệ thống mã hóa đối xứng như DES (Data Encryption Standard) cũng được triển khai rộng rãi trong các ứng dụng thương mại và quân sự.

**Thế kỷ XXI: Mã hóa trong kỷ nguyên kỹ thuật số**

Vào thế kỷ XXI, với sự bùng nổ của Internet và công nghệ di động, mã hóa đã trở thành một phần không thể thiếu trong mọi lĩnh vực của đời sống. Các thuật toán như AES (Advanced Encryption Standard), được phát triển vào năm 2001, cung cấp một mức độ bảo mật cao và hiệu quả, được sử dụng rộng rãi trong ngân hàng, thương mại điện tử, và các hệ thống mạng.

Gần đây, các cuộc tấn công vào hệ thống mã hóa và sự phát triển của máy tính lượng tử đã thúc đẩy nghiên cứu về mã hóa hậu lượng tử (Post-Quantum Cryptography), nhằm đảm bảo an ninh trước các mối đe dọa từ công nghệ mới.

### Các khái niệm cơ bản



Hình 1. Sơ đồ của quá trình mã hóa và giải mã thông tin

Hình trên mô tả quá trình mã hóa và giải mã thông tin từ người gửi đến người nhận. Để hiểu rõ cách hoạt động của các hệ thống mã hóa, cần nắm vững các khái niệm cơ bản như **Plaintext, Ciphertext, Key, Encryption,** và **Decryption.** Đây là những thành phần cốt lõi cấu thành quá trình bảo mật thông tin.

#### Plaintext (Bản rõ)

Plaintext là thông tin ban đầu, được biểu diễn ở dạng có thể đọc và hiểu được. Đây là dữ liệu chưa qua bất kỳ quá trình mã hóa nào và thường là đối tượng cần được bảo vệ khỏi các truy cập trái phép.

Ví dụ:

##### Một email chứa nội dung quan trọng như "Tài liệu mật này chỉ dành cho quản lý."

##### Một số tài khoản hoặc mật khẩu được gửi qua hệ thống mạng.

Plaintext là đầu vào của quá trình mã hóa và đầu ra của quá trình giải mã, đảm bảo dữ liệu được truyền đi hoặc lưu trữ một cách an toàn.

#### Ciphertext (Bản mã)

Ciphertext là kết quả của quá trình mã hóa plaintext. Đây là phiên bản dữ liệu đã được mã hóa và ở dạng không thể đọc hiểu trừ khi có khóa giải mã phù hợp.

Ví dụ:

##### Nội dung "Tài liệu mật này chỉ dành cho quản lý." có thể được mã hóa thành "9f7a8e0b3c..." bằng thuật toán AES.

Ciphertext giúp bảo vệ thông tin bằng cách làm cho dữ liệu trở nên vô nghĩa đối với những ai không có quyền truy cập.

#### Key (Khóa)

Key là một phần thông tin quan trọng được sử dụng trong cả quá trình mã hóa và giải mã. Khóa đóng vai trò quyết định tính bảo mật của hệ thống mã hóa, vì dữ liệu không thể giải mã nếu không có khóa tương ứng.

Có hai loại khóa chính trong mật mã học:

##### Khóa đối xứng (Symmetric Key): Sử dụng cùng một khóa để mã hóa và giải mã dữ liệu. Ví dụ: thuật toán AES, DES.

##### Khóa bất đối xứng (Asymmetric Key): Sử dụng một cặp khóa gồm khóa công khai (public key) để mã hóa và khóa riêng tư (private key) để giải mã. Ví dụ: thuật toán RSA.

Độ dài và độ phức tạp của khóa quyết định mức độ an toàn của hệ thống mã hóa.

#### ****Encryption (Mã hóa)****

Encryption là quá trình chuyển đổi plaintext thành ciphertext để bảo vệ dữ liệu khỏi các truy cập trái phép. Quá trình này sử dụng một thuật toán mã hóa và một khóa để thực hiện chuyển đổi.

Mục đích của mã hóa là đảm bảo dữ liệu vẫn an toàn ngay cả khi bị đánh cắp hoặc bị chặn trong quá trình truyền tải.

Ví dụ:

##### Khi bạn nhập mật khẩu vào một trang web, mật khẩu đó thường được mã hóa trước khi được gửi đến máy chủ để tránh lộ thông tin.

#### ****Decryption (Giải mã)****

Decryption là quá trình ngược lại của mã hóa, tức là chuyển đổi ciphertext trở lại dạng plaintext ban đầu. Quá trình này yêu cầu sử dụng khóa giải mã phù hợp với thuật toán đã mã hóa dữ liệu.

Ví dụ:

##### Khi bạn nhận một email mã hóa, phần mềm hoặc hệ thống sẽ giải mã dữ liệu để hiển thị nội dung email ban đầu.

Mục tiêu của giải mã là đảm bảo rằng dữ liệu chỉ có thể được đọc bởi những người được phép truy cập.

## Tầm quan trọng của mã hóa

### Bảo mật dữ liệu trong thời đại số

Trong bối cảnh công nghệ phát triển mạnh mẽ, dữ liệu đã trở thành tài sản quan trọng không chỉ đối với cá nhân mà còn với tổ chức và quốc gia. Tuy nhiên, cùng với sự bùng nổ của dữ liệu, các mối đe dọa bảo mật ngày càng gia tăng, đòi hỏi những giải pháp hiệu quả để bảo vệ thông tin. Mã hóa, với vai trò là một trong những công cụ bảo mật quan trọng nhất, đóng vai trò không thể thiếu trong việc bảo vệ dữ liệu trong thời đại số. Dưới đây là 5 khía cạnh chính thể hiện tầm quan trọng của mã hóa.

#### ****Bảo vệ thông tin cá nhân và quyền riêng tư****

Thông tin cá nhân hiện nay được thu thập và lưu trữ ở nhiều nơi, từ các nền tảng mạng xã hội đến các hệ thống ngân hàng, chăm sóc sức khỏe. Các thông tin nhạy cảm như mật khẩu, số tài khoản ngân hàng, và hồ sơ y tế rất dễ trở thành mục tiêu của các cuộc tấn công mạng.

Mã hóa đảm bảo rằng, ngay cả khi dữ liệu bị truy cập trái phép, nội dung vẫn không thể được đọc hoặc sử dụng nếu không có khóa giải mã. Ví dụ:

##### Các ứng dụng nhắn tin như Signal và WhatsApp sử dụng mã hóa đầu cuối (end-to-end encryption), đảm bảo rằng chỉ người gửi và người nhận mới có thể đọc được nội dung tin nhắn.

##### Dữ liệu y tế nhạy cảm được mã hóa trên các hệ thống lưu trữ để bảo vệ quyền riêng tư của bệnh nhân.

Bằng cách bảo vệ thông tin cá nhân, mã hóa góp phần tạo ra môi trường số an toàn, nơi người dùng có thể tin tưởng và tự do sử dụng các dịch vụ trực tuyến mà không lo ngại bị xâm phạm quyền riêng tư.

#### ****Đảm bảo an toàn cho giao dịch trực tuyến****

Trong thời đại mà thương mại điện tử và ngân hàng trực tuyến trở nên phổ biến, mã hóa đóng vai trò quan trọng trong việc bảo vệ dữ liệu tài chính và giao dịch.

Các giao thức mã hóa như HTTPS, TLS/SSL không chỉ đảm bảo an toàn trong quá trình truyền tải dữ liệu mà còn ngăn chặn các hình thức tấn công phổ biến như:

##### **Tấn công trung gian (Man-in-the-Middle):** Khi hacker can thiệp vào giao tiếp giữa hai bên.

##### **Đánh cắp thông tin:** Khi dữ liệu tài chính hoặc thông tin thẻ tín dụng bị lộ.

Ví dụ, khi người dùng thực hiện giao dịch mua sắm trực tuyến, dữ liệu thẻ tín dụng được mã hóa trước khi truyền tới máy chủ để đảm bảo rằng thông tin không thể bị đọc trộm trong quá trình gửi. Nhờ mã hóa, các giao dịch trực tuyến trở nên an toàn hơn, xây dựng lòng tin và khuyến khích người dùng tận dụng các dịch vụ trực tuyến.

#### ****Bảo vệ cơ sở hạ tầng quan trọng****

Các cơ sở hạ tầng quan trọng như tài chính, năng lượng, giao thông, và viễn thông phụ thuộc rất lớn vào dữ liệu để vận hành. Một cuộc tấn công vào các hệ thống này có thể gây ra hậu quả nghiêm trọng, ảnh hưởng không chỉ đến tổ chức mà còn đến an ninh quốc gia.

Mã hóa là lớp bảo vệ quan trọng giúp ngăn chặn các cuộc tấn công mạng vào các cơ sở hạ tầng này. Ví dụ:

##### Trong hệ thống tài chính, mã hóa dữ liệu giao dịch ngăn chặn việc lộ thông tin quan trọng như các chiến lược đầu tư hoặc dữ liệu giao dịch giữa các ngân hàng.

##### Các mạng lưới năng lượng sử dụng giao thức mã hóa để bảo vệ hệ thống điều khiển khỏi các cuộc tấn công nhằm gây gián đoạn hoặc chiếm quyền kiểm soát.

Việc áp dụng mã hóa trong các hệ thống này không chỉ bảo vệ dữ liệu mà còn đảm bảo tính ổn định và an toàn trong hoạt động.

#### ****Đối phó với các cuộc tấn công mạng ngày càng tinh vi****

Các hình thức tấn công mạng như ransomware, nghe lén, hoặc đánh cắp dữ liệu ngày càng tinh vi, gây ra những thiệt hại nghiêm trọng cho cả cá nhân và tổ chức. Mã hóa là một giải pháp quan trọng giúp giảm thiểu rủi ro từ các cuộc tấn công này.

Ví dụ:

##### Khi hacker tấn công hệ thống và đánh cắp dữ liệu, nếu dữ liệu đã được mã hóa, hacker sẽ không thể sử dụng thông tin này nếu không có khóa giải mã.

##### Trong trường hợp ransomware, dữ liệu quan trọng được mã hóa từ trước có thể được khôi phục dễ dàng hơn, ngay cả khi tin tặc cố gắng mã hóa trái phép.

Nhờ đó, mã hóa không chỉ là một biện pháp bảo vệ thụ động mà còn là một công cụ hiệu quả giúp đối phó với các mối đe dọa ngày càng tinh vi trong môi trường số.

#### ****Đáp ứng tiêu chuẩn pháp lý và thúc đẩy công nghệ tương lai****

Nhiều quy định và tiêu chuẩn pháp lý hiện nay yêu cầu tổ chức và doanh nghiệp phải bảo vệ dữ liệu của khách hàng bằng các phương pháp mã hóa. Các quy định như GDPR (General Data Protection Regulation) của Châu Âu, HIPAA (Health Insurance Portability and Accountability Act) tại Mỹ, hay PCI DSS (Payment Card Industry Data Security Standard) đều nhấn mạnh vai trò của mã hóa trong việc bảo vệ dữ liệu.

Ngoài ra, mã hóa còn là nền tảng cho nhiều công nghệ hiện đại:

##### **Blockchain:** Mã hóa đảm bảo tính toàn vẹn và bảo mật của các giao dịch trên chuỗi khối.

##### **IoT:** Các thiết bị thông minh kết nối mạng sử dụng mã hóa để bảo vệ dữ liệu khỏi việc bị can thiệp hoặc truy cập trái phép.

##### **Tiền mã hóa:** Các thuật toán mã hóa phức tạp như SHA-256 đảm bảo sự bảo mật và minh bạch trong các giao dịch tiền điện tử.

Mã hóa không chỉ giúp đáp ứng các yêu cầu pháp lý mà còn mở ra cơ hội để phát triển các công nghệ tiên tiến, tạo ra nền tảng an toàn và đáng tin cậy cho tương lai.

Mã hóa đóng vai trò quan trọng không chỉ trong bảo vệ dữ liệu cá nhân và giao dịch trực tuyến, mà còn giúp duy trì tính ổn định và an toàn của các cơ sở hạ tầng quan trọng. Đồng thời, mã hóa góp phần đối phó với các cuộc tấn công mạng ngày càng phức tạp và thúc đẩy sự phát triển của các công nghệ mới. Việc hiểu và ứng dụng mã hóa một cách hiệu quả không chỉ là yêu cầu, mà còn là điều kiện tiên quyết để bảo vệ thông tin trong thời đại số.

### Ứng dụng của mã hóa trong thời đại số hiện nay

Mã hóa là một yếu tố không thể thiếu trong bảo mật dữ liệu trong thời đại số, đóng vai trò quan trọng trong việc bảo vệ thông tin và đảm bảo tính toàn vẹn của các hệ thống. Cùng với sự phát triển của công nghệ, mã hóa đã được áp dụng rộng rãi trong nhiều lĩnh vực, từ giao dịch ngân hàng trực tuyến đến các ứng dụng trong Internet of Things (IoT) và lưu trữ đám mây. Dưới đây là một số ứng dụng phổ biến của mã hóa trong đời sống hiện đại:

#### ****Giao dịch ngân hàng trực tuyến****

Giao dịch ngân hàng trực tuyến là một trong những lĩnh vực yêu cầu mức độ bảo mật cao nhất. Mã hóa được sử dụng để bảo vệ các thông tin nhạy cảm của khách hàng, bao gồm tài khoản ngân hàng, mật khẩu, và thông tin thẻ tín dụng. Mỗi khi người dùng thực hiện một giao dịch, dữ liệu được mã hóa trước khi gửi qua mạng, giúp ngăn chặn việc bị nghe lén hoặc đánh cắp thông tin trong quá trình truyền tải. Các giao thức mã hóa như HTTPS (SSL/TLS) đảm bảo rằng thông tin giữa người dùng và ngân hàng được bảo vệ một cách an toàn, tránh bị tấn công bởi các hacker.

Ví dụ: Khi thực hiện thanh toán trực tuyến, thông tin thẻ tín dụng của người dùng sẽ được mã hóa trước khi gửi đi và chỉ có ngân hàng nhận được thông tin này có thể giải mã và xử lý giao dịch.

#### ****Internet of Things (IoT)****

IoT đề cập đến mạng lưới các thiết bị thông minh kết nối internet, từ máy lạnh, thiết bị gia dụng đến các hệ thống giám sát an ninh. Mặc dù IoT mang lại nhiều tiện ích trong cuộc sống, nhưng cũng tạo ra nhiều rủi ro bảo mật. Các thiết bị IoT thường xuyên thu thập và truyền tải dữ liệu nhạy cảm như thói quen sử dụng, vị trí và các thông tin cá nhân khác. Vì vậy, mã hóa đóng vai trò quan trọng trong việc bảo vệ dữ liệu này khỏi các cuộc tấn công mạng.

Mã hóa giúp bảo vệ các dữ liệu mà thiết bị IoT thu thập, đồng thời bảo vệ các giao tiếp giữa các thiết bị và các nền tảng đám mây khỏi việc bị can thiệp. Ví dụ: Một chiếc camera an ninh IoT sẽ mã hóa video và dữ liệu cảm biến trước khi gửi đến ứng dụng trên điện thoại của người dùng, đảm bảo rằng thông tin không thể bị xem trộm.

#### ****Lưu trữ và xử lý dữ liệu trên đám mây****

Với sự phát triển mạnh mẽ của các dịch vụ đám mây, việc lưu trữ và xử lý dữ liệu trực tuyến đã trở thành xu hướng chủ đạo. Tuy nhiên, việc lưu trữ dữ liệu trên đám mây cũng tiềm ẩn rủi ro lớn về bảo mật, đặc biệt là khi dữ liệu có thể được truy cập từ nhiều địa điểm khác nhau. Mã hóa giúp đảm bảo rằng dữ liệu được bảo vệ trong suốt quá trình truyền tải và khi lưu trữ.

Các dịch vụ đám mây như Google Drive, Dropbox hay iCloud đều sử dụng mã hóa để bảo vệ dữ liệu của người dùng. Dữ liệu được mã hóa trên máy của người dùng trước khi tải lên đám mây và chỉ có người dùng mới có thể giải mã khi cần truy cập. Điều này đảm bảo rằng ngay cả khi dữ liệu bị xâm nhập, chúng cũng không thể đọc được mà không có khóa giải mã.

#### ****An ninh mạng và bảo mật thông tin trong doanh nghiệp****

Mã hóa là công cụ quan trọng trong việc bảo vệ thông tin nhạy cảm của doanh nghiệp, bao gồm dữ liệu khách hàng, hợp đồng và các tài liệu nội bộ. Các phương pháp mã hóa như AES (Advanced Encryption Standard) và RSA được sử dụng để bảo vệ dữ liệu trong các hệ thống mạng nội bộ của công ty. Điều này giúp ngăn chặn việc truy cập trái phép vào các thông tin quan trọng và bảo vệ dữ liệu khỏi các mối đe dọa từ bên ngoài, như tấn công mạng hoặc xâm nhập hệ thống.

Các tổ chức lớn như ngân hàng, tổ chức tài chính hay các công ty công nghệ sử dụng mã hóa không chỉ để bảo vệ thông tin trong quá trình truyền tải mà còn khi lưu trữ trên máy chủ hoặc trong cơ sở dữ liệu.

#### ****Bảo mật trong giao tiếp qua email và tin nhắn****

Mã hóa cũng đóng vai trò quan trọng trong việc bảo vệ các thông tin trong quá trình giao tiếp qua email và tin nhắn. Việc mã hóa email giúp bảo vệ nội dung khỏi việc bị đọc trộm bởi các hacker hoặc đối tượng không có thẩm quyền. Các dịch vụ email như ProtonMail hay các hệ thống email doanh nghiệp thường sử dụng mã hóa đầu cuối (end-to-end encryption) để đảm bảo rằng chỉ người gửi và người nhận mới có thể đọc được nội dung thư.

Tương tự, mã hóa cũng được áp dụng trong các ứng dụng nhắn tin như WhatsApp và Signal, giúp bảo vệ thông tin cá nhân của người dùng khỏi việc bị nghe lén hoặc theo dõi.

## Phân loại các mã hóa

### Mã hóa đối xứng

Mã hóa đối xứng (hay còn gọi là mã hóa khóa bí mật) là phương pháp mã hóa trong đó cùng một khóa được sử dụng để mã hóa và giải mã dữ liệu. Nghĩa là, người gửi và người nhận thông tin phải sử dụng cùng một khóa bí mật để đảm bảo tính bảo mật của thông tin truyền tải. Đây là phương pháp mã hóa được sử dụng phổ biến nhất trong các hệ thống bảo mật dữ liệu vì tính đơn giản và hiệu quả trong việc xử lý một lượng lớn dữ liệu.

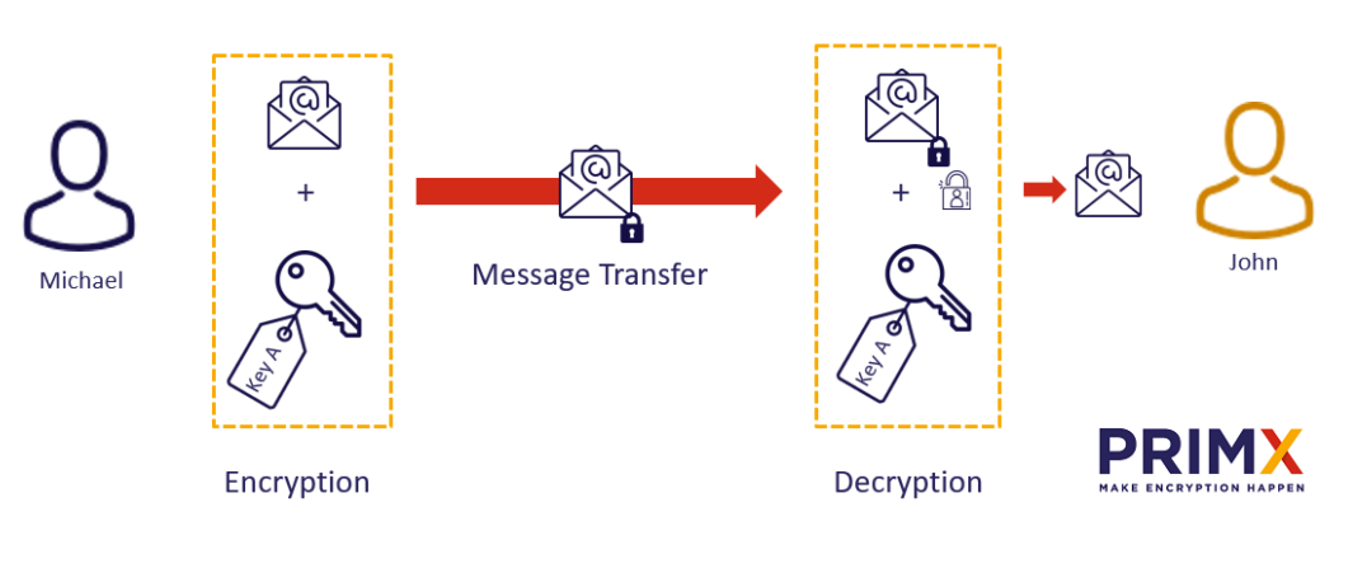
#### ****Nguyên lý hoạt động của mã hóa đối xứng****

Trong mã hóa đối xứng, quá trình mã hóa và giải mã diễn ra theo những bước cơ bản sau:

##### **Mã hóa:** Người gửi sử dụng khóa bí mật (key) để chuyển đổi dữ liệu gốc (plaintext) thành dữ liệu đã mã hóa (ciphertext).

##### **Truyền tải dữ liệu:** Dữ liệu đã mã hóa được truyền tải qua kênh thông tin (có thể là internet, mạng cục bộ, hay các phương tiện truyền tải khác).

##### **Giải mã:** Người nhận sử dụng cùng một khóa bí mật để giải mã dữ liệu đã mã hóa và phục hồi lại thông tin gốc (plaintext).



Hình 2. Truyền thông tin bằng mã hóa đối xứng

Điều quan trọng trong phương pháp mã hóa đối xứng là khóa bí mật phải được bảo vệ cẩn thận trong suốt quá trình truyền tải. Nếu khóa bị lộ, kẻ tấn công có thể dễ dàng giải mã và tiếp cận thông tin nhạy cảm.

#### ****Ưu điểm của mã hóa đối xứng****

**Hiệu suất cao:** Mã hóa đối xứng thường nhanh hơn so với các phương pháp mã hóa bất đối xứng vì thuật toán mã hóa đơn giản và không yêu cầu quá trình tính toán phức tạp.

**Đơn giản trong triển khai:** Việc sử dụng một khóa duy nhất để mã hóa và giải mã giúp giảm thiểu độ phức tạp trong việc triển khai và quản lý hệ thống mã hóa.

**Tiết kiệm tài nguyên:** Mã hóa đối xứng ít tiêu tốn tài nguyên tính toán và băng thông hơn so với các phương pháp mã hóa khác, đặc biệt là trong các hệ thống có khối lượng dữ liệu lớn.

#### ****Nhược điểm của mã hóa đối xứng****

**Vấn đề phân phối khóa:** Vấn đề lớn nhất trong mã hóa đối xứng là cách thức chia sẻ khóa bí mật giữa người gửi và người nhận mà không bị rò rỉ. Nếu khóa bị lộ trong quá trình truyền tải, toàn bộ hệ thống bảo mật sẽ bị phá vỡ.

