RSA

Trong mật mã học, RSA là một thuật toán mật mã hóa khóa công khai. Đây là thuật toán đầu tiên phù hợp với việc tạo ra chữ ký điện tử đồng thời với việc mã hóa. Nó đánh dấu một sự tiến bộ vượt bậc của lĩnh vực mật mã học trong việc sử dụng khóa công cộng. RSA đang được sử dụng phổ biến trong thương mại điện tử và được cho là đảm bảo an toàn với điều kiện độ dài khóa đủ lớn.

Lịch sử

Thuật toán được Ron Rivest, Adi Shamir và Len Adleman mô tả lần đầu tiên vào năm 1977 tại Học viện Công nghệ Massachusetts (MIT). Tên của thuật toán lấy từ 3 chữ cái đầu của tên 3 tác giả.

Thuật toán RSA được MIT đăng ký bằng sáng chế tại Hoa Kỳ vào năm 1983 (Số đăng ký 4.405.829). Bằng sáng chế này hết hạn vào ngày 21 tháng 9 năm 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.

**Hoạt động**

### Mô tả sơ lược

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.

Ta có thể mô phỏng trực quan một hệ mật mã khoá công khai như sau: Bob muốn gửi cho Alice một thông tin mật mà Bob muốn duy nhất Alice có thể đọc được. Để làm được điều này, Alice gửi cho Bob một chiếc hộp có khóa đã mở sẵn và giữ lại chìa khóa. Bob nhận chiếc hộp, cho vào đó một tờ giấy viết thư bình thường và khóa lại (như loại khoá thông thường chỉ cần sập chốt lại, sau khi sập chốt khóa ngay cả Bob cũng không thể mở lại được-không đọc lại hay sửa thông tin trong thư được nữa). Sau đó Bob gửi chiếc hộp lại cho Alice. Alice mở hộp với chìa khóa của mình và đọc thông tin trong thư. Trong ví dụ này, chiếc hộp với khóa mở đóng vai trò khóa công khai, chiếc chìa khóa chính là khóa bí mật.

### Tạo khóa

Giả sử Alice và Bob cần trao đổi thông tin bí mật thông qua một kênh không an toàn (ví dụ như Internet). Với thuật toán RSA, Alice đầu tiên cần tạo ra cho mình cặp khóa gồm khóa công khai và khóa bí mật theo các bước sau:

1. Chọn 2 số nguyên tố lớn p \, và q \, với p \ne q, lựa chọn ngẫu nhiên và độc lập.
2. Tính: n = p q \,.
3. Tính: giá trị hàm số Ơle \phi(n) = (p-1)(q-1) \,.
4. Chọn một số tự nhiên e sao cho 1 < e < \phi(n) \, và là số nguyên tố cùng nhau với \phi(n) \,.
5. Tính: d sao cho d e \equiv 1 \pmod{\phi(n)}.

**Khóa công khai** bao gồm:

* *n*: môđun
* *e*: số mũ công khai (cũng gọi là *số mũ mã hóa*).

**Khóa bí mật** bao gồm:

* *n*, môđun, xuất hiện cả trong khóa công khai và khóa bí mật.
* *d*, số mũ bí mật (cũng gọi là *số mũ giải mã*).

Một dạng khác của khóa bí mật bao gồm:

* *p* and *q*, hai số nguyên tố chọn ban đầu,
* *d mod (p-1)* và *d mod (q-1)* (thường được gọi là *dmp1* và *dmq1*),
* *(1/q) mod p* (thường được gọi là *iqmp*)

Dạng này cho phép thực hiện giải mã và ký nhanh hơn với việc sử dụng định lý số dư Trung Quốc ([tiếng Anh](http://vi.wikipedia.org/wiki/Ti%E1%BA%BFng_Anh): Chinese Remainder Theorem - CRT). Ở dạng này, tất cả thành phần của khóa bí mật phải được giữ bí mật.

Alice gửi khóa công khai cho Bob, và giữ bí mật khóa cá nhân của mình. Ở đây, *p* và *q* giữ vai trò rất quan trọng. Chúng là các phân tố của n và cho phép tính *d* khi biết *e*. Nếu không sử dụng dạng sau của khóa bí mật (dạng CRT) thì *p* và *q* sẽ được xóa ngay sau khi thực hiện xong quá trình tạo khóa.

