Chủ đề 6:
Mã hóa bất đối xứng

PGS.TS. Trần Minh Triết

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Mở đầu

- Vấn đề phát sinh trong các hệ thống mã hóa quy ước là việc quy ước chung mã khóa *k* giữa người gửi A và người nhận B.
- Trên thực tế, nhu cầu thay đổi nội dung của mã khóa *k* là cần thiết, do đó, cần có sự trao đổi thông tin về mã khóa *k* giữa A và B.
- Để bảo mật mã khóa *k*, A và B phải trao đổi với nhau trên một kênh liên lạc thật sự an toàn và bí mật.
- Tuy nhiên, rất khó có thể bảo đảm được sự an toàn của kênh liên lạc nên mã khóa *k* vẫn có thể bị phát hiện bởi người C!

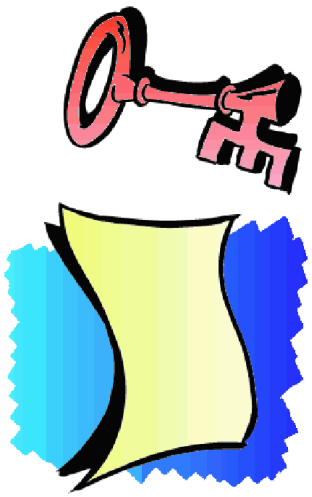
Mở đầu

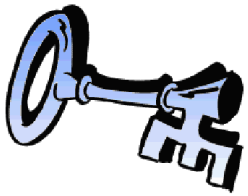
- Ý tưởng về hệ thống mã hóa khóa công cộng được Martin Hellman, Ralph Merkle và Whitfield Diffie tại Đại học Stanford giới thiệu vào năm 1976.
- Sau đó, phương pháp Diffie-Hellman của Martin Hellman và Whitfield Diffie đã được công bố.
- Năm 1977, trên báo "*The Scientific American*", nhóm tác giả Ronald Rivest, Adi Shamir và Leonard Adleman đã công bố phương pháp RSA, phương pháp mã hóa khóa công cộng nổi tiếng và được sử dụng rất nhiều hiện nay trong các ứng dụng mã hóa và bảo vệ thông tin

Mở đầu

- Một hệ thống khóa công cộng sử dụng hai loại khóa trong cùng một cặp khóa:
  - khóa công cộng (public key) được công bố rộng rãi và được sử dụng trong mã hóa thông tin,
  - khóa riêng (private key) chỉ do một người nắm giữ và được sử dụng để giải mã thông tin đã được mã hóa bằng khóa công cộng.
- Các phương pháp mã hóa này khai thác những ánh xạ f mà việc thực hiện ánh xạ ngược f^{-1} rất khó so với việc thực hiện ánh xạ f. Chỉ khi biết được mã khóa riêng thì mới có thể thực hiện được ánh xạ ngược f^{-1} .

Mã hóa khóa công cộng





Phương pháp RSA

- Năm 1978, R.L.Rivest, A.Shamir và L.Adleman đã đề xuất hệ thống mã hóa khóa công cộng RSA (hay còn được gọi là “hệ thống MIT”).
- Trong phương pháp này, tất cả các phép tính đều được thực hiện trên Z_n với n là tích của hai số nguyên tố lẻ p và q khác nhau.
- Khi đó, ta có $\phi(n) = (p–1)\ (q–1)$

Phương pháp mã hóa RSA

- $n = pq$ với p và q là hai số nguyên tố lẻ phân biệt.
- Cho $P = C = Z_n$ và định nghĩa:
- $K = \{((n, p, q, a, b): n = pq, p, q$ là số nguyên tố, $ab \equiv 1 \pmod{\phi(n)}\}$
- Với mỗi $k = (n, p, q, a, b) \in K$, định nghĩa:
- $e_k(x) = x^b \bmod n$ và $d_k(y) = y^a \bmod n$, với $x, y \in Z_n$
- Giá trị n và b được công bố (public key)
- Giá trị p, q, a được giữ bí mật (private key)

Sử dụng phương pháp RSA

- Phát sinh hai số nguyên tố có giá trị lớn p và q
- Tính $n = pq$ và $\phi(n) = (p - 1)(q - 1)$
- Chọn ngẫu nhiên một số nguyên b ($1 < b < \phi(n)$) thỏa $\gcd(b, \phi(n)) = 1$
- Tính giá trị $a = b^{-1} \bmod \phi(n)$ (bằng thuật toán Euclide mở rộng)
- Giá trị n và b được công bố (khóa công cộng)
- giá trị p, q, a được giữ bí mật (khóa riêng)