**Khó khăn trong việc mở rộng:** Trong trường hợp nhiều người tham gia vào việc truyền tải và nhận thông tin, việc quản lý và phân phối khóa bí mật trở nên khó khăn. Mỗi cặp người gửi - người nhận cần phải có một khóa riêng biệt, điều này làm tăng số lượng khóa cần quản lý và bảo vệ.

#### ****Ví dụ về các thuật toán mã hóa đối xứng****

**AES (Advanced Encryption Standard):** AES là một trong những thuật toán mã hóa đối xứng phổ biến nhất, được sử dụng trong các giao thức bảo mật như HTTPS, VPN, và các hệ thống lưu trữ dữ liệu an toàn. AES sử dụng các kích thước khóa 128-bit, 192-bit, hoặc 256-bit, với khả năng bảo mật cao và hiệu suất tốt.

**DES (Data Encryption Standard):** DES là một thuật toán mã hóa cũ, từng được sử dụng rộng rãi trong các hệ thống bảo mật, nhưng hiện nay đã được thay thế bởi AES do sự phát hiện của các lỗ hổng bảo mật. DES sử dụng một khóa 56-bit và đã không còn đủ mạnh để bảo vệ thông tin trước các tấn công hiện đại.

**3DES (Triple DES):** 3DES là một phương pháp mã hóa nâng cấp từ DES, sử dụng ba lượt mã hóa và giải mã liên tiếp với ba khóa khác nhau. Mặc dù tốt hơn DES, nhưng 3DES vẫn không thể so sánh với AES về mặt bảo mật và hiệu suất.

#### ****Ứng dụng của mã hóa đối xứng****

Bảo mật giao dịch ngân hàng: Mã hóa đối xứng được sử dụng trong nhiều giao dịch trực tuyến để bảo vệ thông tin tài chính của khách hàng.

**Mã hóa dữ liệu trong VPN:** Các kết nối VPN thường sử dụng mã hóa đối xứng để bảo vệ dữ liệu truyền tải giữa người dùng và mạng nội bộ.

**Bảo mật email:** Mã hóa đối xứng có thể được sử dụng để bảo vệ nội dung của email trong các hệ thống bảo mật thông tin.

Tóm lại, mã hóa đối xứng là một phương pháp mã hóa mạnh mẽ và hiệu quả trong việc bảo vệ thông tin. Mặc dù có một số thách thức về việc quản lý và phân phối khóa, nhưng với sự phát triển của các kỹ thuật và hệ thống phân phối khóa an toàn (như trong các giao thức như SSL/TLS), mã hóa đối xứng vẫn giữ vai trò quan trọng trong bảo mật thông tin ngày nay.

### Mã hóa bất đối xứng

Mã hóa bất đối xứng (hay còn gọi là mã hóa khóa công khai) là phương pháp mã hóa trong đó hai khóa khác nhau được sử dụng: một khóa công khai (public key) và một khóa riêng (private key). Điều đặc biệt của phương pháp này là người gửi có thể sử dụng khóa công khai của người nhận để mã hóa thông tin, trong khi người nhận sẽ sử dụng khóa riêng của mình để giải mã thông tin đó. Quá trình này cho phép các bên giao tiếp an toàn mà không cần phải trao đổi khóa bí mật trước.

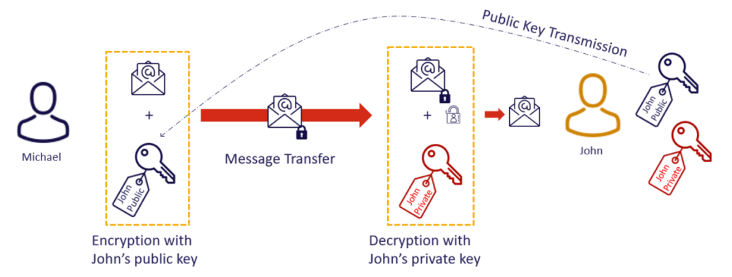
#### ****Nguyên lý hoạt động của mã hóa bất đối xứng****

Mã hóa bất đối xứng sử dụng hai khóa liên kết với nhau, nhưng không thể tính toán được khóa riêng từ khóa công khai và ngược lại. Quy trình mã hóa và giải mã diễn ra như sau:

##### **Mã hóa:** Người gửi sử dụng khóa công khai của người nhận để mã hóa thông tin. Khóa công khai có thể được chia sẻ công khai mà không lo ngại về sự xâm nhập của kẻ tấn công vì chỉ có khóa riêng của người nhận mới có thể giải mã được.

##### **Truyền tải dữ liệu:** Dữ liệu đã mã hóa được truyền qua kênh thông tin.

##### **Giải mã:** Người nhận sử dụng khóa riêng của mình để giải mã thông tin, khôi phục lại dữ liệu gốc.



Hình 3. Truyền thông tin bằng mã hóa bất đối xứng

Điều này có nghĩa là chỉ có người nhận sở hữu khóa riêng mới có thể giải mã thông tin mà không cần phải trao đổi khóa bí mật với người gửi trước đó.

#### ****Ưu điểm của mã hóa bất đối xứng****

**Không cần chia sẻ khóa bí mật:** Một trong những ưu điểm lớn nhất của mã hóa bất đối xứng là không cần phải chia sẻ khóa bí mật giữa người gửi và người nhận. Chỉ cần sử dụng khóa công khai để mã hóa, và khóa riêng để giải mã, giúp bảo mật thông tin ngay cả khi các bên giao tiếp qua mạng không an toàn.

**An toàn hơn trong môi trường mở:** Với việc sử dụng khóa công khai và khóa riêng, mã hóa bất đối xứng cung cấp một phương thức an toàn để truyền tải thông tin trong các môi trường không đáng tin cậy, chẳng hạn như Internet.

**Phù hợp với chữ ký số và xác thực:** Mã hóa bất đối xứng không chỉ được dùng để mã hóa thông tin mà còn có thể sử dụng để tạo chữ ký số, xác nhận danh tính người gửi và đảm bảo tính toàn vẹn của thông tin.

#### ****Nhược điểm của mã hóa bất đối xứng****

**Tốc độ chậm hơn:** So với mã hóa đối xứng, mã hóa bất đối xứng thường chậm hơn nhiều vì các thuật toán mã hóa và giải mã phức tạp hơn. Điều này có thể làm giảm hiệu suất khi cần mã hóa và giải mã một lượng lớn dữ liệu.

**Yêu cầu về tài nguyên tính toán:** Các thuật toán mã hóa bất đối xứng yêu cầu tài nguyên tính toán cao hơn, điều này có thể làm tăng độ trễ và tiêu tốn tài nguyên hệ thống, đặc biệt khi hệ thống phải xử lý nhiều giao dịch.

#### ****Ví dụ về các thuật toán mã hóa bất đối xứng****

**RSA (Rivest-Shamir-Adleman):** RSA là một trong những thuật toán mã hóa bất đối xứng phổ biến nhất và được sử dụng rộng rãi trong các hệ thống bảo mật như SSL/TLS, giao dịch ngân hàng trực tuyến, và chữ ký số. RSA sử dụng hai khóa: một khóa công khai để mã hóa và một khóa riêng để giải mã. An toàn của RSA dựa trên độ khó của bài toán phân tích số lớn thành thừa số nguyên tố.

**ECC (Elliptic Curve Cryptography):** ECC là một thuật toán mã hóa bất đối xứng sử dụng các đường cong elip. ECC được ưa chuộng vì tính bảo mật cao với kích thước khóa nhỏ, giúp tiết kiệm băng thông và tăng hiệu suất. ECC đang được sử dụng trong nhiều ứng dụng bảo mật hiện đại, đặc biệt là trong các thiết bị di động và hệ thống IoT.

**DSA (Digital Signature Algorithm):** DSA là thuật toán sử dụng trong việc tạo và xác nhận chữ ký số. Mặc dù không được sử dụng để mã hóa thông tin, DSA đóng một vai trò quan trọng trong việc đảm bảo tính toàn vẹn và xác thực của thông tin trong các giao dịch số.

#### ****Ứng dụng của mã hóa bất đối xứng****

**Bảo mật giao dịch trực tuyến:** Mã hóa bất đối xứng là nền tảng của các giao thức bảo mật như SSL/TLS, giúp bảo vệ thông tin cá nhân và giao dịch trực tuyến trong môi trường Internet. Khóa công khai được sử dụng để thiết lập kết nối an toàn, trong khi khóa riêng đảm bảo rằng chỉ người nhận mới có thể giải mã thông tin.

##### **Chữ ký số:** Mã hóa bất đối xứng là công nghệ cơ bản để tạo ra chữ ký số, giúp xác minh danh tính người gửi và đảm bảo rằng thông tin không bị thay đổi trong quá trình truyền tải.

##### **Xác thực và bảo mật trong giao tiếp:** Mã hóa bất đối xứng cũng được sử dụng để xác thực người dùng trong các hệ thống mạng, như trong việc đăng nhập vào tài khoản ngân hàng, email, và các hệ thống cần đảm bảo tính xác thực cao.

##### **Bảo mật trong IoT:** Các thiết bị IoT (Internet of Things) sử dụng mã hóa bất đối xứng để bảo vệ dữ liệu truyền tải giữa các thiết bị và máy chủ, đảm bảo rằng thông tin không bị can thiệp hay thay đổi trong quá trình giao tiếp.

Tóm lại, mã hóa bất đối xứng cung cấp một giải pháp bảo mật mạnh mẽ cho việc truyền tải thông tin trong môi trường không an toàn, đặc biệt là trên Internet. Dù có một số nhược điểm về tốc độ và yêu cầu tài nguyên tính toán, nhưng nhờ vào khả năng bảo vệ dữ liệu mà không cần chia sẻ khóa bí mật, mã hóa bất đối xứng đã trở thành một phần không thể thiếu trong các hệ thống bảo mật hiện đại, từ việc bảo vệ giao dịch ngân hàng cho đến xác thực trong các ứng dụng di động và IoT.

### Các kỹ thuật bổ trợ trong bảo mật

Trong môi trường bảo mật hiện đại, ngoài các thuật toán mã hóa đối xứng và bất đối xứng, còn có một số kỹ thuật bổ trợ rất quan trọng giúp tăng cường bảo mật và đảm bảo tính toàn vẹn của thông tin. Trong đó, **Hashing** và **Chữ ký số (Digital Signature)** là hai kỹ thuật chủ yếu. Những công nghệ này không chỉ đóng vai trò quan trọng trong việc bảo vệ dữ liệu mà còn giúp xác thực danh tính, bảo vệ thông tin trong các giao dịch trực tuyến, và cung cấp phương pháp kiểm tra tính toàn vẹn của dữ liệu mà không cần tiết lộ chính nội dung của nó.

#### ****Hashing****

**Hashing** là một kỹ thuật được sử dụng để chuyển đổi bất kỳ lượng dữ liệu nào thành một giá trị có độ dài cố định, gọi là "hash value" (giá trị băm) hay "message digest". Giá trị này không thể đảo ngược, có nghĩa là không thể lấy lại thông tin gốc từ giá trị băm, giúp bảo vệ sự riêng tư của dữ liệu.

**Nguyên lý hoạt động của Hashing:**

Khi một thông điệp hoặc dữ liệu đầu vào được đưa vào một hàm băm (hash function), hàm này sẽ tạo ra một giá trị băm duy nhất. Mỗi thay đổi nhỏ trong dữ liệu đầu vào sẽ dẫn đến một giá trị băm hoàn toàn khác biệt, từ đó giúp phát hiện sự thay đổi của dữ liệu. Mặc dù hàm băm tạo ra một chuỗi ký tự có độ dài cố định (chẳng hạn 256 bit cho SHA-256), nhưng giá trị này là đại diện duy nhất cho toàn bộ dữ liệu đầu vào.

**Ứng dụng của Hashing:**

**Kiểm tra tính toàn vẹn:** Hashing là phương pháp hiệu quả để kiểm tra xem dữ liệu có bị thay đổi trong quá trình truyền tải hay không. Nếu một giá trị băm ban đầu của dữ liệu được gửi cùng với dữ liệu, người nhận có thể tính toán lại giá trị băm và so sánh với giá trị băm đã gửi. Nếu các giá trị trùng khớp, dữ liệu là nguyên vẹn.

**Lưu trữ mật khẩu:** Mật khẩu của người dùng thường được lưu trữ dưới dạng giá trị băm thay vì dạng văn bản rõ. Khi người dùng nhập mật khẩu, hệ thống sẽ băm giá trị mật khẩu và so sánh với giá trị băm đã lưu trữ. Điều này làm tăng độ bảo mật, vì ngay cả khi cơ sở dữ liệu bị tấn công, các mật khẩu không thể bị lộ.

**Tạo chỉ mục tìm kiếm nhanh:** Các cơ sở dữ liệu sử dụng kỹ thuật hashing để tạo bảng băm, giúp việc tìm kiếm dữ liệu trong các bộ dữ liệu lớn trở nên nhanh chóng và hiệu quả.

**Các thuật toán băm phổ biến:**

**MD5:** Một thuật toán băm cũ, thường được sử dụng để kiểm tra tính toàn vẹn của dữ liệu, nhưng hiện nay đã bị coi là không an toàn vì khả năng va chạm (collision).

**SHA-1:** SHA-1 trước đây được sử dụng rộng rãi trong các ứng dụng bảo mật, nhưng hiện tại đã không còn an toàn trong nhiều ứng dụng do các lỗ hổng bảo mật.

**SHA-256:** Một phần của gia đình SHA-2, SHA-256 được coi là an toàn và được sử dụng trong các giao thức bảo mật hiện đại như SSL/TLS và blockchain.

#### ****Chữ ký số****

**Chữ ký số** là một ứng dụng của mã hóa bất đối xứng, sử dụng cặp khóa công khai và khóa riêng để đảm bảo tính xác thực của dữ liệu và xác nhận danh tính của người gửi. Chữ ký số giúp người nhận xác minh rằng thông điệp hoặc tài liệu không bị thay đổi kể từ khi được ký và xác định người gửi có phải là người thực sự đã ký hay không.

**Nguyên lý hoạt động của Chữ ký số**

**Tạo chữ ký:** Người gửi sẽ sử dụng khóa riêng của mình để mã hóa giá trị băm của thông điệp. Kết quả của quá trình này chính là chữ ký số. Thông điệp cùng với chữ ký số sẽ được gửi đi cho người nhận.

**Xác minh chữ ký:** Người nhận sử dụng khóa công khai của người gửi để giải mã chữ ký số và lấy lại giá trị băm. Sau đó, họ tính lại giá trị băm từ thông điệp nhận được và so sánh với giá trị băm đã giải mã. Nếu hai giá trị băm khớp nhau, thì thông điệp được xác nhận là nguyên vẹn và người gửi là chính xác.

**Ứng dụng của Chữ ký số:**

**Xác thực danh tính:** Chữ ký số là một công cụ mạnh mẽ để xác thực danh tính của người gửi trong các giao dịch điện tử, chẳng hạn như ký kết hợp đồng điện tử, giao dịch ngân hàng trực tuyến, hay chứng thực các tài liệu quan trọng.

**Bảo vệ tính toàn vẹn của dữ liệu:** Chữ ký số giúp đảm bảo rằng nội dung của tài liệu không bị thay đổi sau khi được ký, từ đó bảo vệ dữ liệu khỏi các hành vi gian lận hoặc sửa đổi trái phép.

**Chứng thực pháp lý:** Chữ ký số có giá trị pháp lý, có thể được sử dụng để thay thế chữ ký tay trong các hợp đồng, giao dịch pháp lý, và các tài liệu cần chứng thực pháp lý.

**Các thuật toán chữ ký số phổ biến:**

**RSA:** Thuật toán RSA là một trong những thuật toán mã hóa và chữ ký số phổ biến nhất, sử dụng cặp khóa công khai và khóa riêng để mã hóa và giải mã dữ liệu.

**DSA (Digital Signature Algorithm):** Thuật toán này được thiết kế đặc biệt để tạo và xác thực chữ ký số. DSA sử dụng các phép toán số học trên các số nguyên lớn và có hiệu suất tốt trong các ứng dụng chữ ký số.

**ECDSA (Elliptic Curve Digital Signature Algorithm):** ECDSA là một thuật toán chữ ký số sử dụng các đường cong elip, giúp giảm kích thước khóa và tăng tốc độ xác minh chữ ký, đồng thời cung cấp độ bảo mật cao với kích thước khóa nhỏ.

Tóm lại, các công nghệ bảo mật bổ trợ như **Hashing** và **Chữ ký số** là những công cụ không thể thiếu trong hệ thống bảo mật hiện đại. Hashing giúp bảo vệ tính toàn vẹn của dữ liệu, trong khi chữ ký số cung cấp một phương thức đáng tin cậy để xác thực danh tính và bảo vệ dữ liệu khỏi sự thay đổi trái phép. Những công nghệ này đang ngày càng trở thành một phần quan trọng trong việc bảo mật thông tin trong các ứng dụng từ giao dịch trực tuyến cho đến lưu trữ và trao đổi dữ liệu an toàn.

# PHÂN TÍCH CÁC GIẢI THUẬT MÃ HÓA

## Lựa chọn các giải thuật mã hóa tiêu biểu

Việc chọn các giải thuật dưới đây nhằm mục tiêu cung cấp cái nhìn toàn diện về lĩnh vực mật mã học, từ các phương pháp cổ điển đến hiện đại, từ mã hóa đối xứng, bất đối xứng đến hàm băm mật mã. Điều này giúp nghiên cứu đạt được sự đa dạng và độ sâu cần thiết để giải thích và so sánh các phương pháp mã hóa trong các kịch bản ứng dụng khác nhau.

### Ceasar Cipher

Ceasar Cipher là một trong những thuật toán mã hóa cổ điển đơn giản nhất, giúp người đọc dễ dàng hiểu được các khái niệm cơ bản của mật mã học, như khóa mã hóa, dịch chuyển và tính chất tuần hoàn của bảng chữ cái.

Dù không còn sử dụng rộng rãi vì không đảm bảo an toàn trước các tấn công hiện đại, Ceasar Cipher vẫn là nền tảng để so sánh với các thuật toán mã hóa tiên tiến hơn như AES và RSA.

### AES (Advanced Encryption Standard)

AES là thuật toán mã hóa đối xứng được sử dụng rộng rãi trong thực tế nhờ tính bảo mật cao, tốc độ nhanh và khả năng triển khai trên nhiều nền tảng phần cứng, phần mềm. Đây là tiêu chuẩn mã hóa được NIST (Viện Tiêu chuẩn và Công nghệ Hoa Kỳ) công nhận.

AES đại diện cho các thuật toán mã hóa hiện đại trong lĩnh vực bảo mật thông tin, phù hợp để nghiên cứu các ứng dụng thực tiễn, chẳng hạn như bảo mật dữ liệu trong mạng Wi-Fi, VPN và ngân hàng.

### RSA (Rivest–Shamir–Adleman)

RSA là một trong những thuật toán mã hóa bất đối xứng đầu tiên và phổ biến nhất. Thuật toán này dựa trên bài toán phân tích thừa số nguyên tố – một vấn đề khó giải quyết trong toán học, đảm bảo tính bảo mật cao.

RSA minh họa cách mã hóa bất đối xứng hoạt động, nổi bật ở các ứng dụng thực tế như giao dịch thương mại điện tử, chứng thực số (digital signature) và mã hóa email.

### SHA-256 (Secure Hash Algorithm 256-bit)

SHA-256 là hàm băm mật mã, được sử dụng rộng rãi để đảm bảo tính toàn vẹn dữ liệu. Đây là một phần trong bộ tiêu chuẩn SHA-2 của NIST.

Việc nghiên cứu SHA-256 giúp hiểu rõ hơn về cách đảm bảo tính toàn vẹn của thông tin, ví dụ như xác minh chữ ký số, lưu trữ mật khẩu an toàn, và ứng dụng trong blockchain (ví dụ: Bitcoin).

## Giải thuật Ceasar Cipher

### ****Tổng quan về giải thuật Caesar Cipher****

Caesar Cipher, hay còn gọi là Caesar Shift, là một phương pháp mã hóa cổ điển được phát triển từ thời La Mã cổ đại. Giải thuật này thuộc nhóm mã hóa thay thế (substitution cipher), trong đó mỗi ký tự trong văn bản gốc (plaintext) được thay thế bằng một ký tự khác thông qua phép dịch chuyển có hệ thống trong bảng chữ cái.

Tên của giải thuật được đặt theo Julius Caesar, nhà lãnh đạo nổi tiếng của La Mã, người sử dụng phương pháp này để mã hóa thông điệp quân sự. Mục đích của Caesar Cipher là che giấu nội dung thông điệp, giúp bảo vệ các thông tin nhạy cảm khỏi những kẻ không được phép tiếp cận.

### ****Nguyên lý hoạt động****

#### Mã hóa

Giải thuật Caesar Cipher hoạt động dựa trên phép dịch chuyển ký tự trong bảng chữ cái. Mỗi ký tự trong văn bản gốc được thay thế bằng một ký tự khác cách nó một khoảng cố định, được gọi là **khóa dịch chuyển (shift key).**

Với mỗi ký tự P trong plaintext, ký tự tương ứng C trong ciphertext được tính bằng **công thức mã hóa** như sau:

|  |
| --- |
| = ( + k) mod  n |

Trong đó:

##### : Ký tự mã hóa ở vị trí thứ i trong văn bản mã hóa (ciphertext).

##### : Ký tự cần mã hóa ở vị trí thứ i trong văn bản gốc, được biểu diễn dưới dạng số.

##### k: Khóa dịch chuyển (shift key), là số bước dịch chuyển.

##### n: Số ký tự trong bảng chữ cái (thường là 26 đối với bảng chữ cái tiếng Anh).

Dưới đây là một ví dụ cụ thể:

Văn bản gốc: "HELLO"

Khóa k: 3 (được chọn ngẫu nhiên)

Mỗi ký tự Pi trong "HELLO" sẽ được mã hóa bằng công thức: Ci = (Pi + 3) mod 26.)

H (7 + 3) mod 26 = 10 (ký tự "K")  
E (4 + 3) mod 26 = 7 (ký tự "H")  
L (11 + 3) mod 26 = 14 (ký tự "O")  
L (11 + 3) mod 26 = 14 (ký tự "O")  
O (14 + 3) mod 26 = 17 (ký tự "R")

Do đó, khi mã hóa "HELLO" với khóa 3, ta sẽ nhận được "KHOOQ" như là văn bản đã được mã hóa.

#### Giải mã

Để giải mã, ta thực hiện phép dịch chuyển ngược, **công thức giải mã như sau:**

|  |
| --- |
| = ( - k) mod  n |

Trong đó:

##### : Ký tự đã được mã hóa ở vị trí thứ i trong văn bản mã hóa.

##### : Ký tự gốc ở vị trí thứ i trong văn bản gốc, được biểu diễn dưới dạng số.

##### k: Khóa dịch chuyển (shift key), là số bước dịch chuyển.

##### n: Số ký tự trong bảng chữ cái (thường là 26 đối với bảng chữ cái tiếng Anh).

Dưới đây là một ví dụ cụ thể:

Văn bản mã hóa: "KHOOQ"

Khóa ki: 3 (giả sử ta biết khóa đã được sử dụng)

Mỗi ký tự Ci trong "KHOOQ" sẽ được giải mã bằng công thức Pi = (Ci - 3) mod 26.