### Mã hóa

Giả sử Bob muốn gửi đoạn thông tin *M* cho Alice. Đầu tiên Bob chuyển *M* thành một số *m* < *n* theo một hàm có thể đảo ngược (từ *m* có thể xác định lại *M*) được thỏa thuận trước. Quá trình này được mô tả ở phần Chuyển đổi văn bản rõ.

Lúc này Bob có *m* và biết *n* cũng như *e* do Alice gửi. Bob sẽ tính *c* là bản mã hóa của *m* theo công thức:

 c = m^e \mod{n}

Hàm trên có thể tính dễ dàng sử dụng phương pháp tính hàm mũ (theo môđun) bằng (thuật toán bình phương và nhân) Cuối cùng Bob gửi *c* cho Alice.

### Giải mã

Alice nhận *c* từ Bob và biết khóa bí mật *d*. Alice có thể tìm được *m* từ *c* theo công thức sau:

m = c^d \mod{n}

Biết *m*, Alice tìm lại *M* theo phương pháp đã thỏa thuận trước. Quá trình giải mã hoạt động vì ta có

c^d \equiv (m^e)^d \equiv m^{ed} \pmod{n}.

Do *ed* ≡ 1 (mod *p*-1) và *ed* ≡ 1 (mod *q*-1), (theo Định lý Fermat nhỏ) nên:

m^{ed} \equiv m \pmod{p}

và

m^{ed} \equiv m \pmod{q}

Do *p* và *q* là hai số nguyên tố cùng nhau, áp dụng định lý số dư Trung Quốc(CRT), ta có:

m^{ed} \equiv m \pmod{pq}.

hay:

c^d \equiv m \pmod{n}.

### Ví dụ

Sau đây là một ví dụ với những số cụ thể. Ở đây chúng ta sử dụng những số nhỏ để tiện tính toán còn trong thực tế phải dùng các số có giá trị đủ lớn.

Lấy:

|  |  |
| --- | --- |
| *p* = 61 | — số nguyên tố thứ nhất (giữ bí mật hoặc hủy sau khi tạo khóa) |
| *q* = 53 | — số nguyên tố thứ hai (giữ bí mật hoặc hủy sau khi tạo khóa) |
| *n* = *pq* = 3233 | — môđun (công bố công khai) |
| *e* = 17 | — số mũ công khai |
| *d* = 2753 | — số mũ bí mật |

Khóa công khai là cặp (*e*, *n*). Khóa bí mật là *d*. Hàm mã hóa là:

encrypt(*m*) = *me* mod *n* = *m*17 mod 3233

với *m* là văn bản rõ. Hàm giải mã là:

decrypt(*c*) = *cd* mod *n* = *c*2753 mod 3233

với *c* là văn bản mã.

Để mã hóa văn bản có giá trị 123, ta thực hiện phép tính:

encrypt(123) = 12317 mod 3233 = 855

Để giải mã văn bản có giá trị 855, ta thực hiện phép tính:

decrypt(855) = 8552753 mod 3233 = 123

Cả hai phép tính trên đều có thể được thực hiện hiệu quả nhờ thuật toán bình phương và nhân.

### Chuyển đổi văn bản rõ

Trước khi thực hiện mã hóa, ta phải thực hiện việc chuyển đổi văn bản rõ (chuyển đổi từ *M* sang *m*) sao cho không có giá trị nào của M tạo ra văn bản mã không an toàn. Nếu không có quá trình này, RSA sẽ gặp phải một số vấn đề sau:

* Nếu *m* = 0 hoặc *m* = 1 sẽ tạo ra các bản mã có giá trị là 0 và 1 tương ứng
* Khi mã hóa với số mũ nhỏ (chẳng hạn *e* = 3) và *m* cũng có giá trị nhỏ, giá trị m^e cũng nhận giá trị nhỏ (so với *n*). Như vậy phép môđun không có tác dụng và có thể dễ dàng tìm được *m* bằng cách khai căn bậc *e* của *c* (bỏ qua môđun).
* RSA là phương pháp mã hóa xác định (không có thành phần ngẫu nhiên) nên kẻ tấn công có thể thực hiện tấn công lựa chọn bản rõ bằng cách tạo ra một bảng tra giữa bản rõ và bản mã. Khi gặp một bản mã, kẻ tấn công sử dụng bảng tra để tìm ra bản rõ tương ứng.