Một số phương pháp tấn công RSA

- Tính chất an toàn của phương pháp RSA dựa trên cơ sở chi phí cho việc giải mã bất hợp lệ thông tin đã được mã hóa sẽ quá lớn nên xem như không thể thực hiện được
- Vì khóa là công cộng nên việc tấn công bẻ khóa phương pháp RSA thường dựa vào khóa công cộng để xác định được khóa riêng tương ứng. Điều quan trọng là dựa vào *n* để tính *p*, *q* của *n*, từ đó tính được *a*.

Phương pháp sử dụng $\phi(n)$

- Giả sử người tấn công biết được giá trị $\phi(n)$. Khi đó việc xác định giá trị p, q được đưa về việc giải hai phương trình sau

$$n = p \cdot q$$

- Thay $q = n/p$, ta được phương trình bậc hai

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

- p, q chính là hai nghiệm của phương trình bậc hai này. Tuy nhiên vấn đề phát hiện được giá trị $\phi(n)$ còn khó hơn việc xác định hai thừa số nguyên tố của n.

Thuật toán *p*-1 phân tích ra thừa số

Nhập n và B

1. $a = 2$
2. **for** $j = 2$ **to** B **do**

   $a = a^j \bmod n$
3. $d = \gcd(a - 1, n)$
4. **if** $1 < d < n$ **then**

   d là thừa số nguyên tố của n (thành công)

   else

   không xác định được thừa số nguyên tố của n (thất bại)

Thuật toán *p*-1 phân tích ra thừa số

- Thuật toán Pollard *p*-1 (1974) là một trong những thuật toán đơn giản hiệu quả dùng để phân tích ra thừa số nguyên tố các số nguyên lớn. Tham số đầu vào của thuật toán là số nguyên (lẻ) *n* cần được phân tích ra thừa số nguyên tố và giá trị giới hạn *B*.

Thuật toán *p*-1 phân tích ra thừa số

- Ví dụ:
  - Giả sử $n = 15770708441$.
  - Áp dụng thuật toán $p - 1$ với $B = 180$, chúng ta xác định được $a = 11620221425$ ở bước 3 của thuật toán và xác định được giá trị $d = 135979$.
  - Trong trường hợp này, việc phân tích ra thừa số nguyên tố thành công do giá trị 135978 chỉ có các thừa số nguyên tố nhỏ khi phân tích ra thừa số nguyên tố:

$$135978 = 2 \times 3 \times 131 \times 173$$

    - Do đó, khi chọn $B \geq 173$ sẽ đảm bảo điều kiện $135978 \mid B$!

Thuật toán *p*-1 phân tích ra thừa số

- Trong thuật toán $p - 1$ có $B - 1$ phép tính lũy thừa modulo, mỗi phép đòi hỏi tối đa $2\log_2 B$ phép nhân modulo sử dụng thuật toán bình phương và nhân
- Việc tính USCLN sử dụng thuật toán Euclide có độ phức tạp $O((\log n)^3)$.
- Như vậy, độ phức tạp của thuật toán là

$$O(B \log B(\log n)^2 + (\log n)^3)$$

Thuật toán *p*-1 phân tích ra thừa số

- Xác suất chọn giá trị *B* tương đối nhỏ và thỏa điều kiện là rất thấp.
- Khi tăng giá trị *B* (chẳng hạn như $B \approx \sqrt{n}$) thì giải thuật sẽ thành công, nhưng thuật toán này sẽ không nhanh hơn giải thuật chia dần như trình bày trên.

Thuật toán *p*-1 phân tích ra thừa số

- Giải thuật này chỉ hiệu quả khi tấn công phương pháp RSA trong trường hợp *n* có thừa số nguyên tố *p* mà (*p* – 1) chỉ có các ước số nguyên tố rất nhỏ
- Chúng ta có thể dễ dàng xây dựng một hệ thống mã hóa khóa công cộng RSA an toàn đối với giải thuật tấn công *p* – 1. Cách đơn giản nhất là tìm một số nguyên tố *p*1 lớn, mà $p = 2p_1 + 1$ cũng là số nguyên tố, tương tự tìm *q*1 nguyên tố lớn và $q = 2q_1 + 1$ nguyên tố.