K (10 - 3) mod 26 = 7 (ký tự "H")

H (7 - 3) mod 26 = 4 (ký tự "E")

O (14 - 3) mod 26 = 11 (ký tự "L")

O (14 - 3) mod 26 = 11 (ký tự "L")

Q (17 - 3) mod 26 = 14 (ký tự "O")

Do đó, khi ta giải mã "KHOOQ" với khóa 3, sẽ nhận được "HELLO" như là văn bản đã giải mã.

### Cấu trúc dữ liệu

#### Dữ liệu đầu vào (Input)

Plaintext: Một chuỗi ký tự cần mã hóa hoặc giải mã.

##### Kiểu dữ liệu: String.

Key (Shift): Số nguyên xác định mức độ dịch chuyển ký tự.

##### Kiểu dữ liệu: Integer.

#### Dữ liệu đầu ra (Output)

Ciphertext: Một chuỗi ký tự đã được mã hóa hoặc giải mã.

##### Kiểu dữ liệu: String.

#### Bảng chữ cái (Alphabet)

Tập hợp các ký tự có thể mã hóa, thông thường là bảng chữ cái.

##### Kiểu dữ liệu: Array hoặc List.

##### Ví dụ: ['A', 'B', 'C', ..., 'Z'].

#### Hàm ánh xạ vị trí

Ánh xạ từ ký tự sang vị trí và ngược lại trong bảng chữ cái.

Dùng Hash Table hoặc Dictionary để tối ưu truy cập:

##### {'A': 0, 'B': 1, ..., 'Z': 25}.

##### 0: 'A', 1: 'B', ..., 25: 'Z'.

#### Lưu trữ kết quả

Chuỗi mã hóa hoặc giải mã: Sử dụng cấu trúc String để ghép từng ký tự sau xử lý.

### Độ phức tạp của thuật toán

#### Độ phức tạp thời gian (Time Complexity)

Caesar Cipher xử lý từng ký tự trong chuỗi đầu vào và thực hiện phép tính dịch chuyển. Các bước chính:

Duyệt qua từng ký tự trong chuỗi đầu vào (Plaintext):

##### Duyệt qua n ký tự, với n là độ dài chuỗi.

##### Thời gian cho mỗi ký tự: kiểm tra và dịch chuyển (toán học modulo), thời gian là O(1).

Tổng cộng: Thời gian xử lý là O(n) với n là độ dài chuỗi.

Vậy độ phức tạp thời gian là O(n).

#### Độ phức tạp không gian (Space Complexity)

Các yếu tố ảnh hưởng đến bộ nhớ của thuật toán:

Bảng chữ cái (Alphabet):

##### Sử dụng cấu trúc List hoặc Hash Table để lưu bảng chữ cái.

##### Với bảng chữ cái tiếng Anh: 26 ký tự, không đổi.

##### Không gian là O(1).

Chuỗi kết quả (Ciphertext):

##### Tạo chuỗi mới có độ dài bằng với chuỗi đầu vào.

##### Chiếm không gian O(n), với n là độ dài chuỗi.

Biến tạm thời:

##### Một vài biến lưu chỉ số hoặc ký tự hiện tại. Chiếm không gian không đáng kể: O(1).

Vậy độ phức tạp không gian là O(n).

### ****Ưu điểm và hạn chế của Caesar Cipher****

#### Ưu điểm

Dù đơn giản, Caesar Cipher mang lại nhiều ưu điểm nhất định**:**

Dễ triển khai: Mã hóa Caesar không đòi hỏi nhiều tính toán phức tạp hoặc sử dụng bảng tham số phức tạp. Quá trình mã hóa và giải mã có thể được thực hiện dễ dàng bằng cách dịch chuyển các ký tự trong bảng chữ cái theo một khoảng cố định (khóa).

Dễ hiểu: Phương pháp này rất dễ hiểu và có thể được áp dụng mà không cần kiến thức chuyên sâu về toán học hay lý thuyết thông tin. Điều này làm cho nó trở thành một lựa chọn phổ biến trong việc truyền thông tin mật qua các kênh không đảm bảo.

Không yêu cầu lưu trữ khóa: Mã hóa Caesar không yêu cầu việc lưu trữ khóa phức tạp như một số hệ thống mã hóa khác. Khóa chỉ là một giá trị nguyên duy nhất, thường là một con số nhỏ.

#### Nhược điểm

Tuy nhiên, trong thời kỳ hiện đại, mã hóa Caesar không còn được coi là phương pháp mã hóa an toàn vì mã hóa Caesar có những nhược điểm lớn sau:

Dễ bị tấn công bằng phương pháp thử và lỗi (brute force): Với số lượng khóa có thể là một trong số 26 giá trị khác nhau, một tấn công brute force dễ dàng có thể thử mọi giá trị khóa để tìm ra khóa đúng. Trong khi một số kỹ thuật cải tiến như mã hóa đa chùm có thể giảm độ an toàn của tấn công brute force, mã hóa Caesar cơ bản vẫn dễ bị tấn công.

Khả năng chiến thuật tần suất ký tự (frequency analysis): Trong tiếng Anh, có sự phân bố không đều về tần suất xuất hiện của các ký tự trong văn bản. Bằng cách sử dụng phân tích tần suất ký tự, tấn công có thể suy luận được khóa và giải mã dữ liệu.

Sự tiện lợi của công nghệ hiện đại: Các máy tính bình thường có thể giải mã bằng thử brute force rất nhanh

Sự phổ biến của các mã hóa mạnh mẽ khác: Ngày nay, có nhiều phương pháp mã hóa khác an toàn hơn như AES (Advanced Encryption Standard), RSA, hoặc ECC (Elliptic Curve Cryptography),... Các phương pháp này được thiết kế để đối mặt với những thách thức và tấn công phức tạp hơn.

Vì lý do này, mã hóa Caesar hiện nay thường chỉ được sử dụng cho mục đích giáo dục, giải đố,...

### ****Ứng dụng của Caesar Cipher****

Mặc dù không còn được sử dụng trong các hệ thống bảo mật hiện đại do các hạn chế về độ an toàn, Caesar Cipher vẫn có một số ứng dụng như:

##### **Giáo dục:** Được sử dụng làm bài học nhập môn để dạy về mật mã học và các khái niệm cơ bản trong bảo mật.

##### **Giải trí:** Caesar Cipher thường được sử dụng trong các trò chơi giải mã hoặc câu đố.

##### **Bảo mật cấp thấp:** Trong một số trường hợp, Caesar Cipher có thể được sử dụng để che giấu thông tin không quan trọng.

### ****Cải tiến của Caesar Cipher****

Để khắc phục các hạn chế, các giải thuật mã hóa phức tạp hơn đã được phát triển từ Caesar Cipher, chẳng hạn như:

##### **Mã hóa Vigenère:** Sử dụng một khóa gồm nhiều ký tự để tạo ra sự dịch chuyển không cố định, khó phá vỡ hơn.

##### **Mã hóa Hill Cipher:** Dựa trên đại số tuyến tính để mã hóa khối ký tự thay vì từng ký tự riêng lẻ.

Tóm lại, Caesar Cipher dù đơn giản và dễ phá vỡ, vẫn là một mốc son quan trọng trong lịch sử phát triển mật mã. Nó mở ra những ý tưởng ban đầu về mã hóa thông tin và góp phần định hình các phương pháp bảo mật phức tạp hơn ngày nay.

## Giải thuật mã hóa AES

### ****Tổng quan về mã hóa AES****

#### Sự phát triển của chuẩn mã hóa AES

Các thuật toán mã hóa trước đây đều tồn tại những hạn chế nhất định. Các mã hóa cổ điển có thể dễ dàng bị phá vỡ bởi các hệ thống tính toán hiện đại. Thuật toán DES, ví dụ, đã bị phá vào năm 1998 bằng một hệ thống chỉ tốn khoảng 250.000 USD. Hơn nữa, DES được thiết kế cho phần cứng từ thập niên 1970 nên không tối ưu khi chạy trên phần mềm hiện đại. Triple DES, mặc dù cải thiện độ bảo mật bằng cách tăng gấp ba lần số vòng mã hóa, nhưng lại chậm hơn đáng kể và vẫn duy trì kích thước khối 64 bit – một hạn chế về mặt hiệu quả và bảo mật.

Do đó, cần một thuật toán mã hóa mới, kháng lại mọi hình thức tấn công đã biết. Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST) đã nhận thức rõ nhu cầu này và muốn thúc đẩy việc tạo ra một tiêu chuẩn mới. Tuy nhiên, do những tranh cãi xoay quanh thuật toán DES và sự can thiệp của một số cơ quan chính phủ Mỹ, NIST không thể trực tiếp tham gia thiết kế thuật toán. Thay vào đó, NIST tổ chức một cuộc thi toàn cầu vào ngày 2 tháng 1 năm 1997, kêu gọi các nhà nghiên cứu phát triển thuật toán mã hóa mới để bảo vệ thông tin nhạy cảm không thuộc phân loại của chính phủ Mỹ.

Các thuật toán tham gia cuộc thi phải đáp ứng nhiều yêu cầu nghiêm ngặt và thiết kế phải được công khai đầy đủ – điều mà DES không làm được. Sau khi các thuật toán được nộp, quá trình đánh giá kéo dài nhiều năm thông qua các hội thảo mật mã học. Trong vòng đầu tiên, 15 thuật toán được chọn, sau đó rút gọn còn 5 thuật toán ở vòng hai. Các thuật toán này được kiểm tra hiệu suất và độ bảo mật bởi các chuyên gia mật mã hàng đầu thế giới và NIST.

Cuối cùng, thuật toán Rijndael của hai nhà mật mã học người Bỉ, Dr. Joan Daemen và Dr. Vincent Rijmen, đã được chọn. Ngày 26 tháng 11 năm 2001, Rijndael chính thức trở thành tiêu chuẩn mã hóa AES (Advanced Encryption Standard) dưới chuẩn FIPS 197, đánh dấu một bước ngoặt trong lĩnh vực bảo mật thông tin.

#### ****Đặc điểm chính của thuật toán AES****

Giống như DES, AES là một thuật toán mã hóa khối đối xứng, sử dụng cùng một khóa cho cả mã hóa và giải mã. Tuy nhiên, AES khác biệt rõ rệt so với DES ở nhiều khía cạnh. Thuật toán Rijndael, nền tảng của AES, hỗ trợ nhiều kích thước khối và khóa linh hoạt hơn, thay vì bị giới hạn ở 64 bit và 56 bit như DES. Cụ thể, kích thước khối và khóa có thể được chọn độc lập trong các giá trị 128, 160, 192, 224, hoặc 256 bit. Tuy nhiên, tiêu chuẩn AES chỉ chấp nhận kích thước khối cố định 128 bit với ba lựa chọn khóa: 128 bit, 192 bit, hoặc 256 bit. Dựa vào kích thước khóa sử dụng, các phiên bản của AES được đặt tên tương ứng là AES-128, AES-192, và AES-256.

Không giống như DES, AES không dựa trên cấu trúc Feistel. Trong cấu trúc Feistel, nửa khối dữ liệu được sử dụng để sửa đổi nửa còn lại, sau đó hai nửa được hoán đổi. Ngược lại, AES xử lý toàn bộ khối dữ liệu song song trong mỗi vòng, sử dụng các bước thay thế (substitution) và hoán vị (permutation).

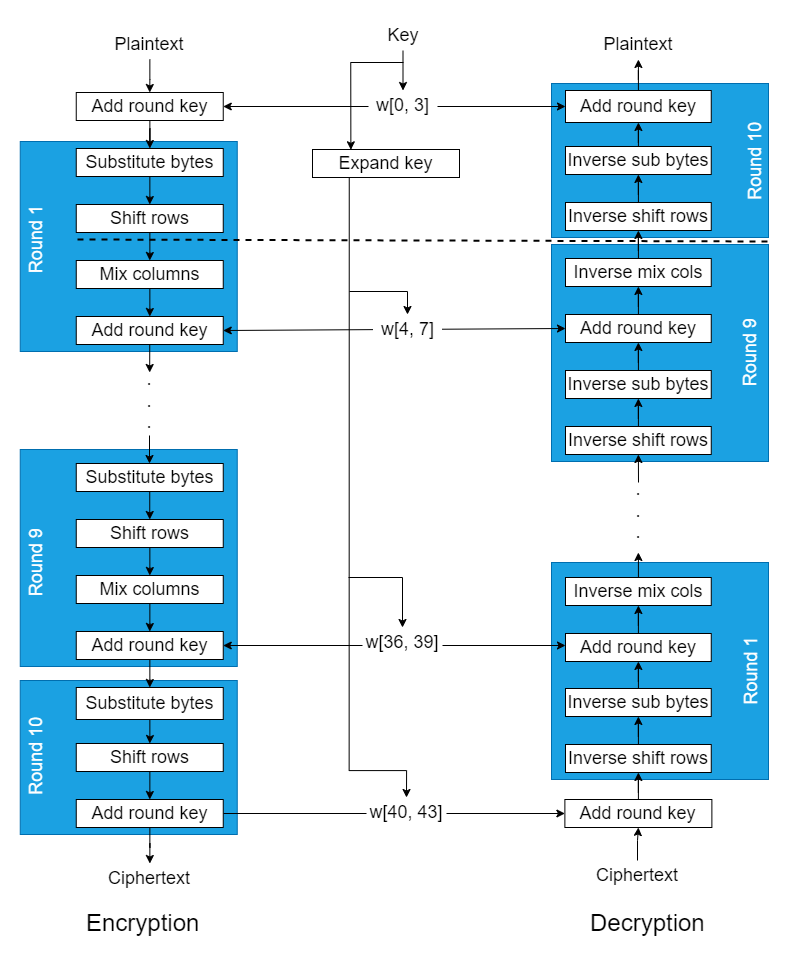
Các tham số của AES phụ thuộc vào độ dài khóa. Chẳng hạn, với khóa 128 bit, thuật toán sẽ thực hiện 10 vòng lặp, trong khi với khóa 192 bit và 256 bit, số vòng lần lượt là 12 và 14. Trong thực tế, khóa 128 bit hiện là kích thước phổ biến nhất được sử dụng, do đó mô tả thuật toán AES thường tập trung vào phiên bản này.

Rijndael được thiết kế với ba đặc điểm chính:

##### Khả năng chống lại mọi dạng tấn công đã biết.

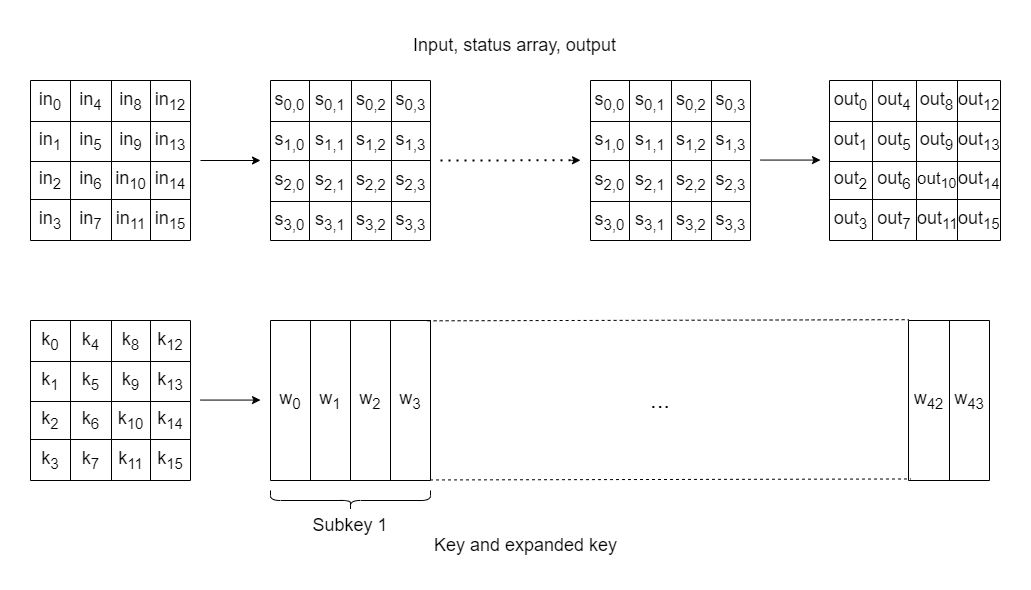
##### Tốc độ nhanh và mã gọn nhẹ trên nhiều nền tảng.

##### Thiết kế đơn giản, dễ triển khai.



Hình 4. Cấu trúc tổng thể của thuật toán AES

Cấu trúc tổng thể của AES được minh họa trong hình 4. Đầu vào là một khối 128 bit, được gọi là ma trận "in", sử dụng cho cả quá trình mã hóa và giải mã. Khối này được sao chép vào một mảng trạng thái (state array), sau đó được sửa đổi qua từng giai đoạn của thuật toán và cuối cùng sao chép sang một ma trận đầu ra như trong hình 5.



Hình 5. Cấu trúc dữ liệu của thuật toán AES

Cả văn bản gốc (plaintext) và khóa đều được biểu diễn dưới dạng ma trận vuông 128 bit gồm các byte. Khóa này sẽ được mở rộng thành một mảng từ khóa (key schedule) gọi là ma trận "w". Lưu ý rằng, thứ tự các byte trong ma trận "in" và ma trận "w" được tổ chức theo cột.

### Cơ chế hoạt động bên trong một vòng AES

Quy trình của thuật toán AES bắt đầu với giai đoạn Add Round Key, sau đó thực hiện 9 vòng lặp, mỗi vòng bao gồm 4 giai đoạn, và kết thúc bằng vòng thứ 10 với 3 giai đoạn. Cơ chế này áp dụng cho cả mã hóa và giải mã, tuy nhiên trong giải mã, mỗi giai đoạn sẽ là phiên bản nghịch đảo của giai đoạn tương ứng trong mã hóa.

Các giai đoạn trong một vòng mã hóa bao gồm:

##### Substitute Bytes (Thay thế byte): Thay thế từng byte bằng giá trị từ bảng S-box.

##### Shift Rows (Dịch hàng): Dịch vòng các hàng trong ma trận trạng thái.

##### Mix Columns (Trộn cột): Kết hợp các giá trị trong từng cột của ma trận.

##### Add Round Key (Thêm khóa vòng): XOR ma trận trạng thái với khóa vòng hiện tại.

Trong vòng thứ 10 của mã hóa, bước Mix Columns được bỏ qua.

Tương tự, trong giải mã, 9 vòng đầu bao gồm các giai đoạn nghịch đảo sau:

##### Inverse Shift Rows (Dịch hàng ngược).

##### Inverse Substitute Bytes (Thay thế byte ngược).

##### Inverse Add Round Key (Thêm khóa vòng ngược).

##### Inverse Mix Columns (Trộn cột ngược).

Vòng thứ 10 của giải mã cũng bỏ qua bước Inverse Mix Columns.

Phần tiếp theo sẽ trình bày chi tiết về từng giai đoạn trong quy trình trên.

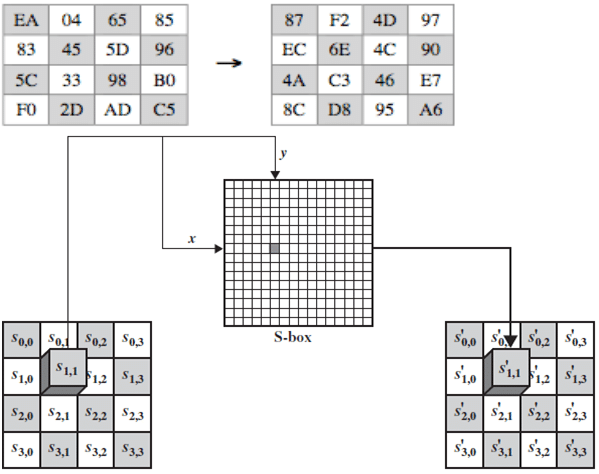
### Substitute Bytes

Giai đoạn SubBytes thực hiện việc tra cứu bảng bằng cách sử dụng một ma trận 16×16 gồm các giá trị byte, gọi là S-box. Ma trận này bao gồm tất cả các tổ hợp có thể có của một chuỗi 8 bit (2⁸ = 16 × 16 = 256). Tuy nhiên, S-box không phải là một sự sắp xếp ngẫu nhiên của các giá trị này mà được tạo ra theo một phương pháp được định nghĩa rõ ràng. Các nhà thiết kế thuật toán Rijndael đã công khai cách tạo ra S-box, khác với DES, nơi không có giải thích cụ thể cho S-box. Trong bài này, chúng ta chỉ cần xem S-box như một bảng tra cứu sẵn có.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | y | | | | | | | | | | | | | | | |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| x | 0 | 63 | 7C | 77 | 7B | F2 | 6B | 6F | C5 | 30 | 01 | 67 | 2B | FE | D7 | AB | 76 |
| 1 | CA | 82 | C9 | 7D | FA | 59 | 47 | F0 | AD | D4 | A2 | AF | 9C | A4 | 72 | C0 |
| 2 | B7 | FD | 93 | 26 | 36 | 3F | F7 | CC | 34 | A5 | E5 | F1 | 71 | D8 | 31 | 15 |
| 3 | 04 | C7 | 23 | C3 | 18 | 96 | 05 | 9A | 07 | 12 | 80 | E2 | EB | 27 | B2 | 75 |
| 4 | 09 | 83 | 2C | 1A | 1B | 6E | 5A | A0 | 52 | 3B | D6 | B3 | 29 | E3 | 2F | 84 |
| 5 | 53 | D1 | 00 | ED | 20 | FC | B1 | 5B | 6A | CB | BE | 39 | 4A | 4C | 58 | CF |
| 6 | D0 | EF | AA | FB | 43 | 4D | 33 | 85 | 45 | F9 | 02 | 7F | 50 | 3C | 9F | A8 |
| 7 | 51 | A3 | 40 | 8F | 92 | 9D | 38 | F5 | BC | B6 | DA | 21 | 10 | FF | F3 | D2 |
| 8 | CD | 0C | 13 | EC | 5F | 97 | 44 | 17 | C4 | A7 | 7E | 3D | 64 | 5D | 19 | 73 |
| 9 | 60 | 81 | 4F | DC | 22 | 2A | 90 | 88 | 46 | EE | B8 | 14 | DE | 5E | 0B | DB |
| A | E0 | 32 | 3A | 0A | 49 | 06 | 24 | 5C | C2 | D3 | AC | 62 | 91 | 95 | E4 | 79 |
| B | E7 | C8 | 37 | 6D | 8D | D5 | 4E | A9 | 6C | 56 | F4 | EA | 65 | 7A | AE | 08 |
| C | BA | 78 | 25 | 2E | 1C | A6 | B4 | C6 | E8 | DD | 74 | 1F | 4B | BD | 8B | 8A |
| D | 70 | 3E | B5 | 66 | 48 | 03 | F6 | 0E | 61 | 35 | 57 | B9 | 86 | C1 | 1D | 9E |
| E | E1 | F8 | 98 | 11 | 69 | D9 | 8E | 94 | 9B | 1E | 87 | E9 | CE | 55 | 28 | DF |
| F | 8C | A1 | 89 | 0D | BF | E6 | 42 | 68 | 41 | 99 | 2D | 0F | B0 | 54 | BB | 16 |

Bảng 1. Bảng S-box

Trong quá trình mã hóa, ma trận được xử lý được gọi là state. Tại mỗi vòng, từng byte trong ma trận state sẽ được ánh xạ sang một byte mới. Quá trình ánh xạ này sử dụng nibble trái của byte để xác định hàng trong S-box và nibble phải để xác định cột. Ví dụ: byte {95} (giá trị hex) sẽ chọn hàng 9 và cột 5 trong S-box, tương ứng với giá trị {2A}. Giá trị này sau đó được sử dụng để cập nhật ma trận state. Hình 6 biểu diễn chi tiết ý tưởng này.