Trên thực tế, ta thường gặp 2 vấn đề đầu khi gửi các bản tin ASCII ngắn với *m* là nhóm vài ký tự ASCII. Một đoạn tin chỉ có 1 ký tự NUL sẽ được gán giá trị *m* = 0 và cho ra bản mã là 0 bất kể giá trị của *e* và *N*. Tương tự, một ký tự ASCII khác, SOH, có giá trị 1 sẽ luôn cho ra bản mã là 1. Với các hệ thống dùng giá trị *e* nhỏ thì tất cả ký tự ASCII đều cho kết quả mã hóa không an toàn vì giá trị lớn nhất của *m* chỉ là 255 và 2553 nhỏ hơn giá trị *n* chấp nhận được. Những bản mã này sẽ dễ dàng bị phá mã.

Để tránh gặp phải những vấn đề trên, RSA trên thực tế thường bao gồm một hình thức chuyển đổi ngẫu nhiên hóa *m* trước khi mã hóa. Quá trình chuyển đổi này phải đảm bảo rằng *m* không rơi vào các giá trị không an toàn. Sau khi chuyển đổi, mỗi bản rõ khi mã hóa sẽ cho ra một trong số khả năng trong tập hợp bản mã. Điều này làm giảm tính khả thi của phương pháp tấn công lựa chọn bản rõ (một bản rõ sẽ có thể tương ứng với nhiều bản mã tuỳ thuộc vào cách chuyển đổi).

Một số tiêu chuẩn, chẳng hạn như PKCS, đã được thiết kế để chuyển đổi bản rõ trước khi mã hóa bằng RSA. Các phương pháp chuyển đổi này bổ sung thêm bít vào M. Các phương pháp chuyển đổi cần được thiết kế cẩn thận để tránh những dạng tấn công phức tạp tận dụng khả năng biết trước được cấu trúc của bản rõ. Phiên bản ban đầu của PKCS dùng một phương pháp đặc ứng (ad-hoc) mà về sau được biết là không an toàn trước tấn công lựa chọn bản rõ thích ứng (adaptive chosen ciphertext attack). Các phương pháp chuyển đổi hiện đại sử dụng các kỹ thuật như chuyển đổi mã hóa bất đối xứng tối ưu (Optimal Asymmetric Encryption Padding - OAEP) để chống lại tấn công dạng này. Tiêu chuẩn PKCS còn được bổ sung các tính năng khác để đảm bảo an toàn cho chữ ký RSA (Probabilistic Signature Scheme for RSA - RSA-PSS).

**Các vấn đề đặt ra trong thực tế**

### Quá trình tạo khóa

Việc tìm ra 2 số nguyên tố đủ lớn *p* và *q* thường được thực hiện bằng cách thử xác suất các số ngẫu nhiên có độ lớn phù hợp (dùng phép kiểm tra nguyên tố cho phép loại bỏ hầu hết các hợp số).

*p* và *q* còn cần được chọn không quá gần nhau để phòng trường hợp phân tích *n* bằng phương pháp phân tích Fermat. Ngoài ra, nếu *p*-1 hoặc *q*-1 có thừa số nguyên tố nhỏ thì cũng có thể dễ dàng bị phân tích và vì thế *p* và *q* cũng cần được thử để tránh khả năng này.

Bên cạnh đó, cần tránh sử dụng các phương pháp tìm số ngẫu nhiên mà kẻ tấn công có thể lợi dụng để biết thêm thông tin về việc lựa chọn (cần dùng các bộ tạo số ngẫu nhiên tốt). Yêu cầu ở đây là các số được lựa chọn cần đồng thời ngẫu nhiên và không dự đoán được. Đây là các yêu cầu khác nhau: một số có thể được lựa chọn ngẫu nhiên (không có kiểu mẫu trong kết quả) nhưng nếu có thể dự đoán được dù chỉ một phần thì an ninh của thuật toán cũng không được đảm bảo. Một ví dụ là bảng các số ngẫu nhiên do tập đoàn Rand xuất bản vào những năm 1950 có thể rất thực sự ngẫu nhiên nhưng kẻ tấn công cũng có bảng này. Nếu kẻ tấn công đoán được một nửa chữ số của *p* hay *q* thì chúng có thể dễ dàng tìm ra nửa còn lại (theo nghiên cứu của Donald Coppersmith vào năm 1997)

Một điểm nữa cần nhấn mạnh là khóa bí mật *d* phải đủ lớn. Năm 1990, Wiener chỉ ra rằng nếu giá trị của *p* nằm trong khoảng *q* và 2*q* (khá phổ biến) và *d* < *n*1/4/3 thì có thể tìm ra được *d* từ *n* và *e*.