Bẻ khóa dựa trên các tấn công lặp lại

- Simons và Norris: hệ thống RSA có thể bị tổn thương khi sử dụng tấn công lặp liên tiếp. Nếu đối thủ biết cặp khóa công cộng {*n*, *b*} và từ khóa *C* thì có thể tính chuỗi các từ khóa sau:

  $C_1 = C^b \pmod{n}$

  $C_2 = C_1^b \pmod{n}$

  ...

  $C_i = C_{i-1}^b \pmod{n}$

- Nếu có một phần tử C_j trong chuỗi $C_1, C_2, C_3, \ldots, C_i$ sao cho $C_j = C$ thì khi đó sẽ tìm được $M = C_{j-1}$ vì

  $C_j = C_{j-1}^b \pmod{n}$

  $C = M^b \pmod{n}$

Bẻ khóa dựa trên các tấn công lặp lại

- Ví dụ: Giả sử anh ta biết {n, b, C}={35, 17, 3},anh ta sẽ tính:

  $C_1 = C^b \pmod{n} = 3^{17} \pmod{35} = 33$

  $C_2 = C_1^b \pmod{n} = 33^{17} \pmod{35} = 3$

- Vì $C_2 = C$ nên $M = C_1 = 33$

Sự che dấu thông tin trong hệ thống RSA

- Hệ thống RSA có đặc điểm là thông tin không phải luôn được che dấu.
- Giả sử người gởi có $b = 17$, $n = 35$. Nếu anh ta muốn gởi bất cứ dữ liệu nào thuộc tập sau

  {1, 6, 7, 8, 13, 14, 15, 20, 21, 22, 27, 28, 29, 34}

  thì kết quả của việc mã hóa lại chính là dữ liệu ban đầu. Nghĩa là, $M = M^b \bmod n$.
- Còn khi $p = 109$, $q = 97$, $b = 865$ thì hệ thống hoàn toàn không có sự che dấu thông tin, bởi vì:

$$\forall M,\ M = M^{865} \bmod (109*97)$$

Sự che dấu thông tin trong hệ thống RSA

- Với mỗi giá trị *n*, có ít nhất 9 trường hợp kết quả mã hóa chính là dữ liệu nguồn ban đầu. Thật vậy,

$$M = M^b \bmod n$$

hay:

$$M = M^b \bmod p \text{ và } M = M^b \bmod q \qquad (*)$$

- Với mỗi *e*, mỗi đẳng thức trong (*) có ít nhất ba giải pháp thuộc tập {0, 1, -1}.
- Số thông điệp không được che dấu (không bị thay đổi sau khi mã hóa):

$$m = [1+\gcd(b-1, p-1)][1+\gcd(b-1), q-1]$$

Nhận xét

- Mấu chốt để có thể giải mã được thông tin là có được giá trị p và q tạo nên giá trị n.
- Khi có được hai giá trị này, ta có thể dễ dàng tính ra được $\phi(n) = (p - 1)(q - 1)$ và giá trị $a = b^{-1} \bmod \phi(n)$ theo thuật toán Euclide mở rộng.
- Nếu số nguyên n có thể được phân tích ra thừa số nguyên tố, tức là giá trị p và q có thể được xác định thì xem như tính an toàn của phương pháp RSA không còn được bảo đảm nữa.

Nhận xét

- Như vậy, tính an toàn của phương pháp RSA dựa trên cơ sở các máy tính tại thời điểm hiện tại chưa đủ khả năng giải quyết việc phân tích các số nguyên rất lớn ra thừa số nguyên tố.
- Năm 1994, Peter Shor, một nhà khoa học tại phòng thí nghiệm AT&T, đã đưa ra một thuật toán có thể phân tích một cách hiệu quả các số nguyên rất lớn trên máy tính lượng tử.

