# Chương 1. Hệ thống mật mã RSA

## 1.1. Thuật toán RSA

Thuật toán RSA được đề xuất bởi Rivest, Shamir và Adleman [41]. Gọi $p$ và $q$ là hai số nguyên tố lớn ngẫu nhiên phân biệt. Mô-đun $n$ là tích của hai số nguyên tố này: $n = pq$. Hàm phi Euler *(Euler's totient function)* của $n$ cho bởi:

$$\phi(n) = (p-1)(q-1).$$

Chọn một số $1 < e < \phi(n)$ sao cho

$$gcd\big(e, \phi(n)\big) = 1,$$

và tính $d$ với

$$d = e^{-1} \, mod \, \phi(n)$$

sử dụng thuật toán Euclid mở rộng [19, 31]. Ở đây, $e$ là số mũ công khai *(public exponent)* và $d$ là số mũ bí mật *(private exponent)*. Thông thường, người ta chọn số mũ công khai nhỏ, ví dụ $e = 2^{16} + 1$. $n$ và $e$ được công khai. Giá trị của $d$ và hai số nguyên tố $p$ và $q$ được giữ bí mật. Việc mã hóa được thực hiện bằng cách tính

$$C = M^e \ (mod \, n),$$

với $M$ là bản rõ *(plaintext)* thỏa $0 \le M < n$. Số $C$ là bản mã *(ciphertext)* tương ứng của $M$. Từ $C$, $M$ được tính bằng

$$M = C^d \ (mod \, n).$$

Tính đúng đắn của thuật toán RSA được chứng minh bằng định lý Euler như sau: Cho $n$ và $a$ là hai số nguyên dương nguyên tố cùng nhau. Khi đó,

$$a^{\phi(n)} = 1 \ (mod \, n).$$

Do $ed = 1 \bmod \phi(n)$, nghĩa là $ed = 1 + K\,\phi(n)$ với $K$ nguyên, chúng ta có thể viết lại:

$$\begin{aligned} C^d &= (M^e)^d \ (\text{mod } n) \\ &= M^{ed} \ (\text{mod } n) \\ &= M^{1 + K\,\phi(n)} \ (\text{mod } n) \\ &= M.\big(M^{\,\phi(n)}\big)^K \ (\text{mod } n) \\ &= M.1 \ (\text{mod } n) \end{aligned}$$

do gcd(*M*,*n*) = 1. Ngoại lệ gcd(*M*,*n*) > 1 có thể được giải quyết như sau. Theo định lý Carmichael

$$M^{\lambda(n)} = 1 \ (mod\ n)$$

với λ(*n*) là hàm Carmichael có dạng đơn giản là *n* = *pq*, cụ thể

$$\lambda(pq) = \frac{(p-1)(q-1)}{gcd(p-1, q-1)}.$$

Chú ý rằng λ(*n*) luôn là ước thật sự (*proper divisor*) của $\phi(n)$ khi *n* là tích của các số nguyên tố lẻ phân biệt; trong trường hợp này, λ(*n*) nhỏ hơn $\phi(n)$. Giờ đây, quan hệ giữa *e* và *d* cho bởi

$$M^{ed} = M \ (mod\ n) \ nếu\ ed = 1 \left(mod\ \lambda(n)\right)$$

Do *n* là tích của các số nguyên tố phân biệt nên điều trên đúng với mọi *M,* do đó nó giúp đối phó với ngoại lệ gcd(*M*,*n*) > 1 nêu trên trong định lý Euler.