Hình 6. Giai đoạn Subsitute Bytes của thuật toán AES

Giai đoạn nghịch đảo của SubBytes, được gọi là InvSubBytes, sử dụng một S-box nghịch đảo. Trong trường hợp này, giá trị {2A} sẽ được ánh xạ trở lại giá trị {95}. Bảng 7.4 thể hiện cả S-box thuận và nghịch, qua đó có thể kiểm chứng sự đúng đắn của quá trình này.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | y | | | | | | | | | | | | | | | |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| x | 0 | 52 | 09 | 6A | D5 | 30 | 36 | A5 | 38 | BF | 40 | A3 | 9E | 81 | F3 | D7 | FB |
| 1 | 7C | E3 | 39 | 82 | 9B | 2F | FF | 87 | 34 | 8E | 43 | 44 | C4 | DE | E9 | CB |
| 2 | 54 | 7B | 94 | 32 | A6 | C2 | 23 | 3D | EE | 4C | 95 | 0B | 42 | FA | C3 | 4E |
| 3 | 08 | 2E | A1 | 66 | 28 | D9 | 24 | B2 | 76 | 5B | A2 | 49 | 6D | 8B | D1 | 25 |
| 4 | 72 | F8 | F6 | 64 | 86 | 68 | 98 | 16 | D4 | A4 | 5C | CC | 5D | 65 | B6 | 92 |
| 5 | 6C | 70 | 48 | 50 | FD | ED | B9 | DA | 5E | 15 | 46 | 57 | A7 | 8D | 9D | 84 |
| 6 | 90 | D8 | AB | 00 | 8C | BC | D3 | 0A | F7 | E4 | 58 | 05 | B8 | B3 | 45 | 06 |
| 7 | D0 | 2C | 1E | 8F | CA | 3F | 0F | 02 | C1 | AF | BD | 03 | 01 | 13 | 8A | 6B |
| 8 | 3A | 91 | 11 | 41 | 4F | 67 | DC | EA | 97 | F2 | CF | CE | F0 | B4 | E6 | 73 |
| 9 | 96 | AC | 74 | 22 | E7 | AD | 35 | 85 | E2 | F9 | 37 | E8 | 1C | 75 | DF | 6E |
| A | 47 | F1 | 1A | 71 | 1D | 29 | C5 | 89 | 6F | B7 | 62 | 0E | AA | 18 | BE | 1B |
| B | FC | 56 | 3E | 4B | C6 | D2 | 79 | 20 | 9A | DB | C0 | FE | 78 | CD | 5A | F4 |
| C | 1F | DD | A8 | 33 | 88 | 07 | C7 | 31 | B1 | 12 | 10 | 59 | 27 | 80 | EC | 5F |
| D | 60 | 51 | 7F | A9 | 19 | B5 | 4A | 0D | 2D | E5 | 7A | 9F | 93 | C9 | 9C | EF |
| E | A0 | E0 | 3B | 4D | AE | 2A | F5 | B0 | C8 | EB | BB | 3C | 83 | 53 | 99 | 61 |
| F | 17 | 2B | 04 | 7E | BA | 77 | D6 | 26 | E1 | 69 | 14 | 63 | 55 | 21 | 0C | 7D |

Bảng 2. Bảng S-box ngược

S-box được thiết kế để kháng lại các dạng tấn công mật mã đã biết. Cụ thể, các nhà thiết kế Rijndael đã đảm bảo rằng:

##### S-box có mối tương quan thấp giữa các bit đầu vào và đầu ra.

##### Đầu ra không thể được mô tả bằng một hàm toán học đơn giản từ đầu vào.

##### S-box không có các điểm cố định (s-box(a) = a) hoặc điểm cố định đối lập (s-box(a) = −a, với −a là bù bit của a).

##### S-box phải có khả năng nghịch đảo (Is-box[s-box(a)] = a) nhưng không phải là nghịch đảo của chính nó (s-box(a) ≠ Is-box(a)).

Thiết kế này đảm bảo rằng S-box vừa bảo mật vừa hỗ trợ hiệu quả cho quá trình mã hóa và giải mã.

### Shift Row Transformation

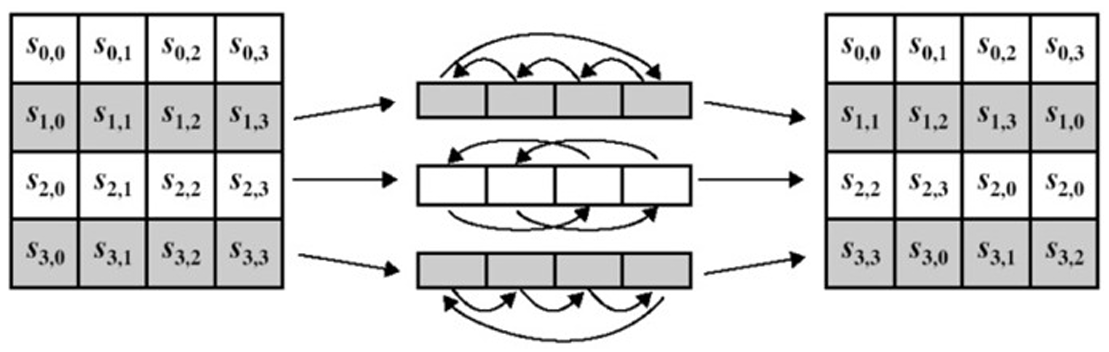
Giai đoạn này, được gọi là ShiftRows, thực hiện một phép hoán vị đơn giản trong ma trận state. Cách hoạt động như sau:

##### Hàng đầu tiên của ma trận state không thay đổi.

##### Hàng thứ hai được dịch 1 byte sang trái theo kiểu vòng lặp (circular shift).

##### Hàng thứ ba được dịch 2 byte sang trái theo kiểu vòng lặp.

##### Hàng thứ tư được dịch 3 byte sang trái theo kiểu vòng lặp.



Hình 7. Giai đoạn ShiftRows

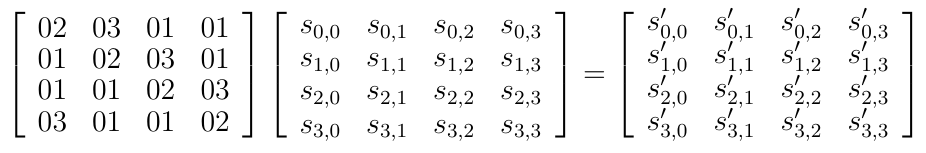
Biến đổi ngược của ShiftRows, gọi là Inverse ShiftRows, thực hiện các phép dịch vòng tương tự nhưng theo chiều ngược lại, áp dụng cho ba hàng cuối, trong khi hàng đầu tiên vẫn giữ nguyên.

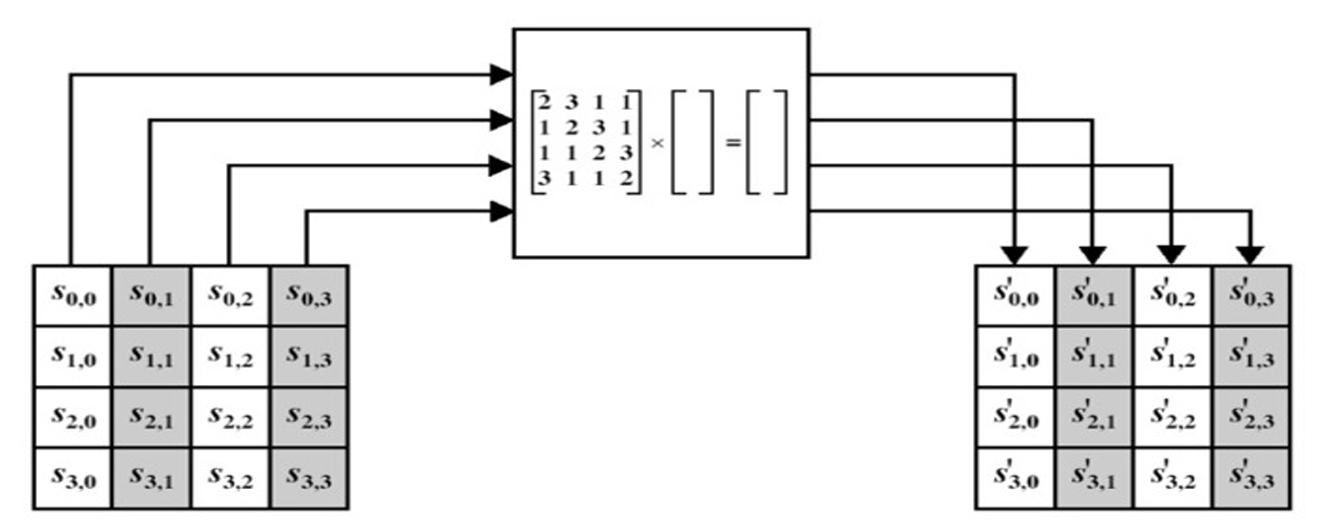
Mặc dù thoạt nhìn, phép dịch này có vẻ đơn giản và không làm thay đổi nhiều, nhưng thực tế, nó có tác động lớn đến cách các byte được sắp xếp trong ma trận state. Hãy nhớ rằng ma trận state được xử lý dưới dạng một mảng gồm bốn cột byte. Ví dụ: cột đầu tiên đại diện cho các byte 1, 2, 3, và 4. Một phép dịch 1 byte thực tế sẽ làm dịch chuyển dữ liệu một khoảng cách tuyến tính là 4 byte.

Phép biến đổi ShiftRows cũng đảm bảo rằng bốn byte của một cột sẽ được phân tán sang bốn cột khác nhau, tăng cường khả năng trộn dữ liệu và cải thiện bảo mật của thuật toán.

### Mix Column Transformation

Giai đoạn này, được gọi là MixColumns, thực chất là một phép thay thế nhưng sử dụng phép toán đại số trong trường hữu hạn GF(2⁸). Trong quá trình này, từng cột của ma trận state được xử lý riêng biệt. Mỗi byte trong một cột sẽ được ánh xạ thành một giá trị mới, giá trị này là hàm của tất cả bốn byte trong cùng cột. Phép biến đổi MixColumns được xác định bằng phép nhân ma trận với ma trận state.

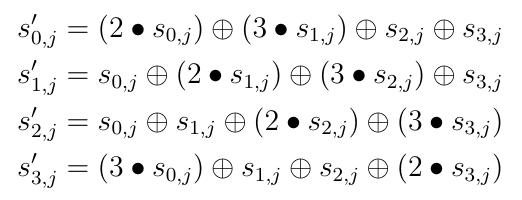




Hình 8. Giai đoạn MixColumns

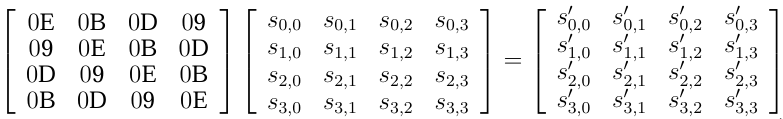
Mỗi phần tử trong ma trận kết quả của phép nhân được tính bằng tổng của các tích giữa phần tử của một hàng và một cột. Trong trường hợp này, các phép cộng và nhân đều được thực hiện trong trường hữu hạn GF(2⁸).

Phép biến đổi MixColumns áp dụng cho một cột duy nhất j (0≤j≤3) của ma trận state có thể được biểu diễn như sau:



Trong biểu thức trên, ký hiệu • biểu thị phép nhân được thực hiện trong trường hữu hạn GF(2⁸).

Phép InvMixColumns được định nghĩa bằng phép nhân ma trận sau:



Trong đó:

##### Ma trận đầu tiên (A) là nghịch đảo của ma trận MixColumns ()

##### Trạng thái trước MixColumns là S và sau MixColumns là S′.

Do đó, khi thực hiện InvMixColumns, ta có:

= =S

Phép tính nghịch đảo này đảm bảo rằng quá trình mã hóa và giải mã trong AES là đối xứng và có thể đảo ngược một cách chính xác.

### Add Round Key

Trong giai đoạn này, được gọi là **AddRoundKey**, 128 bit của ma trận **state** được thực hiện phép **XOR từng bit** với 128 bit của **khóa vòng** tương ứng.

Cách thức hoạt động: Phép XOR được thực hiện theo từng cột của ma trận **state**, trong đó:

##### Mỗi cột gồm 4 byte trong ma trận **state**.

##### Mỗi cột được XOR với một **word** (4 byte) của **khóa vòng**.

Đặc điểm:

##### Đây là một phép biến đổi **đơn giản nhất** trong các bước của AES, giúp đảm bảo tính hiệu quả trong quá trình tính toán.

##### Tuy nhiên, mặc dù đơn giản, phép biến đổi này ảnh hưởng đến **mọi bit** trong ma trận **state**, đảm bảo sự khuếch tán của thông tin từ khóa vào dữ liệu.

Giai đoạn **AddRoundKey** đóng vai trò quan trọng trong việc kết hợp khóa vòng với dữ liệu, giúp tăng cường tính bảo mật cho mỗi vòng của thuật toán AES.

### Key Expansion

Thuật toán mở rộng khóa AES nhận đầu vào là một khóa gồm 4 từ (4-word key) và tạo ra một mảng tuyến tính gồm 44 từ. Mỗi vòng mã hóa sử dụng 4 từ từ mảng này (xem Hình 7.2). Mỗi từ chứa 32 byte, điều này có nghĩa là mỗi khóa con (subkey) có độ dài 128 bit.

Cách thực hiện:

**Bước đầu tiên:** Khóa ban đầu được sao chép vào 4 từ đầu tiên của mảng khóa mở rộng.

**Phần còn lại:** Mảng khóa mở rộng được điền thêm 4 từ mỗi lần.

##### Mỗi từ mới w[i] phụ thuộc vào từ ngay trước nó w[i−1] và từ cách nó 4 vị trí w[i−4].

##### Trong 3/4 trường hợp, chỉ cần sử dụng phép XOR đơn giản.

##### Với các từ có vị trí trong mảng w là bội số của 4, một hàm phức tạp hơn được sử dụng, ký hiệu là g.

Hàm g bao gồm các bước sau:

##### **RotWord:** Thực hiện dịch vòng 1 byte sang trái trên một từ.

##### **SubWord:** Thay thế từng byte của từ đầu vào bằng giá trị tương ứng từ S-box.

##### **Thêm Rcon:** Kết quả của bước 1 và 2 được thực hiện phép XOR với hằng số vòng (round constant) Rcon[j].

Hằng số vòng (Round Constant - Rcon):

##### Hằng số vòng là một từ trong đó 3 byte bên phải luôn bằng 0.

##### Phép XOR với Rcon chỉ ảnh hưởng đến byte đầu tiên bên trái của từ.

##### Hằng số vòng khác nhau cho từng vòng và được định nghĩa như sau:

Rcon[j] = (RC[j],0,0,0)

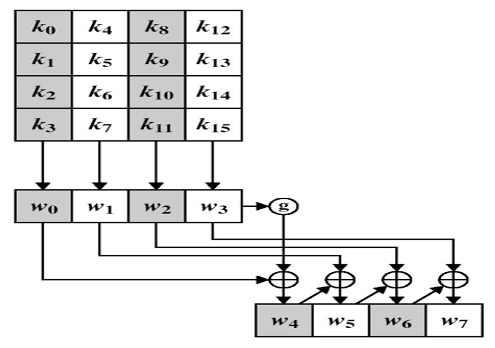
Trong đó:

RC[1] = 1,

RC[j] = 2 . RC[j−1], với phép nhân được định nghĩa trong trường hữu hạn GF().

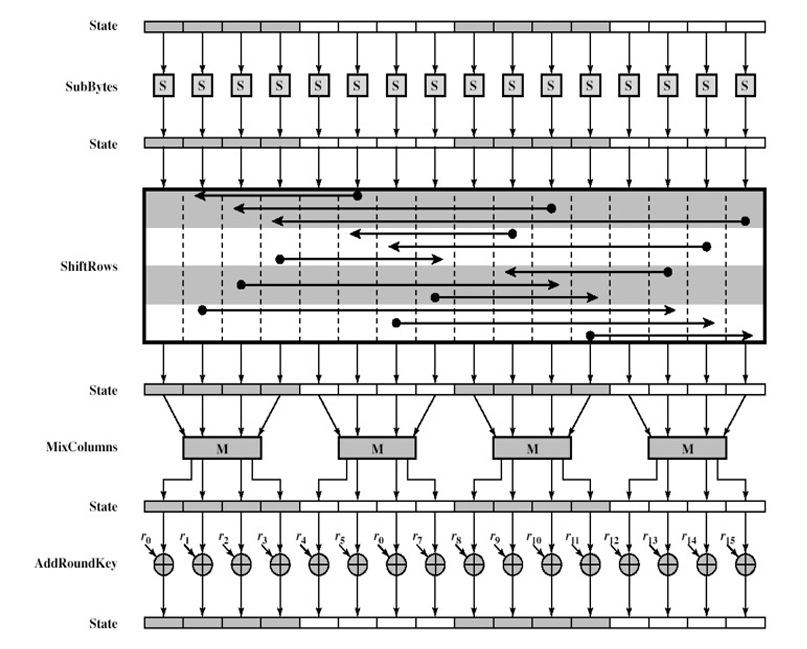
Thiết kế bảo mật:

Quá trình mở rộng khóa được thiết kế để chống lại các dạng tấn công mật mã đã biết. Việc thêm hằng số vòng phụ thuộc vào từng vòng giúp loại bỏ tính đối xứng hoặc sự tương đồng trong cách các khóa vòng được tạo ra.



Hình 9. Mở rộng khóa AES

Hình 10 cung cấp tóm tắt về các vòng mã hóa. Trong đó, cột ShiftRows được biểu diễn dưới dạng dịch chuyển tuyến tính, giúp minh họa rõ hơn vai trò của nó trong quá trình mã hóa.



Hình 10. Vòng mã hóa AES

### Cấu trúc dữ liệu

#### Khối dữ liệu đầu vào (Input Block)

Plaintext: Dữ liệu cần mã hóa, thường là chuỗi nhị phân hoặc byte.

##### Kích thước: 128 bit (16 byte).

##### Dữ liệu này được chia thành Ma trận trạng thái (State Matrix) để xử lý.

#### ****Khóa (Key)****

**Key**: Một chuỗi nhị phân hoặc byte được sử dụng để mã hóa và giải mã. Kích thước của khóa có thể là:

##### **128 bit (16 byte)**: AES-128.

##### **192 bit (24 byte)**: AES-192.

##### **256 bit (32 byte)**: AES-256.

**Key Schedule**: Từ khóa gốc, AES sinh ra một danh sách khóa con (subkeys) cho từng vòng mã hóa.

##### Cấu trúc: Mảng hoặc danh sách các khóa con.

##### Kích thước: (Nr+1)×128 bit, với Nr là số vòng (10, 12, hoặc 14 tùy vào kích thước khóa).

#### ****Ma trận trạng thái (State Matrix)****

Đây là cấu trúc dữ liệu chính để biểu diễn và xử lý dữ liệu trong AES.

Dạng **ma trận 4x4**, với mỗi ô chứa 1 byte (8 bit)

Dữ liệu từ chuỗi Plaintext được ánh xạ vào ma trận trạng thái trước khi xử lý.

#### ****Hộp S-box (Substitution Box)****

**S-box**: Bảng tra cứu (lookup table) 256 phần tử (16x16) dùng để thay thế các byte trong ma trận trạng thái trong bước **SubBytes**.

##### Kích thước: 16×16 (256 byte).

##### Dạng: Mảng 2D hoặc bảng ánh xạ (Hash Table).

#### ****Khóa con (Subkeys)****

Danh sách khóa con sinh ra từ khóa ban đầu qua quá trình mở rộng khóa (Key Expansion).

Kích thước:

##### **AES-128**: 11 khóa con (10 vòng + khóa ban đầu).

##### **AES-192**: 13 khóa con.

##### **AES-256**: 15 khóa con.

Cấu trúc: Mảng hoặc danh sách chứa các giá trị 4×4 byte cho mỗi khóa con.

#### Hộp MixColumns (Galois Field Table)

Bước MixColumns sử dụng một phép nhân ma trận trên trường Galois GF().

Hằng số MixColumns:

##### Ma trận cố định 4×4, mỗi phần tử là một byte.

##### Dạng dữ liệu: Mảng 2D hoặc danh sách.

Dạng dữ liệu: Mảng 2D hoặc danh sách.

|  |  |  |
| --- | --- | --- |
| Thành phần | Cấu trúc dữ liệu | Kích thước |
| Plaintext | Chuỗi hoặc mảng byte | 16 byte (128bit) |
| Key | Chuỗi hoặc mảng byte | 16/24/32 byte |
| State Matrix | Ma trân 4x4 | 16 byte |
| S-box | Mảng 2D hoặc Hash Table | 256 byte |
| Key Schedule | Mảng khóa con | (Nr+1) x 16 byte |
| MixColumns Matrix | Mảng 2D | 4 x 4 byte |

### Độ phức tạp của mã hóa AES

#### Độ phức tạp thời gian (Time Complexity)

Mã hóa AES bao gồm nhiều vòng xử lý (10, 12, hoặc 14 vòng tùy thuộc vào độ dài khóa), với mỗi vòng thực hiện các bước sau:

SubBytes (Thay thế byte):

##### Duyệt qua toàn bộ ma trận trạng thái 4×4 (16 byte).

##### Thời gian truy xuất S-box: O(1) cho mỗi byte.

##### Tổng thời gian: O(16)=O(1) (kích thước ma trận cố định).

ShiftRows (Dịch vòng hàng):

##### Dịch các hàng trong ma trận trạng thái.

##### Thao tác trực tiếp trên 16 byte, tốn thời gian O(1).

MixColumns (Nhân ma trận trên GF()):

##### Xử lý từng cột trong ma trận trạng thái (4 cột).

##### Mỗi cột (4 byte) cần 16 phép toán XOR và nhân trong GF().

##### Tổng thời gian: O(4×4)=O(16)=O(1).

AddRoundKey (XOR với khóa con):

##### XOR từng byte trong ma trận trạng thái với khóa con (16 byte).

##### Thời gian: O(16)=O(1).

Số vòng mã hóa (Nr):

##### Nr=10 (AES-128), Nr=12 (AES-192), Nr=14 (AES-256).

Tổng thời gian:

O(Nr×(SubBytes+ShiftRows+MixColumns+AddRoundKey))=O(Nr×1)=O(Nr).

Với Nr là hằng số nhỏ (10, 12, 14), độ phức tạp thời gian tổng quát là:

O(1) (tính trên kích thước cố định của khối dữ liệu là 128 bit).