Thuật toán Euclide mở rộng

Xác định $b = a^{-1} \bmod n$ trên Z_n

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Thuật toán Euclide mở rộng

- **Bước 1**: Xây dựng bảng (gồm 6 cột) như sau:

Dòng	r_0	r_1	r_2	q	t_0	t_1

Trên mỗi dòng, ta có: $r_0 = r_1 \times q + r_2$

- **Bước 2**: Điền giá trị vào dòng đầu tiên $r_0 = n$, $r_1 = a$, $t_0 = 0$, $t_1 = 1$

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	n	a			0	1

Thuật toán Euclide mở rộng

- **Bước 3**: Trên dòng *i* đang xét, tính giá trị

  $r_2 = r_0 \bmod r1$,

  $q = \lfloor r_0 / r_1 \rfloor$

Dòng	r_0	r_1	r_2	q	t_0	t_1
…	…	…	…	…	…	…
i			$r_0 \bmod r_1$	$\lfloor r_0 / r_1 \rfloor$		

Thuật toán Euclide mở rộng

- **Bước 4**: Tính giá trị t_1 (của dòng i) từ giá trị q, t_0 và t_1 của dòng i-1.

Dòng	r_0	r_1	r_2	q	t_0	t_1
…	...	...	...	...	…	…
i -1				X	Y	Z
i						$Y-X\times Z$ mod n

Thuật toán Euclide mở rộng

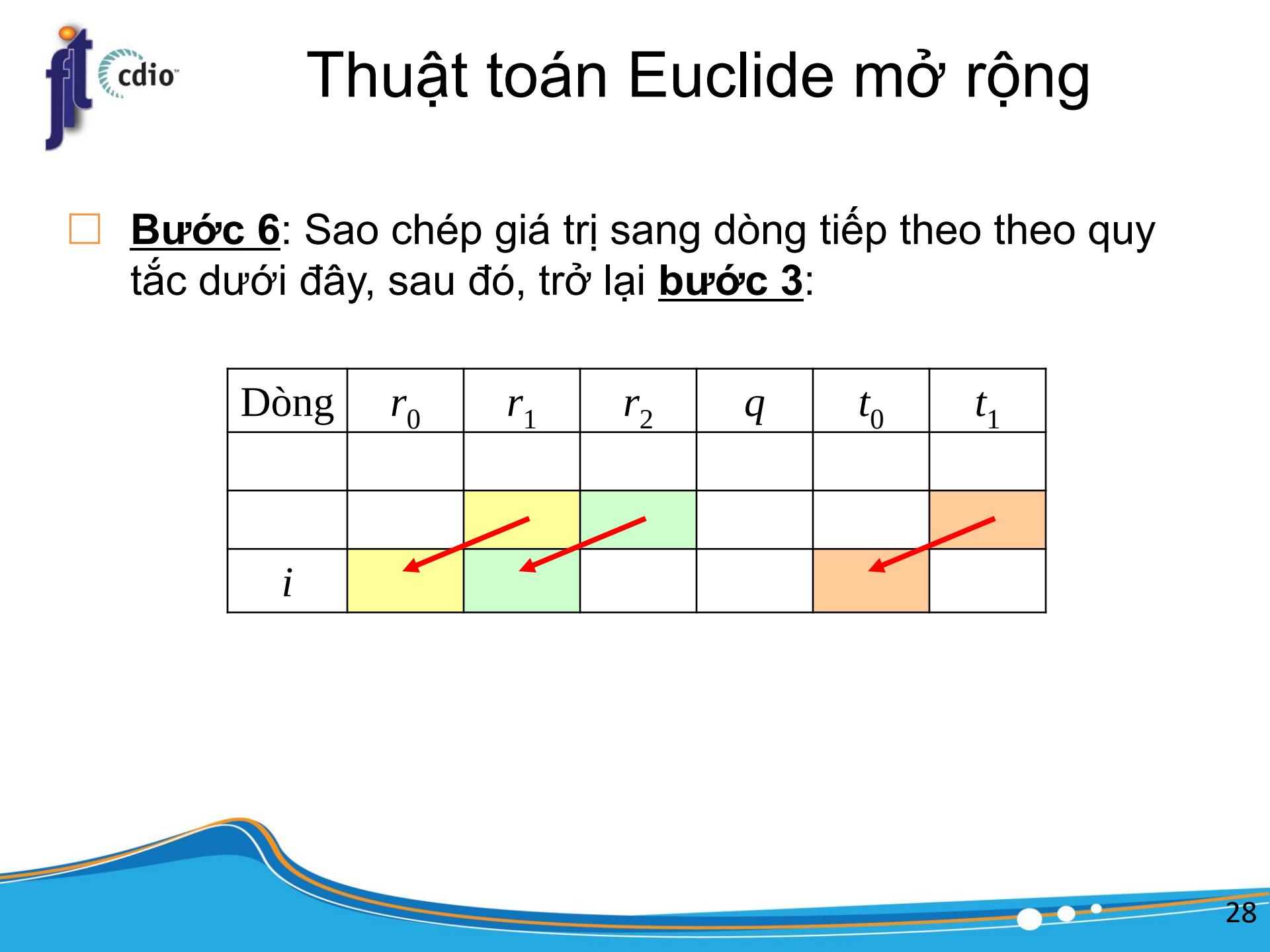
- **Bước 5**: Trên dòng *i* đang xét:
  - **Nếu** $r_2 = 0$ **thì**:
    - **Nếu** $r_1 = 1$ **thì** giá trị t_1 (của dòng đang xét) là phần tử nghịch đảo của *a* trong Z_n
    - **Ngược lại** (tức là $r_1 \neq 1$) **thì** không tồn tại phần tử nghịch đảo của *a* trong Z_n. Rõ ràng trường hợp này chỉ xảy ra khi USCLN(*a*, *n*) $\neq 1$
    - Chấm dứt thuật toán
  - **Ngược lại** (tức là $r_2 \neq 0$) thì sang bước 6