Ví dụ, chúng ta xây dựng một hệ thống mã hóa RSA như sau: Cho *p* = 11, *q* = 13, và tính

$$n = p \cdot q = 11 \cdot 13 = 143,$$

$$\phi(n) = (p-1) \cdot (q-1) = 10 \cdot 12 = 120.$$

Chúng ta cũng tính hàm Carmichael của *n*

$$\lambda(pq) = \frac{(p-1)(q-1)}{gcd(p-1, q-1)} = \frac{10 \cdot 12}{gcd(10,12)} = \frac{120}{2} = 60.$$

Số mũ công khai *e* được chọn thỏa 1 < *e* < $\phi(n)$ và

$$gcd\left(e, \phi(n)\right) = gcd(e, 120) = 1.$$

Ví dụ, *e* = 17 thỏa ràng buộc này. Số mũ bí mật *d* được tính bởi

$$d = e^{-1} \left(mod\ \phi(n)\right) = \ 17^{-1} \ (mod\ 120) = 113$$

được tính bằng cách sử dụng thuật toán Euclid mở rộng hay bất cứ thuật toán nào khác để tính phần tử nghịch đảo mô-đun. Do đó, người dùng công bố số mũ công khai *e* và mô-đun *n*: (*e*,*n*) = (13,143), và giữ bí mật các giá trị sau: *d* = 113, *p* = 11, *q* = 13. Một tiến trình mã hóa/giải mã tiêu biểu được thực hiện như sau:

| | | | |
|---|---|---|---|
| Bản rõ: | $M = 50$ | Bản mã: | $C = 85$ |
| Mã hóa: | $C := M^e \pmod n$ | Giải mã: | $M := C^d \pmod n$ |
| | $C := 50^{17} \pmod{143}$ | | $M := 85^{113} \pmod{143}$ |
| | $C = 85$ | | $M = 50.$ |

## 1.2. Trao đổi các thông điệp bí mật:

Thư mục khóa công khai chứa các cặp ($e$,$n$) cho mỗi người dùng. Những người dùng muốn gởi các thông điệp bí mật đến người dùng khác sẽ tham khảo thư mục này để nhận các tham số này. Ví dụ, thư mục này được sắp xếp như sau:

| Người dùng | Các khóa công khai |
|---|---|
| Alice | ($e_a$,$n_a$) |
| Bob | ($e_b$,$n_b$) |
| Cathy | ($e_c$,$n_c$) |
| … | … |

Cặp $n_a$ và $e_a$ tương ứng là mô-đun và số mũ công khai cho Alice. Như trong ví dụ, chúng tôi sẽ trình bày cách Alice gởi thông điệp bí mật $M$ cho Bob. Trong ví dụ giao thức đơn giản của chúng tôi, Alice thực hiện các bước sau đây:

1. Alice xác định tên của Bob trong thư mục và nhận số mũ công khai vùng với mô-đun: ($e_b$,$n_b$).
2. Alice tính $C = M^{e_b} \ (mod\ n_b)$.
3. Alice gởi $C$ cho Bob thông qua mạng.
4. Bob tiếp nhận $C$.
5. Bob sử dụng số mũ bí mật và mô-đun của anh ấy, và tính $M = C^{d_b} \ (mod\ n_b)$ để thu được $M$.

## 1.3. Ký văn bản điện tử

Thuật toán RSA cung cấp một tiến trình để ký văn bản điện tử và kiểm tra chữ ký có xác thực không. Việc ký văn bản điện tử khác với ký văn bản giấy, ở chỗ các văn bản giấy đều dùng một chữ ký. Chữ ký điện tử không thể bất biến, nó là một hàm số của văn bản điện tử khi được tạo. Sau khi chữ ký (chỉ là một phần nhỏ khác của dữ liệu điện tử) của một văn bản điện tử được tạo ra, nó được đính kèm vào văn bản để cho những ai muốn kiểm tra sự xác thực của văn bản và chữ ký. Ở đây chúng tôi sẽ minh họa ngắn gọn tiến trình ký sử dụng hệ mã RSA. Giả sử Alice muốn ký một thông điệp, và Bob muốn có bằng chứng rằng thông điệp này được ký bởi Alice. Trước tiên, Alice thực hiện các bước sau:

1. Alice tạo thông điệp $M$ và tính $S = M^{d_a} \ (mod\ n_a)$
2. Alice làm cho thông điệp $M$ và chữ ký $S$ luôn sẵn sàng để được xác thực.