#### Độ phức tạp không gian (Space Complexity)

Các thành phần chính ảnh hưởng đến bộ nhớ:

##### Ma trận trạng thái có kích thước: 4×4=16 byte (O(1)).

##### Khóa ban đầu: 16, 24, hoặc 32 byte (O(1)).

##### Khóa con (Key Schedule): 176 byte (AES-128), 208 byte (AES-192), 240 byte (AES-256) đều là O(1) vì kích thước không phụ thuộc vào độ dài chuỗi đầu vào.

##### S-box: Bảng tra cứu 256 phần tử (256 byte): O(1).

##### Dữ liệu tạm thời: Dùng cho từng bước như nhân trong GF() hoặc XOR: O(1).

##### Tổng không gian: Không gian sử dụng cố định và không tăng theo độ dài đầu vào nên độ phức tạp là O(1).

|  |  |
| --- | --- |
| Loại phức tạp | Giá trị |
| Thời gian | O(1) (với khối 128-bit cố định) |
| Không gian | O(1) (kích thước cố định) |

Độ phức tạp thời gian và không gian có thể coi là hằng số vì AES xử lý dữ liệu theo khối (block size cố định là 128 bit). Nếu mã hóa nhiều khối (chế độ CBC, CTR), thời gian và không gian sẽ tỷ lệ tuyến tính với số lượng khối cần mã hóa.

### Ứng dụng của mã hóa AES

Thuật toán mã hóa AES (Advanced Encryption Standard) được sử dụng rộng rãi trong nhiều lĩnh vực nhờ vào tính bảo mật cao, hiệu suất tốt và khả năng chống lại các cuộc tấn công. Một số ứng dụng tiêu biểu của AES bao gồm:

Bảo mật dữ liệu trong truyền thông AES được tích hợp trong các giao thức truyền thông như HTTPS để mã hóa dữ liệu giữa trình duyệt và máy chủ web, đảm bảo an toàn cho các giao dịch trực tuyến như ngân hàng và mua sắm. Bên cạnh đó, trong các hệ thống VPN (Mạng riêng ảo), AES giúp mã hóa dữ liệu truyền qua mạng công cộng, bảo vệ tính riêng tư và ngăn chặn việc nghe lén dữ liệu.

Bảo mật trên thiết bị di động: Các ứng dụng nhắn tin như WhatsApp, Signal sử dụng AES để mã hóa tin nhắn và cuộc gọi đầu cuối, đảm bảo nội dung chỉ có người gửi và người nhận truy cập được. Ngoài ra, các hệ điều hành như Android và iOS tích hợp AES để mã hóa dữ liệu lưu trữ trên thiết bị, bảo vệ thông tin cá nhân khỏi các hành vi xâm nhập hoặc lạm dụng nếu thiết bị bị mất cắp.

Bảo mật dữ liệu lưu trữ: Trong lĩnh vực lưu trữ dữ liệu, AES được ứng dụng để mã hóa các ổ cứng, USB bảo mật và các phần mềm mã hóa tệp tin như BitLocker, VeraCrypt. Các dịch vụ lưu trữ đám mây như Google Drive, Dropbox cũng sử dụng AES để mã hóa dữ liệu trước khi tải lên, giúp bảo vệ người dùng khỏi các cuộc tấn công hoặc rò rỉ dữ liệu.

Bảo mật trong giao dịch tài chính: AES đóng vai trò quan trọng trong việc mã hóa thông tin nhạy cảm như số thẻ tín dụng, tài khoản ngân hàng trong các hệ thống giao dịch điện tử. Hơn nữa, các máy POS (Point of Sale) cũng sử dụng AES để bảo vệ thông tin thanh toán của khách hàng khỏi các cuộc tấn công.

Bảo mật trong hệ thống IoT (Internet of Things): Các thiết bị IoT như camera an ninh, cảm biến môi trường, và thiết bị nhà thông minh sử dụng AES để mã hóa dữ liệu truyền tải giữa thiết bị và máy chủ, đảm bảo không ai có thể truy cập hoặc điều khiển trái phép các thiết bị này. Điều này đặc biệt quan trọng trong các ứng dụng IoT yêu cầu tính bảo mật cao, như giám sát nhà ở hoặc cơ sở công nghiệp.

Hệ thống quản lý và kiểm soát truy cập: AES được ứng dụng trong các thẻ RFID và hệ thống kiểm soát truy cập ra vào để mã hóa dữ liệu trao đổi, giúp ngăn chặn các hành vi sao chép hoặc xâm nhập trái phép. Ví dụ, các hệ thống quản lý tòa nhà thông minh hay bãi đỗ xe tự động thường sử dụng AES để bảo vệ các hoạt động giao tiếp dữ liệu.

Mã hóa email và tệp tin: Các dịch vụ email bảo mật như ProtonMail sử dụng AES để mã hóa email đầu cuối, đảm bảo nội dung email không bị đọc lén bởi bên thứ ba. Ngoài ra, các phần mềm nén tệp tin như WinRAR, 7-Zip cũng tích hợp AES để bảo vệ dữ liệu cá nhân hoặc doanh nghiệp, đặc biệt khi chia sẻ qua Internet.

Blockchain và tiền mã hóa: AES đóng vai trò hỗ trợ trong các hệ thống blockchain để mã hóa dữ liệu giao dịch, giúp đảm bảo tính an toàn và bảo mật của mạng lưới. Ngoài ra, trong tiền mã hóa, AES được sử dụng để bảo vệ các khóa cá nhân, giảm thiểu rủi ro bị đánh cắp hoặc lạm dụng, từ đó nâng cao độ tin cậy của hệ thống.

## Giải thuật RSA

### Lịch sử phát triển của giải thuật RSA

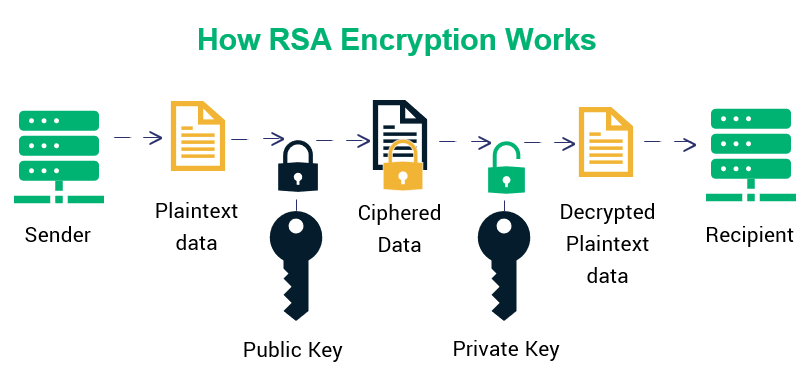
Thuật toán được [Ron Rivest](https://vi.wikipedia.org/w/index.php?title=Ron_Rivest&action=edit&redlink=1), [Adi Shamir](https://vi.wikipedia.org/w/index.php?title=Adi_Shamir&action=edit&redlink=1) và [Len Adleman](https://vi.wikipedia.org/w/index.php?title=Len_Adleman&action=edit&redlink=1) mô tả lần đầu tiên vào năm [1977](https://vi.wikipedia.org/wiki/1977) tại [Học viện Công nghệ Massachusetts](https://vi.wikipedia.org/wiki/H%E1%BB%8Dc_vi%E1%BB%87n_C%C3%B4ng_ngh%E1%BB%87_Massachusetts) (MIT). Tên của thuật toán lấy từ 3 chữ cái đầu của tên 3 tác giả.

Trước đó, vào năm [1973](https://vi.wikipedia.org/wiki/1973), [Clifford Cocks](https://vi.wikipedia.org/w/index.php?title=Clifford_Cocks&action=edit&redlink=1), một nhà toán học người Anh làm việc tại [GCHQ](https://vi.wikipedia.org/w/index.php?title=GCHQ&action=edit&redlink=1), đã mô tả một thuật toán tương tự. Với khả năng tính toán tại thời điểm đó thì thuật toán này không khả thi và chưa bao giờ được thực nghiệm. Tuy nhiên, phát minh này chỉ được công bố vào năm [1997](https://vi.wikipedia.org/wiki/1997) vì được xếp vào loại tuyệt mật.

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm [1983](https://vi.wikipedia.org/wiki/1983) (Số đăng ký 4.405.829). Bằng sáng chế này hết hạn vào ngày [21 tháng 9](https://vi.wikipedia.org/wiki/21_th%C3%A1ng_9) năm [2000](https://vi.wikipedia.org/wiki/2000). Tuy nhiên, do thuật toán đã được công bố trước khi có đăng ký bảo hộ nên sự bảo hộ hầu như không có giá trị bên ngoài Hoa Kỳ. Ngoài ra, nếu như công trình của Clifford Cocks đã được công bố trước đó thì bằng sáng chế RSA đã không thể được đăng ký.

### Cơ chế hoạt động của RSA

Thuật toán RSA có hai khóa: khóa công khai (hay khóa công cộng) và khóa bí mật (hay khóa cá nhân). Mỗi khóa là những số cố định sử dụng trong quá trình mã hóa và giải mã. Khóa công khai được công bố rộng rãi cho mọi người và được dùng để mã hóa. Những thông tin được mã hóa bằng khóa công khai chỉ có thể được giải mã bằng khóa bí mật tương ứng. Nói cách khác, mọi người đều có thể mã hóa nhưng chỉ có người biết khóa cá nhân (bí mật) mới có thể giải mã được.



Hình 11. Cơ chế hoạt động của mã hóa RSA

#### Cơ chế hoạt động

Được mô tả lần đầu bởi Ron Rivest, Adi Shamir và Len Adleman vào 1977 tại Học viện Công nghệ Massachusetts (MIT). Hoạt động của RSA dựa trên 4 bước chính: sinh khóa, chia sẻ key, mã hóa và giải mã.

**Quá trình sinh khóa:**

Việc tạo khóa trong RSA dựa trên việc tìm ra bộ ba số tự nhiên: e, d, và n, với yêu cầu rằng khi mã hóa và giải mã thông điệp m, công thức sau được thỏa mãn:

|  |
| --- |
| mod n = mod n |

Một điểm quan trọng là private key d phải được bảo mật tuyệt đối. Ngay cả khi ai đó biết được e, n, hay thông điệp m, họ cũng không thể tính được d. Cụ thể, quá trình sinh khóa trong RSA gồm các bước như sau:

**Bước 1: Chọn hai số nguyên tố lớn p và q:**

##### p và q phải được chọn ngẫu nhiên và bí mật.

##### Kích thước của p và q (thường từ 512 đến 2048 bit) quyết định độ an toàn của RSA.

**Bước 2: Tính giá trị n:**

|  |
| --- |
| n = p.q |

##### n được gọi là modulus và là một phần của khóa công khai.

##### Độ dài của n (số bit) là độ dài của khóa RSA.

**Bước 3: Tính hàm Euler ϕ(n). Hàm Euler được tính như sau:**

|  |
| --- |
| ϕ(n) = (p−1) × (q−1) |

##### Đây là số các số nguyên nhỏ hơn n mà không có ước chung lớn hơn 1 với n.

**Bước 4: Chọn số nguyên e** thỏa mãn:

|  |
| --- |
| 1< e < ϕ(n), và gcd(e, ϕ(n)) = 1 |

**Bước 5: Tính khóa bí mật d, với** d là nghịch đảo modular của eee theo ϕ(n):

|  |
| --- |
| d \* e ≡ 1 mod ϕ(n) |

##### Giá trị d được tính bằng thuật toán Euclid mở rộng.

**Bước 6: Từ các kết quả được tính ở trên sẽ ra được khóa như sau:**

##### **Public key:**Là bộ số (e, n), và có thể chia sẻ công khai.

##### **Private key:**Là bộ số (d, n), cần được giữ bí mật.

**Mã hóa thông điệp:**

Người gửi sử dụng khóa công khai (n,e) để mã hóa thông điệp.

Bước 1: Biến thông điệp thành dạng số nguyên m. Thông điệp gốc m được chuyển đổi sang dạng số nguyên sao cho:

|  |
| --- |
| 0 ≤ m < n |

Bước 2: Tính ciphertext c. Thông điệp m được mã hóa thành ciphertext c theo công thức:

|  |
| --- |
| c = mod n |

Ciphertext ccc sau đó được gửi đến người nhận.

**Giải mã thông điệp:**

Người nhận sử dụng khóa bí mật (n,d) để giải mã ciphertext c.

Bước 1: Tính lại plaintext m. Plaintext m được tính bằng công thức:

|  |
| --- |
| mod n |

Bước 2: Chuyển đổi lại dạng gốc: Giá trị m được chuyển đổi ngược thành thông điệp ban đầu.

### Cấu trúc dữ liệu

#### Số nguyên lớn (Large Integers)

RSA dựa trên các phép toán với số nguyên rất lớn (thường có độ dài từ 1024, 2048, đến 4096 bit).

Các số nguyên này được sử dụng trong việc tạo khóa, mã hóa, và giải mã.

Các cấu trúc dữ liệu cơ bản:

##### Số nguyên nguyên tố lớn p và q: Hai số nguyên tố lớn được chọn để tính n.

##### Số nguyên n: n=p×q, được sử dụng làm modulus cho khóa công khai và khóa bí mật.

##### Số nguyên ϕ(n): ϕ(n)=(p−1)×(q−1), được dùng để tính toán khóa riêng d.

#### Khóa RSA (RSA Keys)

**Khóa công khai (Public Key):** Khóa công khai được sử dụng để mã hóa dữ liệu.

Bao gồm:

##### n: Modulus (kích thước rất lớn, thường là 2048 bit hoặc hơn).

##### e: Số mũ công khai (exponent), thường là số nhỏ như 65537 để tối ưu hóa hiệu năng.

Dạng dữ liệu: Struct hoặc Tuple chứa (n,e).

**Khóa bí mật (Private Key):** Khóa bí mật được sử dụng để giải mã dữ liệu.

Bao gồm:

##### n: Modulus (giống với khóa công khai).

##### d: Số mũ bí mật (private exponent), được tính bằng: d ≡ (mod ϕ(n))

Dạng dữ liệu: Struct hoặc Tuple chứa (n,d).

**Thông tin bổ sung (Optimization for Private Key):**

RSA thường lưu thêm các giá trị để tối ưu hóa giải mã:

##### p và q: Hai số nguyên tố lớn.

##### = d mod (p-1) và =d mod  (q−1): Private exponent cho từng số nguyên tố.

##### ​= mod p: Đảo ngược modular của q theo p.

Các giá trị này được lưu trong cấu trúc dữ liệu mở rộng:

Private Key Struct: (p, q, , , )

#### Dữ liệu đầu vào và đầu ra

Plaintext (dữ liệu cần mã hóa):

##### Dạng: Số nguyên m (biểu diễn chuỗi dưới dạng số nguyên).

##### Điều kiện: 0 ≤ m < n.

Ciphertext (dữ liệu sau khi mã hóa):

##### Dạng: Số nguyên c.

##### Điều kiện: 0 ≤ c < n.

Cả plaintext và ciphertext đều được lưu dưới dạng số nguyên lớn.

#### Hàm băm (Hashing) và Padding

RSA không mã hóa trực tiếp chuỗi dài mà thường áp dụng các kỹ thuật padding để tăng cường bảo mật:

PKCS#1 v1.5 hoặc OAEP:

##### Dữ liệu gốc được mở rộng hoặc định dạng lại trước khi mã hóa.

##### Dạng: Mảng byte (kích thước phù hợp với modulus n).

Hàm băm:

##### Thường sử dụng SHA-256 hoặc SHA-512 để sinh chữ ký số.

#### Cấu trúc số học Modular

RSA sử dụng các phép toán số học modular:

Modular Exponentiation:

##### c = mod  n (Mã hóa).

##### m = mod  n (Giải mã).

Các phép toán modular lớn thường sử dụng cấu trúc như:

##### Montgomery Multiplication: Tăng hiệu suất tính toán.

Tổng hợp cấu trúc dữ liệu:

|  |  |  |
| --- | --- | --- |
| Thành phần | Dạng dữ liệu | Kích thước |
| p,q | Số nguyên | ≈ n/2 bit |
| n = p x q | Số nguyên | n bit |
| e | Số nguyên | 16 bit hoặc 32 bit |
| d | Số nguyên | n bit |
| ϕ(n) | Số nguyên | n bit |
| , , | Số nguyên | ≈ n/2 bit |
| Plaintext, Ciphertext | Số nguyên | ≤ n bit |

### Độ phức tạp thuật toán

#### Độ phức tạp thời gian

**Tạo khóa:**

##### Sinh số nguyên tố p,q: Sử dụng thuật toán kiểm tra nguyên tố (như Miller-Rabin) có độ phức tạp O(), trong đó k là số bit của p hoặc q.

##### Tìm nghịch đảo modular d: Sử dụng thuật toán mở rộng Euclid (O()).

Vậy độ phức tạp tổng: O(), với k là số bit của n.

**Mã hóa:**

Tính toán lũy thừa modular mod  n bằng thuật toán bình phương và nhân (exponentiation by squaring):

##### Số phép nhân modular: O(log(e)).

##### Phép nhân modular: O().

Tổng độ phức tạp: O(⋅log(e)).

##### Với e thường là số nhỏ (ví dụ: e=65537e = 65537e=65537), log(e) là hằng số, vì vậy thực tế là O().

**Giải mã:**

Tính toán lũy thừa modular mod n (tương tự mã hóa nhưng d lớn hơn e):

##### Số phép nhân modular: O(log(d)).

##### Phép nhân modular: O().

Tổng độ phức tạp: O(. log(d)).

Vì d có cùng kích thước với n, log(d) ≈ k, do đó thực tế là O().

Sử dụng CRT để giảm kích thước tính toán bằng cách làm việc với p và q riêng lẻ:

##### Tính =d mod  (p−1), = d mod (q-1).

##### Giảm kích thước tính toán từ O() xuống O.

Do vậy tổng độ phức tạp là: O.

#### Độ phức tạp không gian

RSA cần lưu trữ các số nguyên lớn p, q, n, ϕ(n), e, d và các giá trị trung gian trong quá trình tính toán.

Không gian lưu trữ:

##### Mỗi số nguyên p, q, n, e, d: O(k) bit.

##### Tổng: O(k), với k là số bit của modulus n.

|  |  |  |
| --- | --- | --- |
| Hoạt động | Độ phức tạp thời gian | Độ phức tạp không gian |
| Tạo khóa | O( | O(k) |
| Mã hóa | O() | O(k) |
| Giải mã | O( hoặc O (với CRT) | O(k) |

RSA có độ phức tạp thời gian cao, đặc biệt trong giai đoạn giải mã vì cần thực hiện các phép tính với số mũ lớn. Việc tối ưu RSA bằng CRT giúp giảm đáng kể chi phí tính toán trong quá trình giải mã. RSA thường không được sử dụng để mã hóa dữ liệu lớn trực tiếp, thay vào đó, nó được dùng để mã hóa khóa đối xứng trong các hệ thống lai.

### Ưu điểm và nhược điểm của mã hóa RSA

#### ****Ưu điểm của RSA****

**Bảo mật cao:** RSA là một trong những thuật toán mã hóa bất đối xứng phổ biến và an toàn nhất hiện nay. Độ an toàn của RSA dựa trên bài toán phân tích số nguyên nnn (là tích của hai số nguyên tố lớn ppp và qqq) thành các thừa số nguyên tố. Đây là một bài toán rất khó và hiện tại chưa có giải thuật hiệu quả nào giải quyết được trong thời gian hợp lý khi ppp và qqq đủ lớn (thường trên 1024 hoặc 2048 bit). Điều này làm cho RSA trở thành lựa chọn phù hợp trong các ứng dụng bảo mật yêu cầu mức độ an toàn cao, chẳng hạn như giao thức HTTPS, ngân hàng điện tử, hoặc trao đổi khóa trong mã hóa đối xứng.

**Tính bất đối xứng:** Một ưu điểm quan trọng của RSA là tính bất đối xứng, tức là không cần chia sẻ khóa bí mật giữa các bên. Người gửi sử dụng khóa công khai (e,n)(e, n)(e,n) để mã hóa, trong khi chỉ người nhận có khóa bí mật (d,n)(d, n)(d,n) mới có thể giải mã thông điệp. Điều này loại bỏ các rủi ro liên quan đến việc trao đổi và bảo mật khóa trong các thuật toán mã hóa đối xứng, nơi cả hai bên đều phải sử dụng chung một khóa bí mật.

**Tích hợp chữ ký số:** RSA không chỉ được sử dụng để mã hóa thông tin mà còn hỗ trợ việc tạo và xác minh chữ ký số. Trong trường hợp này, người gửi sử dụng khóa bí mật để ký vào thông điệp, và người nhận sử dụng khóa công khai để xác minh chữ ký. Điều này giúp đảm bảo tính xác thực (authenticity) và tính toàn vẹn (integrity) của thông tin, đồng thời chứng minh được nguồn gốc của thông tin. Khả năng này làm cho RSA trở thành giải pháp lý tưởng trong các hệ thống yêu cầu bảo mật và chứng thực, chẳng hạn như trong giao dịch tài chính hoặc các văn bản pháp lý.

#### ****Nhược điểm của RSA****

**Tốc độ chậm:** RSA thường chậm hơn rất nhiều so với các thuật toán mã hóa đối xứng như AES. Nguyên nhân chính là RSA thực hiện các phép toán số mũ modulo lớn, yêu cầu nhiều tài nguyên tính toán hơn. Điều này khiến RSA không phù hợp để mã hóa lượng lớn dữ liệu. Thay vào đó, RSA thường được sử dụng để mã hóa các khóa bí mật (key) trong các hệ thống mã hóa kết hợp (hybrid encryption system), sau đó sử dụng các thuật toán đối xứng như AES để mã hóa dữ liệu thực tế.

**Yêu cầu kích thước khóa lớn:** Để đảm bảo tính bảo mật trước các tấn công hiện đại (chẳng hạn như tấn công bằng máy tính lượng tử trong tương lai), RSA đòi hỏi kích thước khóa rất lớn, thường từ 2048 bit trở lên. Kích thước khóa lớn làm tăng chi phí lưu trữ, truyền tải và xử lý dữ liệu, gây khó khăn trong các hệ thống hạn chế tài nguyên như thiết bị IoT hoặc hệ thống nhúng.

**Dễ bị tấn công nếu không chọn số nguyên tố lớn hoặc không bảo mật khóa bí mật:** RSA phụ thuộc hoàn toàn vào việc lựa chọn các số nguyên tố lớn và bí mật. Nếu ppp và qqq được chọn không đủ lớn hoặc không đủ ngẫu nhiên, kẻ tấn công có thể dễ dàng phân tích số nnn thành các thừa số nguyên tố. Ngoài ra, việc để lộ khóa bí mật hoặc sử dụng cùng một khóa trong nhiều hệ thống khác nhau sẽ làm giảm tính bảo mật, dễ bị tấn công bởi các phương pháp như **tấn công thời gian** (timing attack), **tấn công phần cứng** (hardware attack), hoặc **tấn công brute-force**.

### Ứng dụng của mã hóa RSA

Nhìn chung, RSA đã mở ra kỷ nguyên mới cho chữ ký số, mang lại sự an toàn và đáng tin cậy cho giao tiếp và giao dịch trong thế giới kỹ thuật số. Dưới đây là một số ứng dụng RSA ở thời điểm hiện tại, theo dõi chi tiết bên dưới.