Dòng	r_0	r_1	r_2	q	t_0	t_1
…	…	…	…	…	…	…
i		$r_1 = 1$?	$r_2 = 0$?			

Thuật toán Euclide mở rộng

- **Bước 6**: Sao chép giá trị sang dòng tiếp theo theo quy tắc dưới đây, sau đó, trở lại **bước 3**:

Dòng	r_0	r_1	r_2	q	t_0	t_1
i						



Thuật toán Euclide mở rộng

- Ví dụ 1: n=101, a = 25

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	101	25	1	4	0	1

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	101	25	1	4	0	1
1	25	***1***	***0***	25	1	***97***

Vậy $25^{-1} = 97$ (trong Z_{101})

Thuật toán Euclide mở rộng

❑Ví dụ 2: n = 1024, a = 173

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1
1	173	159	14	1	1	1019

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1
1	173	159	14	1	1	1019
2	159	14	5	11	1019	6

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1
1	173	159	14	1	1	1019
2	159	14	5	11	1019	6
3	14	5	4	2	6	953

Thuật toán Euclide mở rộng

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1
1	173	159	14	1	1	1019
2	159	14	5	11	1019	6
3	14	5	4	2	6	953
4	5	4	1	1	953	148

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	1024	173	159	5	0	1
1	173	159	14	1	1	1019
2	159	14	5	11	1019	6
3	14	5	4	2	6	953
4	5	4	1	1	953	148
5	4	***1***	***0***	4	148	***805***

Vậy $173^{-1} = 805$ (trong Z_{1024})

Áp dụng:

Xét phương pháp mã hóa bằng phép nhân (Multiplicative Cipher)

Cho $P = C = (\mathbf{Z}_n)^m$, $K = \{k \in \mathbf{Z}_n : \gcd(k, n) = 1\}$

Với mỗi khóa $k \in \mathbf{Z}_n$, định nghĩa:

$e_k(x) = kx \bmod n$ và $d_k(y) = k^{-1} y \bmod n$ với $x, y \in \mathbf{Z}_n$

➔ Cho trước giá trị k và n, ta cần xác định k^{-1} (trong Z_n) để giải mã thông tin

Áp dụng:

Xét phương pháp mã hóa Affine (Affine Cipher) :

Cho $P = C = Z_n$

$K = \{(a,b) \in \mathbf{Z}_n \times \mathbf{Z}_n : \gcd(a,n) = 1\}$

Với mỗi khóa $k = (a,b) \in K$, định nghĩa:

$e_k(x) = (ax + b) \bmod n$ và $d_k(x) = (a^{-1}(y - b)) \bmod n$ với $x, y \in \mathbf{Z}_n$

$E = \{e_k, k \in K\}$ và $D = \{D_k, k \in K\}$

➔ Cho trước giá trị a, b và n, ta cần xác định a^{-1} (trong Z_n) để giải mã thông tin

Áp dụng:

Xác định số mũ trong phương pháp RSA

Phát sinh hai số nguyên tố có giá trị lớn p và q

Tính $n = pq$ và $\phi(n) = (p - 1)(q - 1)$

Chọn ngẫu nhiên một số nguyên b ($1 < b < \phi(n)$) thỏa $\gcd(b, \phi(n)) = 1$

Tính giá trị $a = b^{-1} \bmod \phi(n)$ (bằng thuật toán Euclide mở rộng)

Giá trị n và b được công bố (khóa công cộng), trong khi giá trị p, q, a được giữ bí mật (khóa riêng)

➔ Cho trước giá trị b, p và q, ta cần xác định $a = b^{-1} \bmod \phi(pq)$.