Bob thực hiện các bước sau để xác thực chữ ký của Alice trên văn bản $M$:

1. Bob nhận $M$ và $S$, xác định tên của Alice trong danh bạ để lấy các thông số $e$ và $n$ $(e_a, n_a)$
2. Bob tính $M' = S^{e_a} \ (mod \ n_a)$.
3. Nếu $M'$ = $M$ thì chữ ký được xác thực. Nếu không, hoặc thông điệp gốc $M$ hoặc chữ ký $S$ bị sửa đổi, vì vậy chữ ký không hợp lệ.

Chú ý rằng các ví dụ giao thức cho ở đây chỉ để minh họa, chúng là những giao thức 'giáo khoa' (*'textbook'*); trong thực tế, các giao thức này thường phức tạp hơn. Ví dụ, kỹ thuật mã hóa khóa bí mật (*secret-key*) cũng có thể được dùng để gởi những thông điệp bí mật. Tương tự, việc ký được áp dụng cho các thông điệp có độ dài tùy ý. Chữ ký thường được tính bằng cách trước tiên tính giá trị băm của thông điệp dài và sau đó ký giá trị băm này. Đọc giả có thể tham khảo thêm báo cáo [42] và Các Chuẩn Mã Hóa Khóa Công Khai [43] được xuất bản bởi RSA Data Security, để trả lời cho các câu hỏi của vấn đề này.

## 1.4. Tính lũy thừa Modulo

Mỗi khi hệ mã RSA được cài đặt, nghĩa là mô-đun $n$, số mũ công khai $e$ và bí mật $d$ được xác định và thành phần công khai được công bố, người gửi cũng như những người nhận thực hiện một thao tác đơn để ký, xác thực, mã hóa và giải mã. Thuật toán RSA ở khía cạnh này là một trong những hệ mã đơn giản nhất. Thao tác bắt buộc là tính $M^e \pmod n$, nghĩa là lũy thừa modulo. Phép lũy thừa modulo là phép toán phổ biến để xáo trộn; nó được dùng trong nhiều hệ mã. Ví dụ, sơ đồ trao đổi khóa Diffe-Hellman đòi hỏi phép lũy thừa modulo [8]. Hơn nữa, sơ đồ chữ ký ElGamal và Chuẩn Chữ Ký Số (DDS) được đề xuất mới đây của Viện Tiêu chuẩn và Công nghệ Quốc gia [34] cũng đòi hỏi tính lũy thừa modulo. Tuy nhiên, chúng ta lưu ý rằng quá trình tính lũy thừa trong một hệ mã dựa trên bài toán logarit rời rạc có chút khác biệt: Cơ sở ($M$) và Mô-đun ($n$) được biết trước. Điều này cho phép một số tiền-tính toán vì các lũy thừa của cơ sở có thể được tính toán trước và được lưu lại [6]. Trong quá trình tính lũy thừa đối với thuật toán RSA, chúng ta biết trước số mũ $e$ và mô-đun $n$ nhưng không biết cơ sở; do đó việc tối ưu hóa vẻ không phù hợp. Trọng tâm của báo cáo này là về hệ mã RSA như tiêu đề của nó.

Trong chương tiếp theo, chúng tôi sẽ đánh giá các kỹ thuật để cài đặt phép toán lũy thừa modulo trên các máy tính thông thường, như máy tính cá nhân, các vi xử lý, các vi điều khiển, các bộ xử lý tính hiệu, các máy trạm và các máy tính lớn. Báo cáo này không bao gồm các đoạn mã thực sự; nó chủ yếu đề cập khía cạnh toán học và thuật toán của việc cài đặt phần mềm của thuật toán RSA. Ngoài ra còn có cấu trúc phần cứng để thực hiện phép nhân và lũy thừa mô-đun, ví dụ, xem [40, 28, 46,15, 24, 25, 26, 50]. Bản đánh giá tóm lược có thể được tìm thấy trong [5].