Mặc dù *e* đã từng có giá trị là 3 nhưng hiện nay các số mũ nhỏ không còn được sử dụng do có thể tạo nên những lỗ hổng (đã đề cập ở phần chuyển đổi văn bản rõ). Giá trị thường dùng hiện nay là 65537 vì được xem là đủ lớn và cũng không quá lớn ảnh hưởng tới việc thực hiện hàm mũ.

### Tốc độ

RSA có tốc độ thực hiện chậm hơn đáng kể so với DES và các thuật toán mã hóa đối xứng khác. Trên thực tế, Bob sử dụng một thuật toán mã hóa đối xứng nào đó để mã hóa văn bản cần gửi và chỉ sử dụng RSA để mã hóa khóa để giải mã (thông thường khóa ngắn hơn nhiều so với văn bản).

Phương thức này cũng tạo ra những vấn đề an ninh mới. Một ví dụ là cần phải tạo ra khóa đối xứng thật sự ngẫu nhiên. Nếu không, kẻ tấn công (thường ký hiệu là Eve) sẽ bỏ qua RSA và tập trung vào việc đoán khóa đối xứng.

### Phân phối khóa

Cũng giống như các thuật toán mã hóa khác, cách thức phân phối khóa công khai là một trong những yếu tố quyết định đối với độ an toàn của RSA. Quá trình phân phối khóa cần chống lại được tấn công đứng giữa (*man-in-the-middle attack*). Giả sử Eve có thể gửi cho Bob một khóa bất kỳ và khiến Bob tin rằng đó là khóa (công khai) của Alice. Đồng thời Eve có khả năng đọc được thông tin trao đổi giữa Bob và Alice. Khi đó, Eve sẽ gửi cho Bob khóa công khai của chính mình (mà Bob nghĩ rằng đó là khóa của Alice). Sau đó, Eve đọc tất cả văn bản mã hóa do Bob gửi, giải mã với khóa bí mật của mình, giữ 1 bản copy đồng thời mã hóa bằng khóa công khai của Alice và gửi cho Alice. Về nguyên tắc, cả Bob và Alice đều không phát hiện ra sự can thiệp của người thứ ba. Các phương pháp chống lại dạng tấn công này thường dựa trên các chứng thực khóa công khai (digital certificate) hoặc các thành phần của hạ tầng khóa công khai (public key infrastructure - PKI).

### Tấn công dựa trên thời gian

Vào năm 1995, Paul Kocher mô tả một dạng tấn công mới lên RSA: nếu kẻ tấn công nắm đủ thông tin về phần cứng thực hiện mã hóa và xác định được thời gian giải mã đối với một số bản mã lựa chọn thì có thể nhanh chóng tìm ra khóa *d*. Dạng tấn công này có thể áp dụng đối với hệ thống chữ ký điện tử sử dụng RSA. Năm 2003, Dan Boneh và David Brumley chứng minh một dạng tấn công thực tế hơn: phân tích thừa số RSA dùng mạng máy tính (Máy chủ web dùng SSL). Tấn công đã khai thác thông tin rò rỉ của việc tối ưu hóa định lý số dư Trung quốc mà nhiều ứng dụng đã thực hiện.

Để chống lại tấn công dựa trên thời gian là đảm bảo quá trình giải mã luôn diễn ra trong thời gian không đổi bất kể văn bản mã. Tuy nhiên, cách này có thể làm giảm hiệu suất tính toán. Thay vào đó, hầu hết các ứng dụng RSA sử dụng một kỹ thuật gọi là che mắt. Kỹ thuật này dựa trên tính nhân của RSA: thay vì tính *cd mod n*, Alice đầu tiên chọn một số ngẫu nhiên *r* và tính *(rec)d mod n*. Kết quả của phép tính này là *rm mod n* và tác động của r sẽ được loại bỏ bằng cách nhân kết quả với nghịch đảo của r. Đỗi với mỗi văn bản mã, người ta chọn một giá trị của r. Vì vậy, thời gian giải mã sẽ không còn phụ thuộc vào giá trị của văn bản mã.