Lưu ý, ta có: $\phi(n) = (p - 1)(q - 1)$

Áp dụng:

Ví dụ : Chọn p=13, q=17. Ta có $n = pq = 221$, $\phi(n) = (p-1)(q-1) = 192$

Chọn $b = 25$.

Tính $a = b^{-1} \bmod \phi(n) = 25^{-1} \bmod 192 = 169$

Dòng	r_0	r_1	r_2	q	t_0	t_1
0	192	25	17	7	0	1
1	25	17	8	1	1	185
2	17	8	1	2	185	8
3	8	***1***	***0***	8	8	***169***

Vấn đề số nguyên tố

KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

Vấn đề số nguyên tố

- Để bảo đảm an toàn cho hệ thống mã hóa RSA, số nguyên $n = pq$ phải đủ lớn để không thể dễ dàng tiến hành việc phân tích n ra thừa số nguyên tố.
- Hiện tại, các thuật toán phân tích thừa số nguyên tố đã có thể giải quyết được các số nguyên có trên 130 chữ số (thập phân).
- Để an toàn, số nguyên tố p và q cần phải đủ lớn, ví dụ như trên 100 chữ số.
- Vấn đề đặt ra ở đây là giải quyết bài toán: làm thế nào để kiểm tra một cách nhanh chóng và chính xác một số nguyên dương n là số nguyên tố hay hợp số?

Vấn đề số nguyên tố

- Theo định nghĩa, một số nguyên dương *n* là số nguyên tố khi và chỉ khi *n* chỉ chia hết cho 1 và *n* (ở đây chỉ xét các số nguyên dương).
- Từ đó suy ra, *n* là số nguyên tố khi và chỉ khi *n* không có ước số dương nào thuộc đoạn
- . Như vậy, ta có:

  n là số nguyên tố

Vấn đề số nguyên tố

- Việc kiểm tra một số nguyên dương *n* là số nguyên tố theo phương pháp trên sẽ đưa ra kết quả hoàn toàn chính xác.
- Tuy nhiên, thời gian xử lý của thuật toán rõ ràng là rất lớn, hoặc thậm chí không thể thực hiện được, trong trường hợp *n* tương đối lớn.

Thuật toán Miller-Rabin

- Trên thực tế, việc kiểm tra một số nguyên dương *n* là số nguyên tố thường áp dụng các phương pháp thuộc nhóm thuật toán Monte Carlo,
- ví dụ:
  - thuật toán Solovay-Strassen hay thuật toán Miller-Robin;
  - thuật toán Miller-Robin thường được sử dụng phổ biến hơn.

Thuật toán thuộc nhóm Monte Carlo

- Thuật toán thuộc nhóm Monte Carlo được sử dụng trong việc khẳng định hay phủ định một vấn đề nào đó. Thuật toán luôn đưa ra câu trả lời và câu trả lời thu được chỉ có khả năng hoặc là “Có” (yes) hoặc là “Không” (no)

- Thuật toán “yes-biased Monte Carlo” là thuật toán Monte Carlo, trong đó, câu trả lời “Có” (Yes) luôn chính xác nhưng câu trả lời “Không” (No) có thể không chính xác

Thuật toán Miller-Rabin

- Ưu điểm: Xử lý nhanh (số nguyên dương *n* có thể được kiểm tra trong thời gian tỉ lệ với $\log_2 n$, tức là số lượng các bit trong biểu diễn nhị phân của *n*)
- Có khả năng kết luận của thuật toán không hoàn toàn chính xác, nghĩa là có khả năng một hợp số *n* lại được kết luận là số nguyên tố, mặc dù xác suất xảy ra kết luận không chính xác là không cao.
- Có thể khắc phục bằng cách thực hiện thuật toán nhiều lần để giảm khả năng xảy ra kết luận sai xuống dưới ngưỡng cho phép ➔ kết luận có độ tin cậy cao