#### Chữ ký số

RSA là thuật toán tiên phong trong việc tạo ra chữ ký số, đánh dấu cột mốc quan trọng trong lĩnh vực bảo mật và khoa học dữ liệu. Nhờ cơ chế mã hóa bất đối xứng, RSA không chỉ đảm bảo tính bảo mật mà còn giúp kiểm tra tính toàn vẹn của dữ liệu – một yếu tố quan trọng trong các giao dịch số.

Chữ ký số giống như chữ ký tay nhưng hoạt động trong môi trường điện tử và hiện nay đã trở thành công cụ không thể thiếu trong các hợp đồng và giao dịch trực tuyến. Dù ngày càng phổ biến, việc cài đặt và sử dụng chữ ký số vẫn còn nhiều thách thức, khiến việc triển khai thực tế còn nhiều bất cập, dù việc đảm bảo tính xác thực và bảo mật khá ổn định.

#### Bảo mật kết nối trên web, email, VPN và các ứng dụng chat

RSA thường được sử dụng để mã hóa nội dung trong các ứng dụng chat, giúp bảo vệ các cuộc trò chuyện cá nhân và dữ liệu nhạy cảm. Bằng cách sử dụng cặp khóa công khai và khóa riêng (public key và private key), RSA đảm bảo rằng chỉ những người có quyền truy cập vào khóa riêng mới có thể giải mã dữ liệu được mã hóa. Điều này giúp ngăn chặn việc nghe lén, đánh cắp thông tin trên đường truyền bởi các hacker, từ đó bảo vệ sự riêng tư và bảo mật của người dùng trong các môi trường giao tiếp số.

#### Giao thức TLS/SSL

Khác đôi chút với các ứng dụng phía trên RSA đóng vai trò trong việc thiết lập kênh truyền tải an toàn giữa máy chủ và client trong TLS/SSL. Tại đây, RSA được sử dụng để mã hóa quá trình trao đổi khóa, đảm bảo rằng dữ liệu truyền tải qua internet, chẳng hạn như thông tin đăng nhập, thanh toán hoặc dữ liệu cá nhân luôn được bảo mật tuyệt đối. Nhờ có RSA, người dùng có thể truy cập các trang web với giao thức HTTPS và có thể yên tâm trước các cuộc tấn công man-in-the-middle và các hành vi đánh cắp dữ liệu khác. RSA là một thành phần quan trọng giúp duy trì sự an toàn và bảo mật của các giao dịch trực tuyến thông qua TLS/SSL.

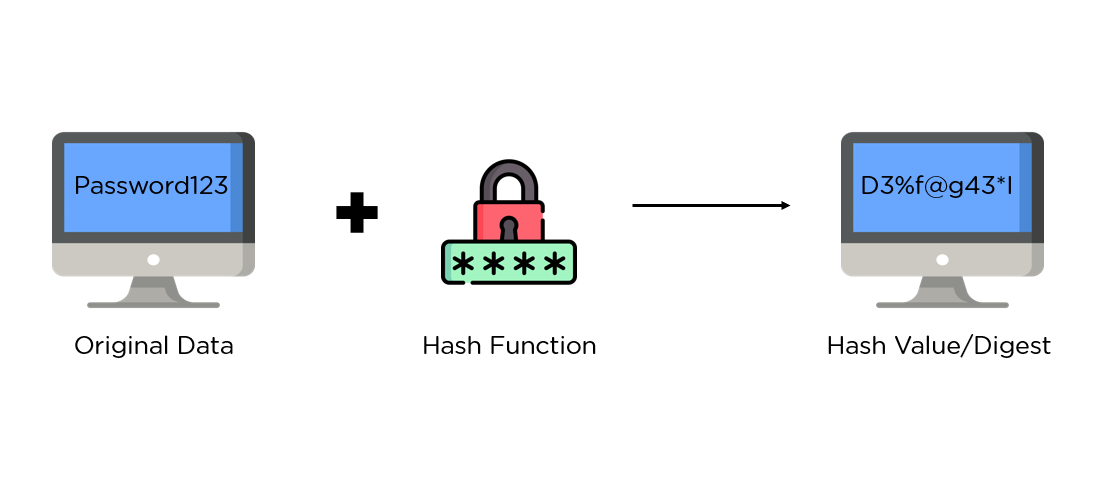
## Giải thuật SHA256

### Giới thiệu

Trong số nhiều tiến bộ được thấy trong bảo mật mạng, [mã hóa](https://www.simplilearn.com/data-encryption-methods-article) và băm là những nguyên tắc cốt lõi của các mô-đun bảo mật bổ sung. Thuật toán băm an toàn với kích thước tóm tắt là 256 bit, hay thuật toán SHA 256, là một trong những thuật toán băm được sử dụng rộng rãi nhất. Mặc dù có những biến thể khác, SHA 256 đã đi đầu trong các ứng dụng thực tế.

#### Hashing

Hashing là quá trình xáo trộn thông tin thô đến mức không thể tái tạo lại thành dạng ban đầu. Nó lấy một phần thông tin và truyền thông tin đó qua một hàm thực hiện các phép toán trên văn bản thuần túy. Hàm này được gọi là hàm băm và đầu ra được gọi là giá trị băm/bản tóm tắt.



Hình 12. Hoạt động của hàm băm

 Như được thấy từ hình ảnh trên, hàm băm có trách nhiệm chuyển đổi văn bản thuần túy thành bản tóm tắt băm tương ứng. Chúng được thiết kế để không thể đảo ngược, nghĩa là bản tóm tắt của bạn không nên cung cấp cho bạn văn bản thuần túy gốc bằng bất kỳ cách nào cần thiết. Hàm băm cũng cung cấp cùng một giá trị đầu ra nếu đầu vào không thay đổi, bất kể số lần lặp lại.

#### Thuật toán SHA-256

SHA 256 là một phần của họ thuật toán SHA 2, trong đó SHA là viết tắt của Secure Hash Algorithm. Được công bố vào năm 2001, đây là nỗ lực chung giữa NSA và NIST để giới thiệu một thế hệ kế nhiệm cho họ SHA 1, vốn đang dần mất đi sức mạnh trước [các cuộc tấn công bằng vũ lực.](https://www.simplilearn.com/tutorials/cryptography-tutorial/brute-force-attack)

Ý nghĩa của số 256 trong tên tượng trưng cho giá trị băm cuối cùng, nghĩa là bất kể kích thước của văn bản thuần túy/văn bản rõ, giá trị băm sẽ luôn là 256 bit.

Các thuật toán khác trong họ SHA ít nhiều giống với SHA 256. Bây giờ, hãy tìm hiểu thêm một chút về các nguyên tắc của chúng.

Bắt đầu hành trình chuyển đổi thông qua [Cyber ​​security Bootcamp](https://www.simplilearn.com/cybersecurity-bootcamp) của chúng tôi , nơi bạn sẽ đi sâu vào sự phức tạp của các công nghệ tiên tiến như thuật toán SHA-256. Khám phá các nguyên tắc mật mã khiến thuật toán này trở thành nền tảng của bảo mật blockchain, đồng thời rèn luyện các kỹ năng của bạn trong việc phòng thủ chống lại các mối đe dọa mạng.

### Các bước thực hiện giải thuật

SHA-256 chuyển đổi một thông điệp bất kỳ thành một chuỗi băm cố định dài 256 bit. Giải thuật bao gồm các bước sau:

**Bước 1: Đệm thông điệp (Message Padding)**

Thông điệp đầu vào sẽ được mở rộng để độ dài của nó là bội số của 512 bit, bao gồm:

##### **Thêm một bit '1':** Thêm một bit '1' ngay sau thông điệp gốc.

##### **Thêm các bit '0':** Sau đó, thêm các bit '0' cho đến khi độ dài của thông điệp đạt gần tới bội số của 512 bit, chỉ còn lại 64 bit cuối.

##### **Thêm độ dài thông điệp:** Cuối cùng, thêm độ dài của thông điệp gốc (trước khi đệm) vào 64 bit cuối (biểu diễn dưới dạng nhị phân).

Ví dụ: Một thông điệp 448 bit sẽ được thêm 1 bit '1', 63 bit '0', và 64 bit để biểu diễn độ dài, tạo thành 512 bit.

**Bước 2: Chia khối (Message Parsing)**

Sau khi đệm, thông điệp được chia thành các khối 512 bit. Nếu thông điệp sau khi đệm có L bit, thì số khối sẽ là:

|  |
| --- |
| N = L / 512. |

**Bước 3: Khởi tạo hằng số (Initialization)**

SHA-256 sử dụng tám giá trị hằng ban đầu (hằng số khởi tạo) 32-bit, được biểu diễn dưới dạng thập lục phân:

|  |
| --- |
| =6a09e667, =bb67ae85, =3c6ef372, =a54ff53a, =510e527f, =9b05688c, =1f83d9ab, =5be0cd19 |

Bước 4: Mở rộng từ khóa (Message Schedule Array)

Với mỗi khối 512 bit, giải thuật tạo ra một mảng 64 từ (word) 32 bit, ký hiệu là W[0] đến W[63], với các bước:

##### Tách khối đầu vào thành 16 từ đầu tiên W[0] đến W[15], mỗi từ có 32 bit.

##### Với i=16 đến 63, tính W[i] bằng công thức:

|  |
| --- |
| W[i]=(W[i−2]) + W[i−7] + (W[i−15]) + W[i−16]W[i] |

Với:

##### (x) = (x≫7) ⊕ (x≫18) ⊕ (x≫3)

##### (x) = (x≫17) ⊕ (x≫19) ⊕ (x≫10)

##### Ký hiệu ≫ là dịch vòng phải, ⊕ là XOR.

**Bước 5: Nén khối (Compression Function)**

Với mỗi khối 512 bit, thuật toán thực hiện 64 vòng lặp. Ở mỗi vòng, thuật toán sử dụng tám biến trạng thái a, b, c, d, e, f, g, h (ban đầu được khởi tạo từ ​ đến ​) và một tập hợp các hằng số vòng K[i], với i=0 đến 63.

**Bước nén bao gồm:**

##### Tính các giá trị trung gian:

|  |
| --- |
| =(e≫6) ⊕ (e≫11) ⊕ (e≫25)  Ch = (e∧f) ⊕ (¬e∧g)  T1 = h + + Ch + K[i] + W[i]  = (a≫2) ⊕ (a≫13) ⊕ (a≫22)  Maj = (a∧b) ⊕ (a∧c) ⊕ (b∧c)  T2 = + Maj |

##### Cập nhật các giá trị trạng thái:

|  |
| --- |
| h = g, g = f, f = e, e = d + T1  d = c, c = b, b = a, a = T1 + T2 |

Bước 6: Cập nhật giá trị băm (Hash Update):

Sau khi hoàn thành 64 vòng lặp, các giá trị trung gian a, b, c, d, e, f, g, h sẽ được cộng với giá trị băm trước đó (, ​,..., ​):

|  |
| --- |
| =+a, =+b, ..., =+h |

Bước 7: Xuất giá trị băm (Output)

Kết thúc xử lý tất cả các khối, thuật toán hợp nhất , ​,..., ​ thành một chuỗi băm 256 bit, đây chính là kết quả của SHA-256.

### Cấu trúc dữ liệu của SHA-256

#### Dữ liệu đầu vào (Input Data):

Chuỗi nhị phân có độ dài tùy ý.

#### Khối dữ liệu (Message Blocks):

##### Chia dữ liệu sau khi padding thành các khối 512-bit.

##### Mỗi khối chứa 161616 từ (word), mỗi từ dài 32 bit.

#### Lịch trình mở rộng thông điệp (Message Schedule):

Mảng W[64]:

##### W[0] đến W[15]: Trích xuất từ khối dữ liệu.

##### W[16] đến W[63]: Tính toán từ W[0] đến W[15].

#### Biến trạng thái (State Variables):

##### Gồm 8 biến h0​,h1​,h2​,h3​,h4​,h5​,h6​,h7​, mỗi biến dài 32 bit.

##### Khởi tạo bằng giá trị hằng số.

#### Hằng số vòng (Round Constants):

Mảng K[64]: 64 hằng số cố định, mỗi hằng số dài 32 bit.

#### Biến tạm thời (Temporary Variables):

a, b, c, d, e, f, g, h: 8 biến tạm, mỗi biến dài 32 bit.

#### Đầu ra (Output Hash):

Giá trị băm cuối cùng: Chuỗi nhị phân dài 256-bit (gồm 8 biến trạng thái nối lại).

|  |  |  |
| --- | --- | --- |
| Thành phần | Kích thước | Vai trò |
| Dữ liệu đầu vào | Tùy ý | Chuỗi cần băm |
| Khối dữ liệu 512-bit | 512 bit | Từng khối dữ liệu đầu vào sau khi padding |
| Lịch trình W[64] | 64 từ, 32 bit/word | Lưu trữ thông điệp mở rộng |
| Biến trạng thái h[i] | 8 biến, 32 bit/biến | Giá trị trung gian trong quá trình tính toán |
| Hằng số K[64] | 64 hằng, 32 bit/hằng | Giá trị cố định dùng trong mỗi vòng nén |
| Biến tạm a-h | 8 biến, 32 bit/biến | Lưu trạng thái tạm thời trong từng vòng xử lý |
| Đầu ra | 256 bit | Chuỗi băm cuối cùng, cố định 256 bit |

### Độ phức tạp của thuật toán

#### Độ phức tạp thời gian (Time Complexity):

SHA-256 xử lý dữ liệu qua các bước chính, mỗi bước có độ phức tạp thời gian riêng:

**Bước Padding và Chia Khối:**

##### Padding thêm các bit để làm dữ liệu đầu vào có độ dài là bội số của 512.

##### Chia dữ liệu thành các khối 512-bit.

##### Độ phức tạp: O(n), với n là số bit của dữ liệu đầu vào.

**Mở rộng thông điệp (Message Schedule):**

##### Mỗi khối 512-bit được mở rộng thành 646464 từ, với mỗi từ 32-bit.

##### Mỗi từ từ W[16] đến W[63] được tính toán dựa trên các từ trước đó.

##### Độ phức tạp: O(64) cho mỗi khối, tức O(m . 64) với m là số khối (bằng ⌈n/512⌉).

**Vòng nén (Compression Rounds):**

##### SHA-256 thực hiện 64 vòng tính toán cho mỗi khối.

##### Mỗi vòng thực hiện nhiều phép toán (xor, dịch vòng, cộng, AND, OR).

##### Độ phức tạp: O(64) cho mỗi khối, tức O(m . 64).

**Kết hợp trạng thái:**

##### Sau khi xử lý tất cả các khối, trạng thái cuối cùng được kết hợp.

##### Độ phức tạp: O(1).

**Tổng quát:** T(n) = O(n) + O(m.64) = O(n) + O(⌈n/512⌉⋅64)

Với m=⌈n/512⌉, độ phức tạp thời gian là: T(n)=O(n)

#### Độ phức tạp không gian (Space Complexity):

SHA-256 yêu cầu bộ nhớ để lưu trữ các cấu trúc dữ liệu:

**Biến trạng thái:**

##### 8 biến h0​,h1​,…,h7​, mỗi biến dài 32 bit.

##### Không gian: 8×32=256 bit.

**Khối dữ liệu:**

##### Lưu một khối dữ liệu đầu vào (512-bit) và mở rộng thành 64 từ (mỗi từ 32-bit).

##### Không gian: 64×32=2048 bit.

**Hằng số vòng:**

##### 64 hằng số K[i], mỗi hằng dài 32-bit.

##### Không gian: 64×32=2048 bit.

**Biến tạm thời:**

##### 8 biến tạm thời a, b,…, h, mỗi biến dài 32-bit.

##### Không gian: 8×32=256 bit.

**Tổng không gian:** S=256+2048+2048+256=4608 bit=576 byte

**Độ phức tạp không gian:** O(1), vì không gian không phụ thuộc vào kích thước dữ liệu đầu vào n.

|  |  |
| --- | --- |
| Loại độ phức tạp | Giá trị |
| Độ phức tạp thời gian | O(n) |
| Độ phức tạp không gian | O(1) |

Tóm lại, SHA-256 được thiết kế để xử lý hiệu quả dữ liệu lớn với độ phức tạp không gian cố định và thời gian tuyến tính theo kích thước đầu vào.

### ****Tính chất của SHA-256****

**Tính duy nhất (Uniqueness):** SHA-256 đảm bảo rằng hai thông điệp khác nhau, dù chỉ khác một bit, sẽ tạo ra hai chuỗi băm hoàn toàn khác nhau (đặc tính avalanche effect). Điều này rất quan trọng trong việc phát hiện thay đổi dữ liệu và đảm bảo tính toàn vẹn thông tin. Xác suất để hai thông điệp khác nhau có cùng giá trị băm (va chạm) là cực kỳ thấp, nhờ vào chiều dài 256 bit của chuỗi băm.

**Tính bất đối xứng (Irreversibility):** SHA-256 được thiết kế để không thể suy ngược lại thông điệp gốc từ giá trị băm. Tính chất này dựa trên các phép toán phức tạp như XOR, dịch vòng và các phép toán phi tuyến tính, giúp ngăn chặn việc giải mã chuỗi băm bằng cách tính toán ngược.

**Tốc độ cao (Efficiency):** SHA-256 hoạt động hiệu quả với dữ liệu lớn nhờ cách chia thông điệp thành các khối nhỏ 512 bit và xử lý tuần tự. Điều này giúp nó được áp dụng rộng rãi trong các ứng dụng cần băm dữ liệu nhanh chóng, chẳng hạn như xác thực người dùng hoặc kiểm tra tính toàn vẹn của dữ liệu trong thời gian thực.

**Chống tấn công va chạm (Collision Resistance):** Thuật toán SHA-256 được thiết kế để làm giảm nguy cơ xảy ra tấn công va chạm, tức là hai thông điệp khác nhau nhưng tạo ra cùng một giá trị băm. Với không gian đầu ra lên tới 22562^{256}2256 giá trị, khả năng xảy ra va chạm là vô cùng thấp, ngay cả khi xét đến các tấn công mạnh như tấn công ngày sinh (birthday attack).

### ****Ứng dụng của SHA-256****

**Chữ ký số và chứng thực (Digital Signature and Authentication):** SHA-256 được sử dụng để tạo chữ ký số trong các hệ thống bảo mật như SSL/TLS, giúp xác minh tính xác thực của người gửi và đảm bảo rằng dữ liệu không bị thay đổi trong quá trình truyền tải. Chữ ký số thường sử dụng kết hợp SHA-256 và RSA hoặc ECDSA để tạo ra một hệ thống bảo mật mạnh mẽ.

**Bảo mật mật khẩu (Password Hashing):** SHA-256 được sử dụng để mã hóa mật khẩu trước khi lưu trữ vào cơ sở dữ liệu. Việc mã hóa mật khẩu giúp bảo vệ thông tin người dùng ngay cả khi cơ sở dữ liệu bị xâm nhập. Kết hợp với các kỹ thuật như "salt" (chuỗi ngẫu nhiên thêm vào mật khẩu) giúp tăng cường khả năng chống lại các cuộc tấn công brute force hoặc rainbow table.

**Giao dịch blockchain:** SHA-256 là thuật toán băm chính trong các giao thức blockchain như Bitcoin. Trong Bitcoin, SHA-256 được dùng để:

##### Xác thực các giao dịch.

##### Đảm bảo tính toàn vẹn của khối (block).

##### Tạo hàm băm PoW (Proof of Work) để giải quyết các bài toán khai thác mỏ (mining).

**Kiểm tra tính toàn vẹn của tệp dữ liệu (Data Integrity Check):**  
SHA-256 được sử dụng để tạo "checksum" cho các tệp tin. Bằng cách so sánh giá trị SHA-256 được tính toán từ tệp tải về với giá trị được cung cấp từ nguồn gốc, người dùng có thể kiểm tra xem tệp tin có bị thay đổi hoặc hỏng hóc trong quá trình truyền tải hay không.

**Hệ thống mã hóa và bảo mật:** SHA-256 là một phần quan trọng trong các giao thức mã hóa như IPsec, S/MIME, và PGP. Nó giúp tăng cường độ an toàn cho các hệ thống email, mạng riêng ảo (VPN), và truyền thông bảo mật.

**Lưu trữ blockchain phi tập trung:** SHA-256 cũng được áp dụng trong các hệ thống lưu trữ phi tập trung (decentralized storage) để tạo ra các giá trị định danh duy nhất cho dữ liệu. Ví dụ, khi dữ liệu được lưu trên các nền tảng như IPFS, SHA-256 được sử dụng để tạo hash định danh duy nhất, giúp truy xuất nhanh chóng và xác minh tính toàn vẹn dữ liệu.

Nhờ các tính chất và ứng dụng mạnh mẽ, SHA-256 trở thành một công cụ không thể thiếu trong lĩnh vực bảo mật thông tin hiện đại.

# MÔ PHỎNG VÀ TRIỂN KHAI THỰC NGHIỆM

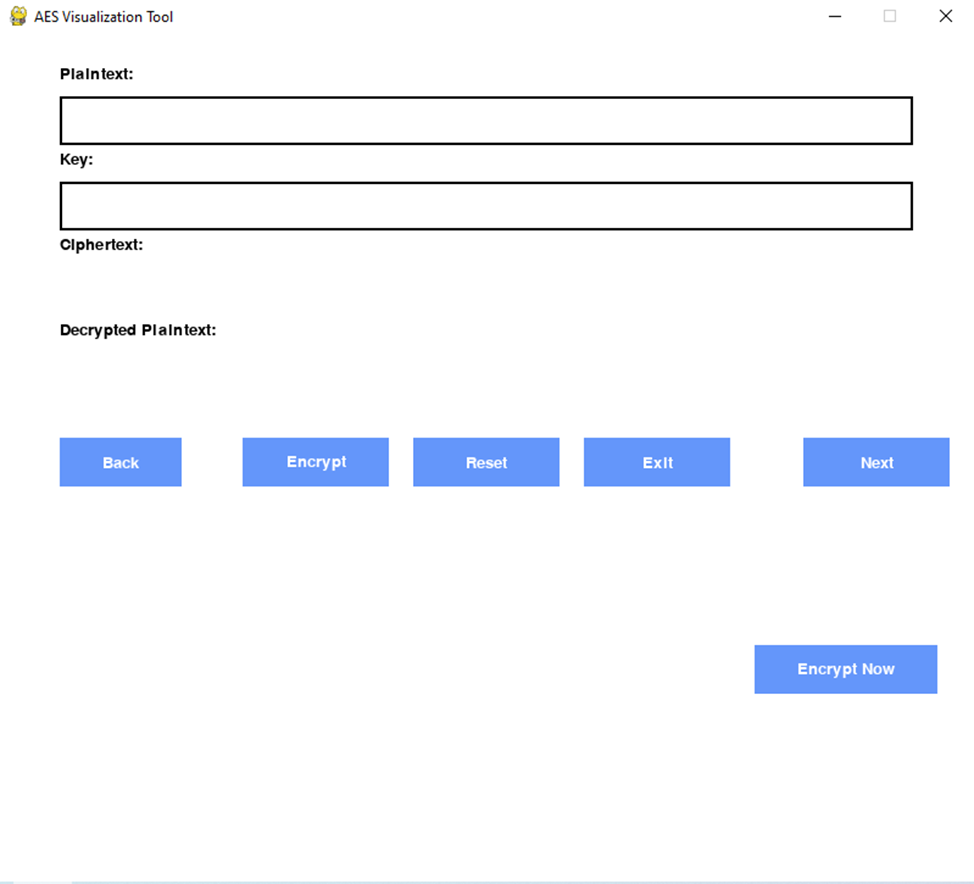
## Công cụ và môi trường thực hiện

## Mô phỏng các thuật toán mã hóa

### Mã hóa AES

#### Mã hóa

Đây là giao diện khi chạy chương trình:



Hình 13. Giao diện mô phỏng giải thuật AES

Trong đó: Plaintext và key là nơi nhập trường thông tin, plaintext là thông tin cần được mã hoá, key là thông tin đóng vai trò trong quá trình mã hoá và giải mã, ciphertext là kết quả sau khi mã hoá, còn decrypted plaintext là kết quả của việc mã giá trị ciphertext.

Các nút màu xanh trong giao diện có chức năng như sau:

##### Nút Encrypt thực hiện việc mã hoá, và nó sẽ tự động chuyển sang nút decrypt neues như thực hiện xong phần mã hoá.

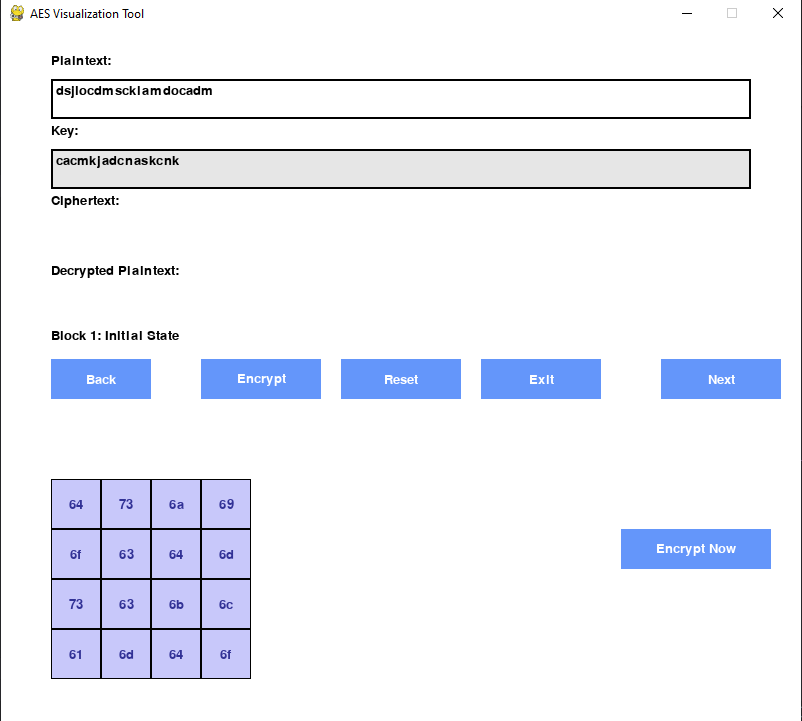
##### Nút Reset dùng để xoá toàn bộ các thông tin có trên giao diện và thực hiện một chu trình trực quan thuật toán mới.

##### Nút exit dùng để thoát khỏi ứng dụng.

##### Nút Back và Next dùng để theo dõi từng bước quá trình mã hoá và giải mã

##### Nút Encrypt Now dùng để cho ra kết quả mã hoá trực tiếp thay vì theo dõi từng bước bằng nút Next, sau khi thực hiện xong bước mã hoá thì nút đó sẽ được chuyển thành nút Decrypt Now và nút đó có chức năng cho ra kết quả mã hoá ngay lập tức.

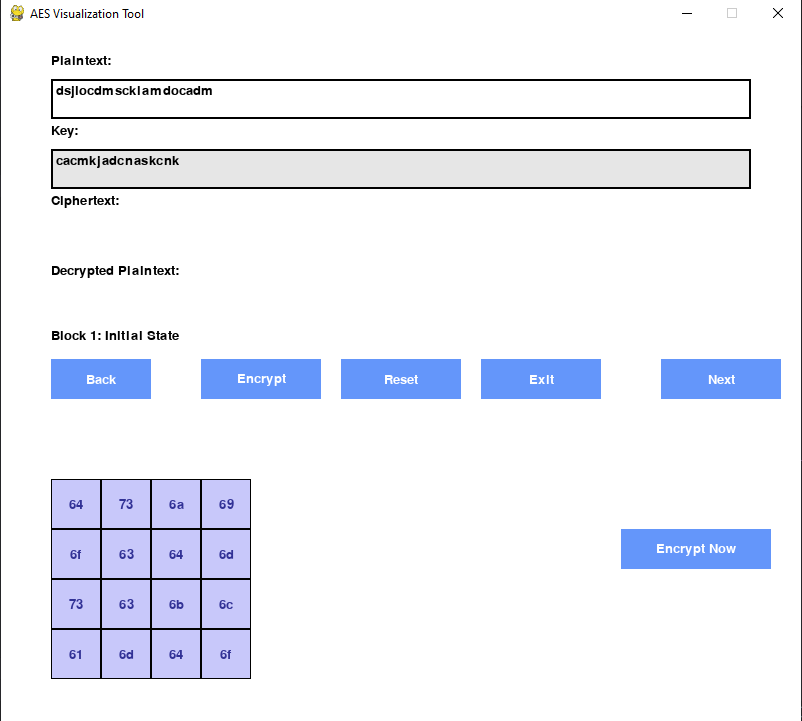
Tiến hành nhập giá trị bất kỳ cho plaintext và khoá



Hình 14. Giao diện Plaintext và Key khi nhập chuỗi giá trị bất kỳ

Ở đây giá trị plaintext không giới hạn số lượng ký tự nhập vào, còn giá trị key sẽ giới hạn 16 ký tự tương ứng với độ dài khoá là 128 bit.

Sau khi nhập xong các trường thông tin trên, đến bước mã hoá. Nhấn nút Encrypt:

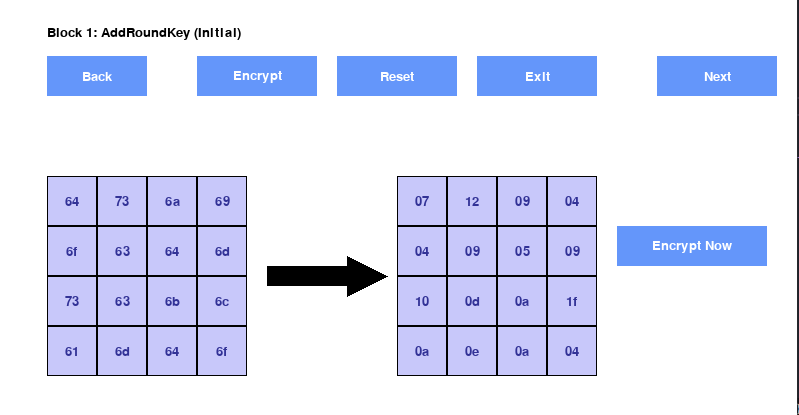


Hình 15. Khởi tạo ma trận trạng thái

Sau khi nhấn nút Encrypt thì một khối ma trận 4x4 hiện ra thể hiện trạng thái đầu tiên của mã hoá, các thông tin được thể hiện như là Block 1 thể hiện thông tin của 16 ký tự đầu tiên của plaintext đã được chuyển dang dạng UTF-8 và thể hiện bằng ma trận. Ở đây nếu plaintext dài hơn 16 ký tự thì sẽ được chia ra thành các block để tiến hành mã hoá từng block, nếu block nào còn thiếu dữ liệu sẽ được padding thêm giá trị là số byte cần thêm để lấp đầy khối cuối cùng.

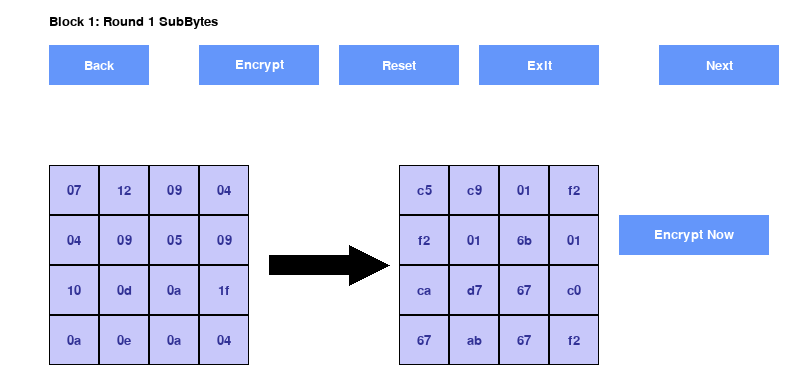
Tiếp tục nhấn next để theo dõi các bước của phần mã hoá:

Bước Add Round Key:



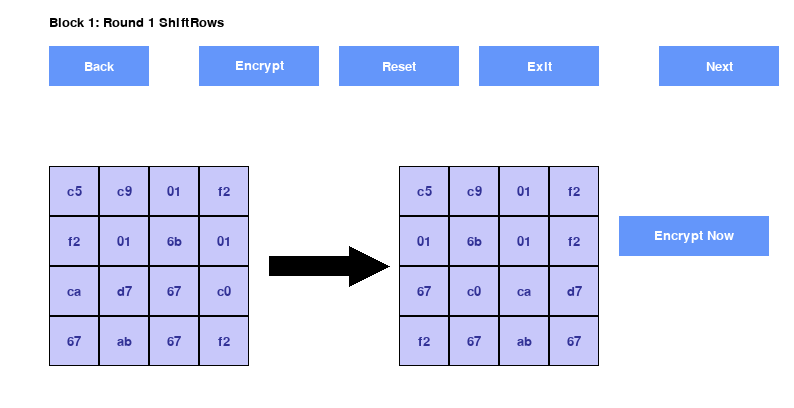
Hình 16. Bước Add Round Key

Bước Sub Bytes:



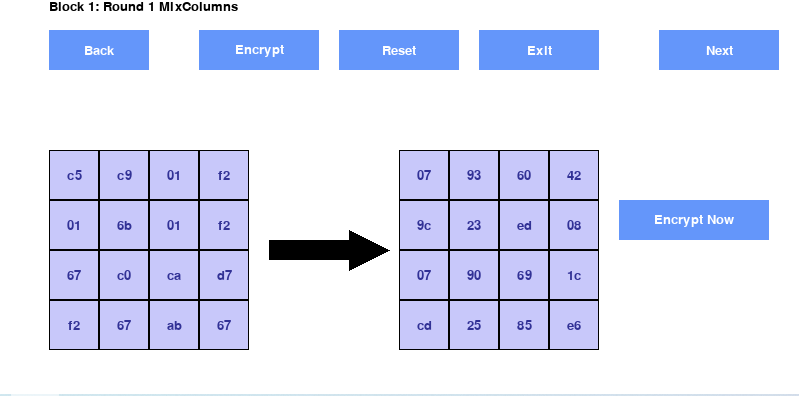
Hình 17. Bước Sub Bytes

Bước Shift Rows:



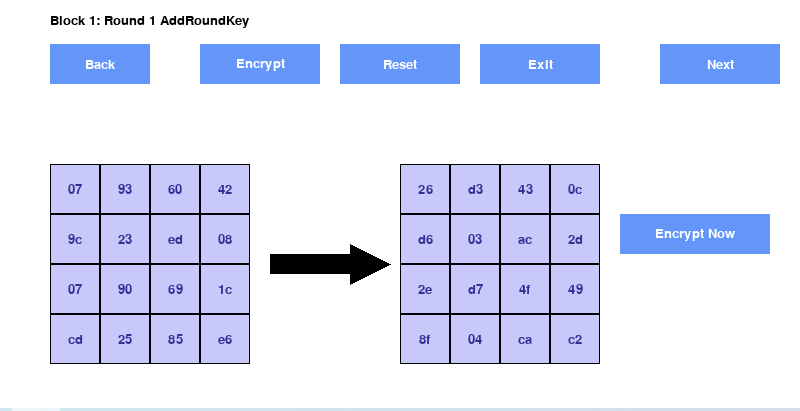
Hình 18. Bước Shift Row của vòng thứ nhất

Bước Mix Column:



Hình 19. Bước Mix Columns của vòng thứ nhất

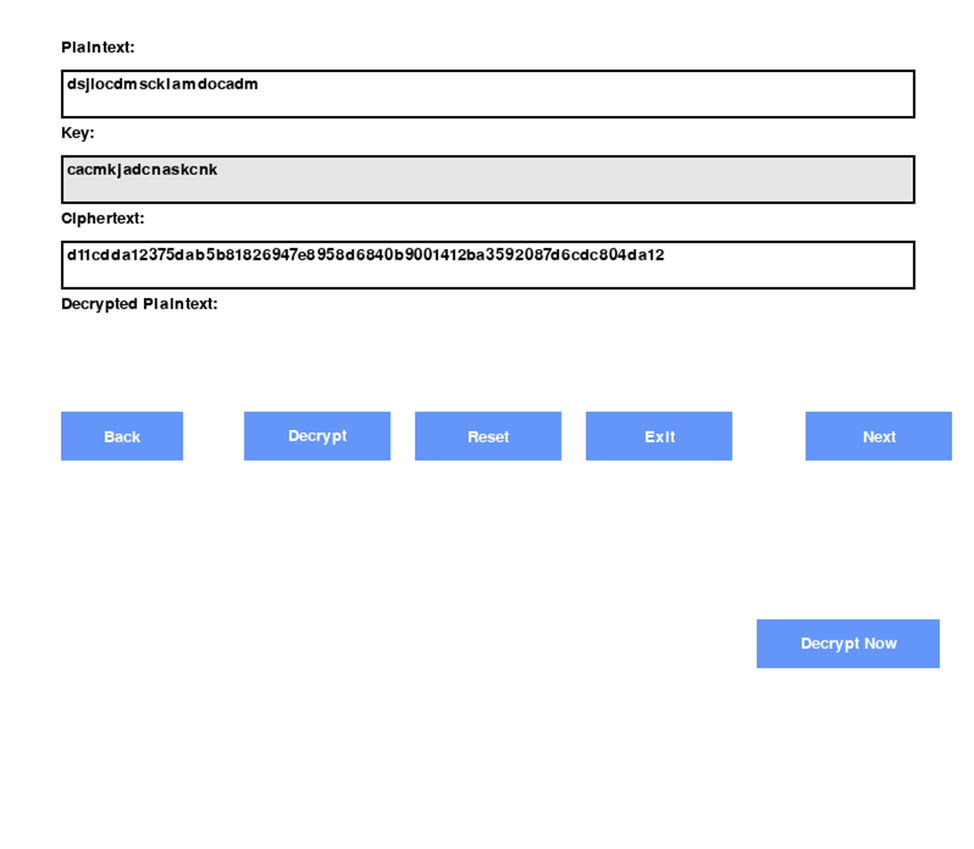
Bước Add Round Key:



Hình 20. Bước Add Round Key của vòng thứ nhất

Theo quy trình của giải mã AES, sẽ có 9 bước thức hiện lặp lại một chu trình các phần như trên, đến bước thứ 10, thì sẽ bỏ phần Mix Column đi.

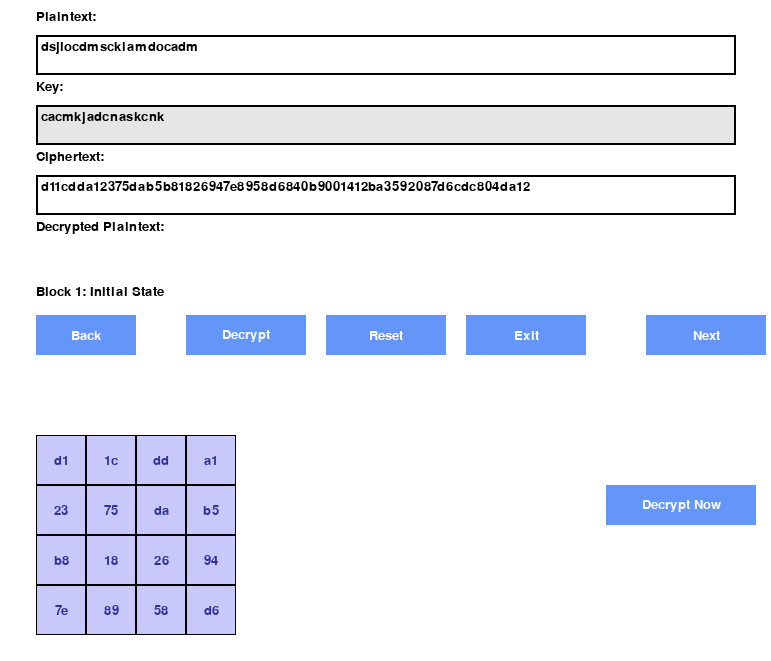
Ngoài ra nút Encrypt Now sẽ giúp bạn cho ra kết quả của phần mã hoá luôn, đồng thời các nút Encrypt và Encrypt Now được chuyển thành Decrypt và Decrypt Now do đã thực hiện xong quá trình mã hoá.



Hình 21. Kết quả sau khi ấn nút Encrypt Now

#### Giải mã

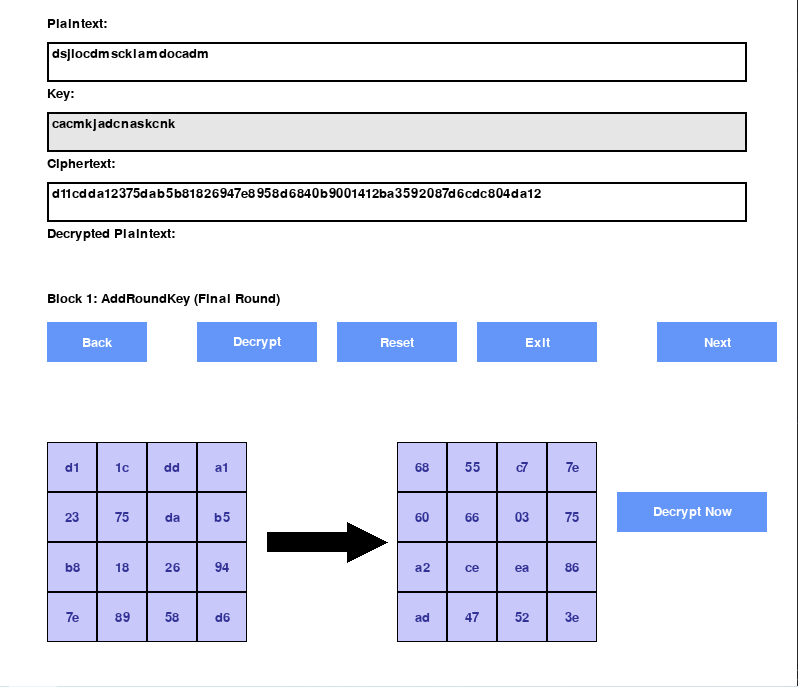
Tiếp tục đến phần giải mã, nhấn nút Decrypt và theo dõi quá trình giải mã như sau:



Hình 22. Giao diện sau khi ấn nút Decrypt

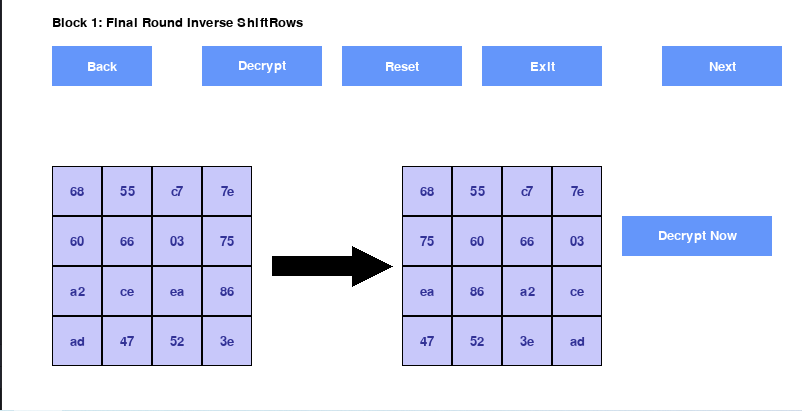
Lấy 16 khối byte trong Ciphertext và đưa vào một ma trận 4x4

Tiếp đến là bước Add Round Key



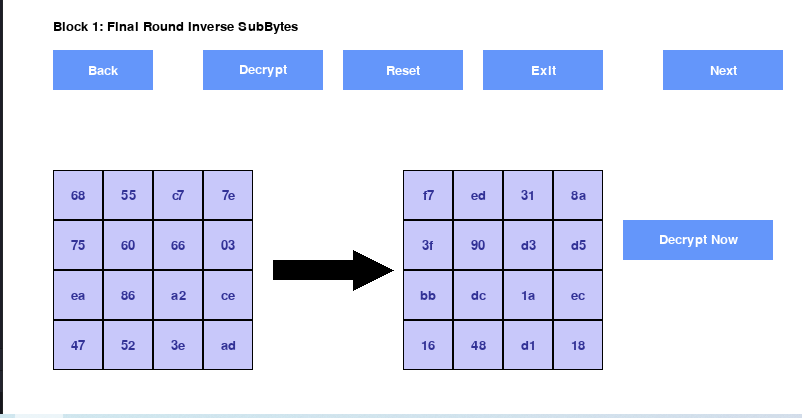
Hình 23. Bước Add Round Key

Bước Inverse Shift Row:



Hình 24. Bước Inverse Shift Rows

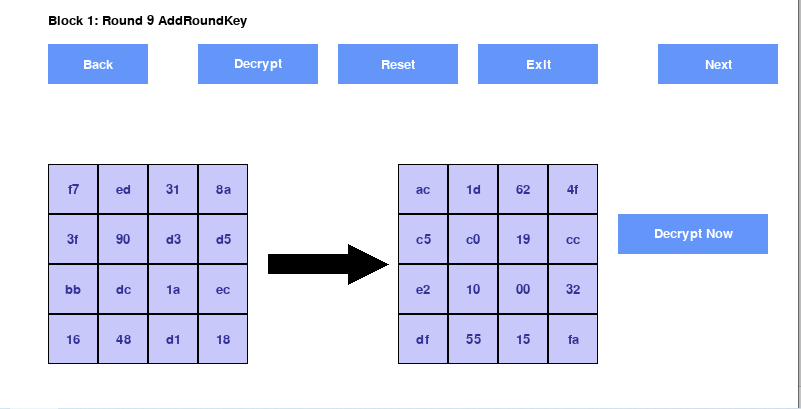
Bước Inverse Sub Byte:



Hình 25. Bước Inverse Sub Bytes

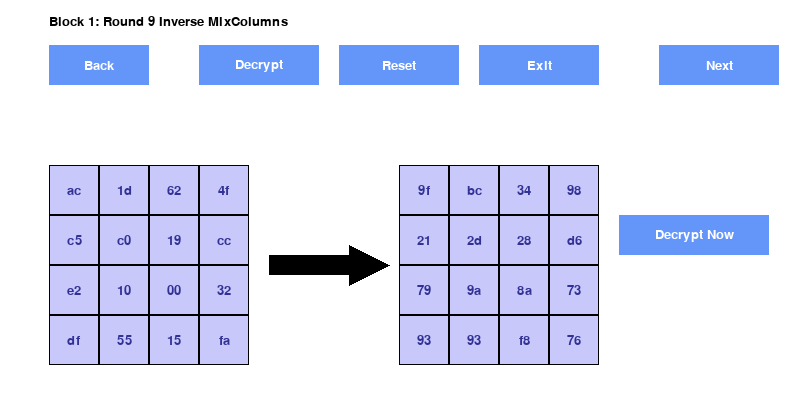
Do quá trình mã hoá là ngược lại của giải mã nên bước đầu tiên sẽ gồm có Add Round Key, Inverse Shift Row và Inverse Sub Byte. Sau đó là thực hiện 9 lần liên tiếp một chu trình gồm các phần như sau:

Bước Add Round Key ở vòng lặp thứ 9:



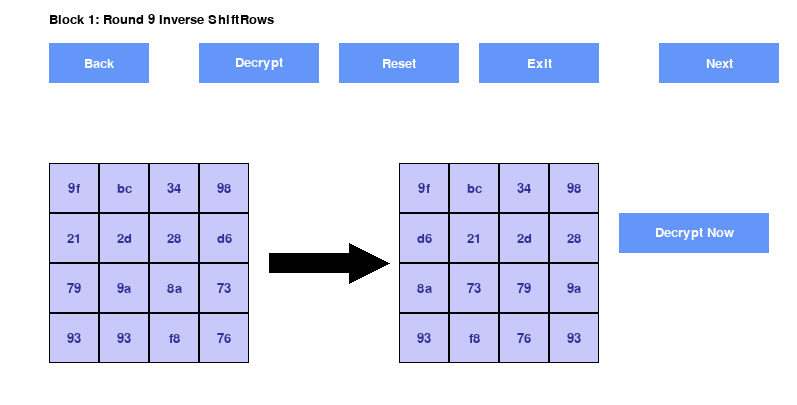
Hình 26. Bước Add Round Key ở vòng lặp thứ 9

Bước Inverse Mix Column ở vòng lặp thứ 9:



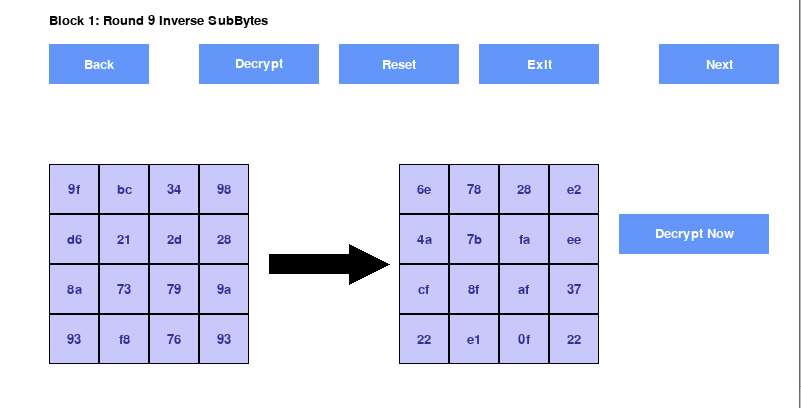
Hình 27. Bước Inverse Mix Columns ở vòng lặp thứ 9

Bước Inverse Shift Row ở vòng lặp thứ 9:



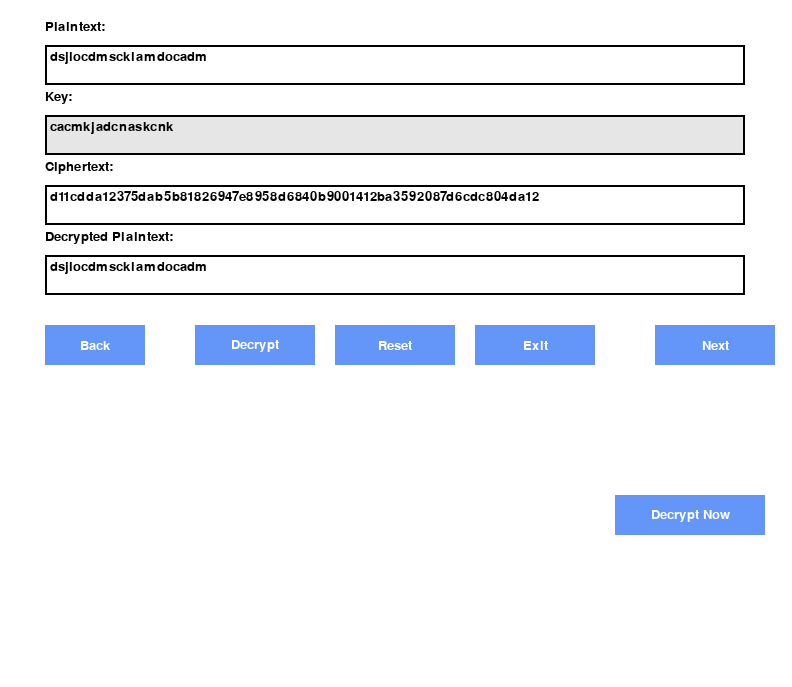
Hình 28. Bước Inverse Shift Rows ở vòng lặp thứ 9

Bước Inverse Sub Bytes ở vòng lặp thứ 9:



Hình 29. Bước Inverse Sub Bytes ở vòng lặp thứ 9

Chạy hết 9 vòng, ta đến bước cuối cũng Add Round Key, sau đó kết quả thu được là Plaintext giống với giá trị ban đầu nhập vào.

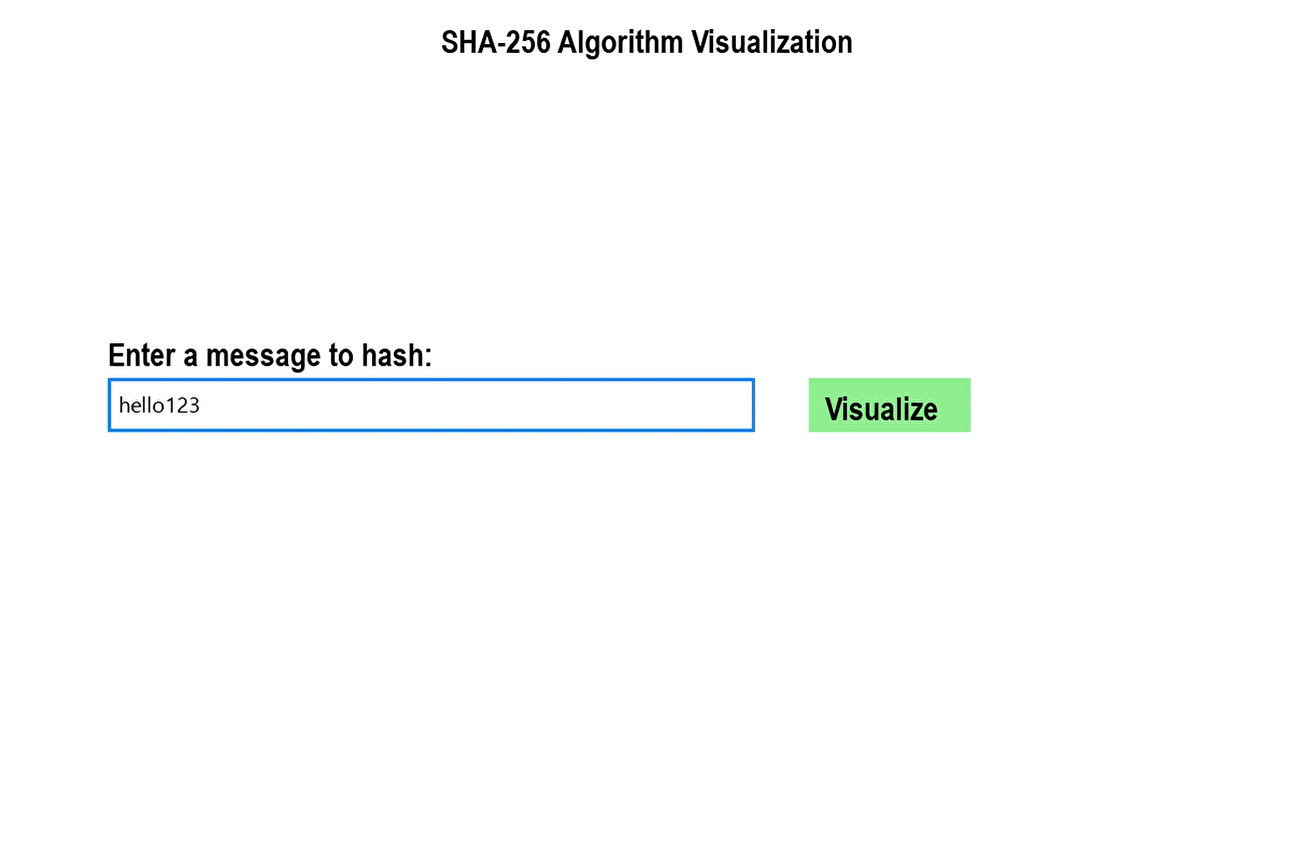


Hình 30. Kết quả sau khi ấn nút Decrypt Now

Ngoài ra có thể sử dụng nút Decrypt Now để cho ra kết quả Plaintext luôn của quá trình giải mã.

### Giải thuật SHA-256

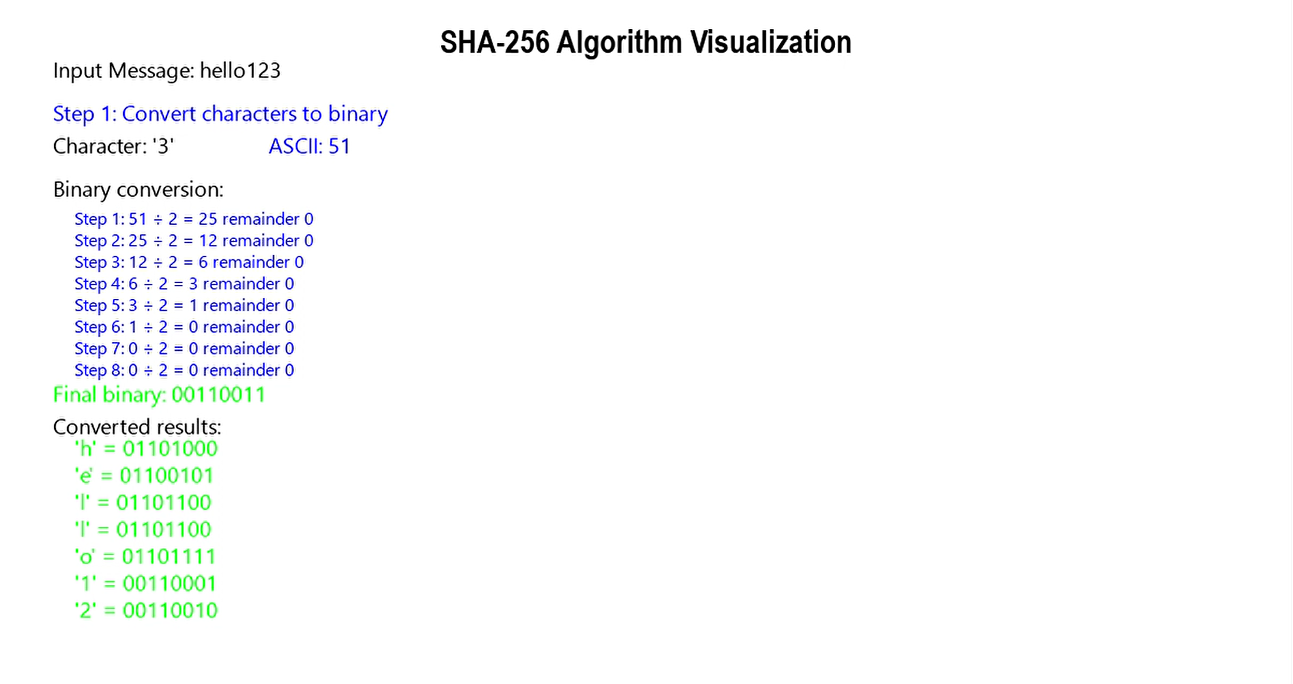
Tiến hành nhập một chuỗi bất kỳ để mô phỏng thuật toán:



Hình 32. Nhập chuỗi tin nhắn bất kỳ

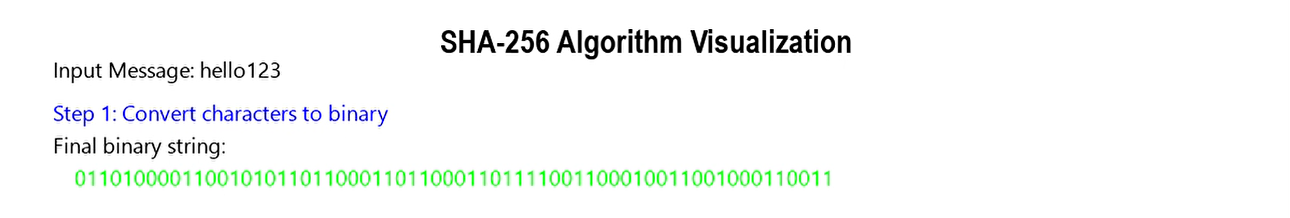
Bước 1: Chuyển đổi các ký tự sang nhị phân

##### Mỗi ký tự trong thông điệp được chuyển đổi thành mã ASCII và sau đó thành chuỗi nhị phân 8 bit.



Hình 34. Chuyển đổi các ký tự sang dạng nhị phân

Kết quả thu được sau khi hoàn thành bước 1:



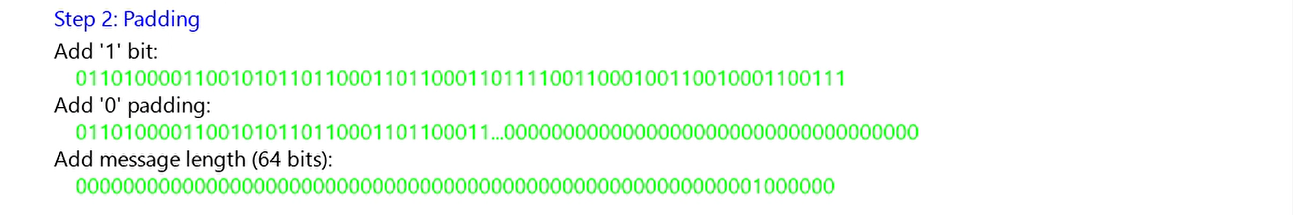
Hình 35. Kết quả thu được sau bước 1

Bước 2: Padding

##### Thêm bit '1' vào cuối thông điệp nhị phân.

##### Thêm các bit '0' để đảm bảo độ dài của thông điệp là bội số của 512, trừ 64 bit cuối.

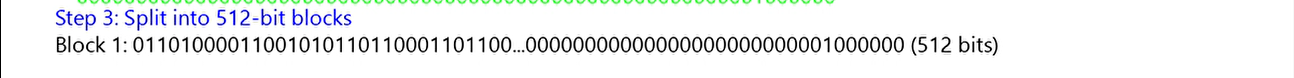
##### Thêm độ dài của thông điệp ban đầu (trước khi padding) dưới dạng nhị phân 64 bit vào cuối.



Hình 36. Kết quả thu được sau bước 2

Bước 3: Chia thông điệp thành các khối 512-bit

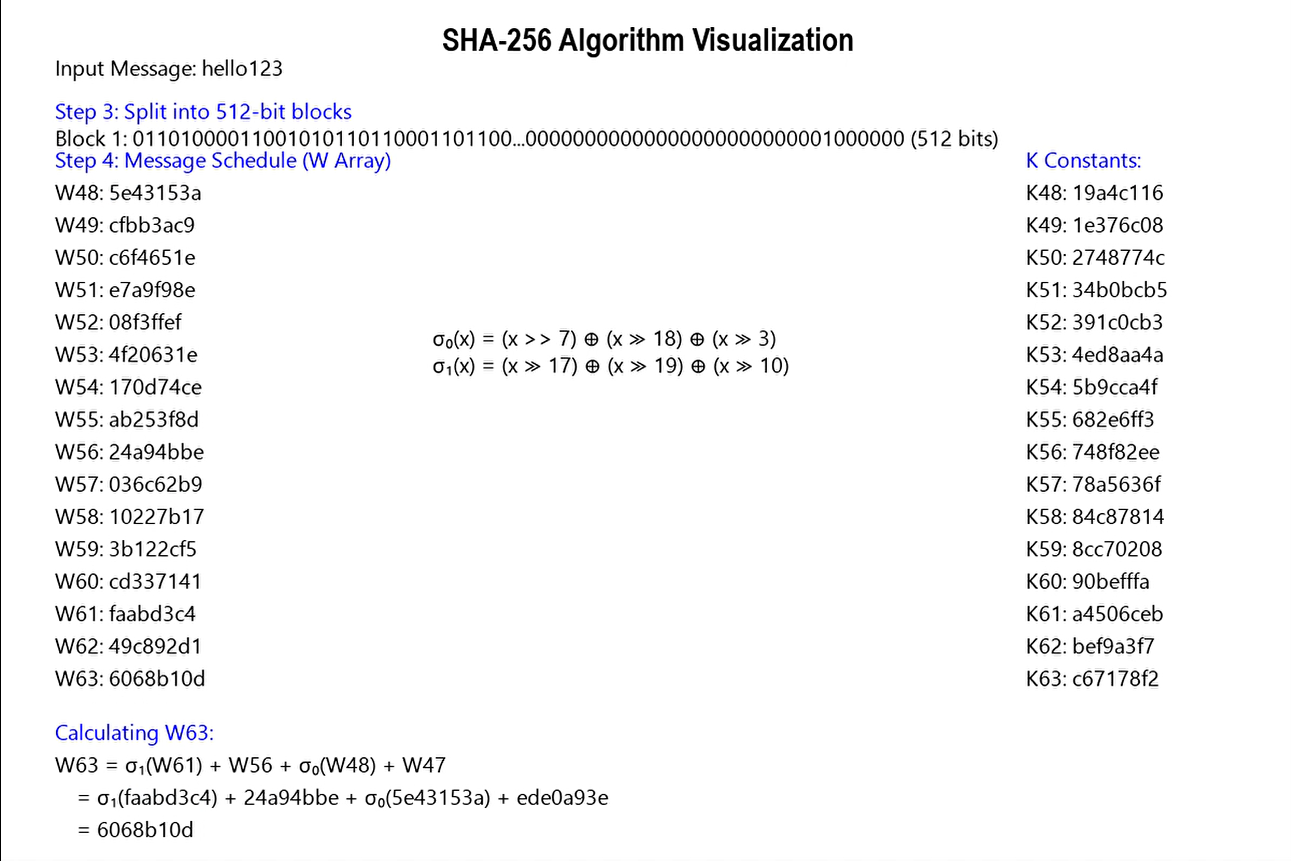
##### Thông điệp sau khi padding được chia thành các khối 512-bit.



Hình 37. Kết quả thu được sau bước 3

Bước 4: Tạo lịch trình thông điệp (W Array)

##### Mỗi khối 512-bit được chia thành 16 từ 32-bit và mở rộng thành 64 từ 32-bit bằng cách sử dụng các hàm sigma nhỏ.

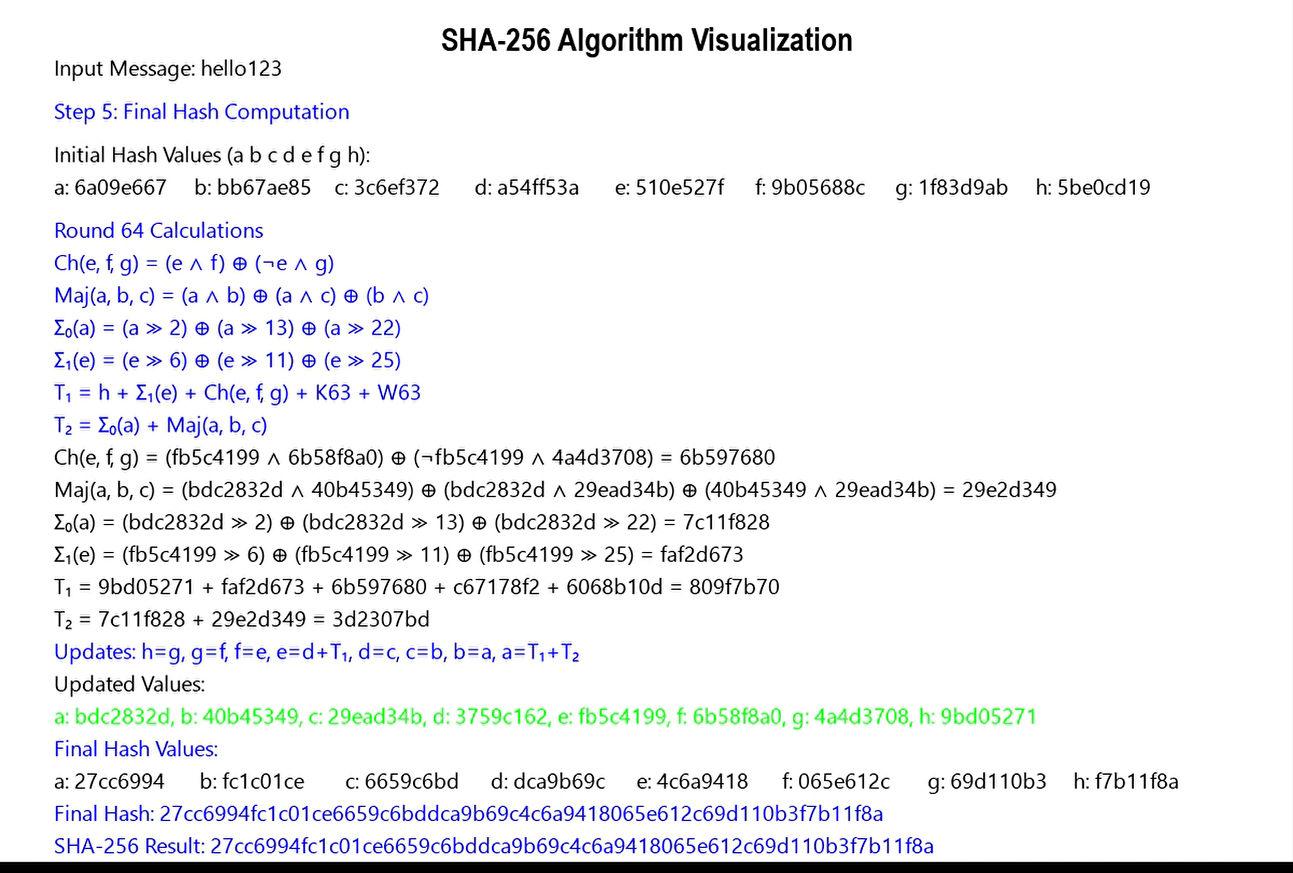


Hình 38. Kết quả thu được sau bước 4

Bước 5: Tính toán băm cuối cùng

##### Sử dụng các giá trị băm ban đầu và các hằng số K, thực hiện 64 vòng lặp nén để cập nhật các giá trị băm.

##### Sau tất cả các vòng lặp, cộng các giá trị băm ban đầu với các giá trị băm cuối cùng để tạo ra băm cuối cùng.



Hình 39. Kết quả thu được sau bước 5 và kết quả cuối cùng

## Đánh giá hiệu năng thuật toán

# KẾT LUẬN VÀ ĐỀ XUẤT