Thuật toán Miller-Rabin

Phân tích số nguyên dương $n = 2^k m + 1$ với m lẻ
Chọn ngẫu nhiên số nguyên dương $a \in \{1, 2, ..., n - 1\}$
Tính $b = a^m \bmod p$
if $b \equiv 1 \pmod p$ then
    Kết luận "p là số nguyên tố" và dừng thuật toán
end if
for $i = 0$ to $k - 1$
    if $b \equiv p - 1 \pmod p$ then
        Kết luận "p là số nguyên tố" và dừng thuật toán
    else
        $b = b^2 \bmod p$
    end if
end for
Kết luận "p là hợp số"

Thuật toán Miller-Rabin

- Thuật toán Miller-Rabin là thuật toán "yes-biased Monte Carlo" đối với phát biểu "số nguyên dương *n* là hợp số".
- Xác suất xảy ra kết luận sai, nghĩa là thuật toán đưa ra kết luận "*n* là số nguyên tố" khi *n* thật sự là hợp số, chỉ tối đa là 25%.
- Nếu áp dụng thuật toán *k* lần với các giá trị *a* khác nhau mà ta vẫn thu được kết luận "*n* là số nguyên tố" thì xác suất chính xác của kết luận này là $1-4^{-k} \rightarrow 1$, với *k* đủ lớn.

Xử lý số học

- Tính giá trị của biểu thức $z = x^b \bmod n$
- Thuật toán "bình phương và nhân"

  Biểu diễn b dạng nhị phân $b_{l-1}b_{l-2}...b_1b_0$, $b_i \in \{0, 1\}$, $0 \leq i < l$

  $z = 1$

  $x = x \bmod n$

  for $i = l-1$ **downto** 0

  $z = z^2 \bmod n$

  if $b_i = 1$ **then**

  $z = z \times x \bmod n$

  end if

  end for

Phép nhân c = a * b

```
t = 0
c = 0
while ( b > 1 )
{
    if ( IsOdd (b) )
        t = t + c
    c = c << 1 // c = c + c
    b = b >> 1
}
c = c + t
```

Phép lũy thừa $c = a^b$

```
t = 1
c = a
while ( b > 1 )
{
    if ( IsOdd (b) )
        t = t * c
    c = c * c
    b = b >> 1
}
c = c * t
```

Phép nhân modulo c = (a*b) % n (n≠0)

```
if ( b == 0 )
    c = 0
t = 0
c = a % n
b = b % n
while ( b > 1 )
{
    if ( IsOdd (b) )
        t = (t + c) % n;
    c = ( c << 1) % n
    b = b >> 1
}
c = (c + t) % n;
```

Phép lũy thừa modulo $c = a^b \% n$ $(n \neq 0)$

```
t = 1
c = a % n
while ( b > 1 )
{
    if ( IsOdd (b) )
        t = (t * c) % n
    c = (c * c) % n
    b = b >> 1
}
c = (t * c) % n
```

Mã hóa đối xứng VS mã hóa bất đối xứng

- Các phương pháp mã hóa quy ước có ưu điểm xử lý rất nhanh so với các phương pháp mã hóa khóa công cộng.
- Do khóa dùng để mã hóa cũng được dùng để giải mã nên cần phải giữ bí mật nội dung của khóa và mã khóa được gọi là khóa bí mật (secret key). Ngay cả trong trường hợp khóa được trao đổi trực tiếp thì mã khóa này vẫn có khả năng bị phát hiện. Vấn đề khó khăn đặt ra đối với các phương pháp mã hóa này chính là bài toán trao đổi mã khóa.

Mã hóa đối xứng VS mã hóa bất đối xứng

- Khóa công cộng dễ bị tấn công hơn khóa bí mật.
- Để tìm ra được khóa bí mật, người giải mã cần phải có thêm một số thông tin liên quan đến các đặc tính của văn bản nguồn trước khi mã hóa để tìm ra manh mối giải mã thay vì phải sử dụng phương pháp vét cạn mã khóa.
- Ngoài ra, việc xác định xem thông điệp sau khi giải mã có đúng là thông điệp ban đầu trước khi mã hóa hay không lại là một vấn đề khó khăn.
- Đối với các khóa công cộng, việc công phá hoàn toàn có thể thực hiện được với điều kiện có đủ tài nguyên và thời gian xử lý.