Mật mã bất đối xứng
Public key cryptography

Điểm yếu của mã đối xứng

- Việc phân phối và quản lý khoá khó khăn, nhất là với các hệ thống có nhiều người sử dụng
  - N người dùng ➔ $n(n-1)/2$ mối quan hệ ➔ mỗi người cần quản lý $(n-1)$ khoá
- Không thể sử dụng vào chữ ký điện tử
  - Không thể đảm bảo được tính "không chối từ" (sẽ học ở các buổi sau)

Ý tưởng của Diffie-Hellman về hệ mã bất đối xứng (mã công khai)

- Về nguyên tắc, mã công khai được thiết kế trên quan điểm "hướng tới 1 người dùng", chứ không phải hướng tới 1 cặp người dùng (như mã bí mật)
  - Được sử dụng với nhiều mục đích khác ngoài việc mã hoá
- Đề xuất bởi Diffie và Hellman (1976) trong bài báo "New Directions in Cryptography"
  - Cơ chế mã hoá
  - Cơ chế phân phối khoá
    - Thuật toán phân phố khoá Diffie-Hellman
  - Chữ ký điện tử

Đề xuất của Diffie-Hellman

- Mỗi người dùng tạo 2 khoá: 1 khoá giữ bí mật (secret (private) key) và 1 khoá công khai cho tất cả mọi người khác (public key)
  - Khóa bí công khai được dùng để mã hoá, khoá bí mật được dùng để giải mã

$$X = D(z, E(Z, X))$$

  - Khóa bí mật được dùng để tạo chữ ký điện tử, khoá công khai được dùng để xác thực chữ ký điện tử

$$X = E(Z, D(z, X))$$

- Mã công khai còn được gọi là mã bất đối xứng (asymmetric key cryptosystems)
  - Kể cả biết được bản mã (cipher text) và khoá công khai (public-key) thì cũng không thể tính ngược lại được bản rõ và khoá bí mật

Nguyên tắc cấu tạo một hệ PK (trapdoor)

- Một hệ mã PKC có thể được tạo dựng trên cơ sở sử dụng một hàm kiểu one - way (1 chiều). Một hàm f được gọi là one-way nếu:
  1. Đối với mọi X tính ra $Y = f(X)$ là dễ dàng.
  2. Khi biết Y rất khó để tính ra X.
- Ví dụ. Cho n số nguyên tố $p_1, p_2, \dots p_n$ ta có thể dễ dàng tính được $n = p_1 \times p_2 \times \cdots \times p_n$, tuy nhiên khi biết N, việc tìm các thừa số nguyên tố của nó là khó khăn hơn rất nhiều

- Cần một hàm one-way đặc biệt, trang bị một trap-door (cửa bẫy), sao cho nếu biết trap-door này thì việc tính X khi biết $f(X)$ (tức là đi tìm nghịch đảo của f) là dễ, còn ngược lại thì khó

- Một hàm one-way có trap door như thế ➔ một hệ mã PKC
  - Lấy E (hàm sinh mã) là hàm one- way có trap-door
  - Trap- door chính là khoá mật, mà nếu biết nó thì có thể dễ dàng tính được cái nghịch đảo của E tức là biết D, còn nếu không biết thì rất khó tính được.

Trapdoor Knapsack dựa trên bài toán đóng thùng

- 1978, hai ông Merkle - Hellman đã đề xuất một thuật toán mã hoá PKC dựa trên bài toán ĐÓNG THÙNG như sau:
  - Cho 1 tập hợp các số dương a_i, $1 \le i \le n$ và 1 số T dương. Hãy tìm 1 tập hợp chỉ số $S \subseteq \{1, 2, \dots, n\}$ sao cho: $\sum_{i \in S} ai = T$

- Bài toán này là một bài toán khó, theo nghĩa là chưa tìm được thuật toán nào tốt hơn là thuật toán thử-vét cạn
  - Thời gian xử lý vét cạn có thể tỉ lệ luỹ thừa theo kích thước input n.

  - VD: $(a_1, a_2, a_3, a_4) = (2, 3, 5, 7) \quad T = 7.$
    Như vậy ta có 2 đáp số $S = (1, 3)$ và $S = (4)$

- $a_1 < a_2 < \cdots < a_n$
- $a_i > a_{i-1} + \cdots + a_1$
- $T = \sum a_{k_1} + a_{k_2} + \cdots + a_{k_m}$
- $a_{q_1} \mid a_{q_1} < T < a_{q_1+1}$
- $T = a_{q_1} + (T - a_{q_1})$
- $T = a_q + \cdots ..$
- $a_{q_2} < T - a_{q_1} < a_{q_2+1}$
- $a_1 < a_2 < \cdots < a_{q_2} < a_{q_2+1} < \cdots < a_{q_1} < \cdots$
- $T = a_{q_1} + a_{q_2} + \cdots$

Hệ PKC Merkle - Hellman

- Từ bài toán đóng thùng này chúng ta sẽ khảo sát các khả năng vận dụng để tạo ra thuật toán mã khối PKC. Sơ đồ đầu tiên như sau:
  - Chọn một vector $a = (a_1, a_2, \dots, an)$ - được gọi là vector mang (cargo vector)
  - Với một khối tin $X = (X_1, X_2, X_3 \dots, X_n)$, ta thực hiện phép mã hoá như sau: $T = \sum a_i X_i$ (*)
  - Việc giải mã là: Cho mã T, vector mang a, tìm các X_i sao cho thoả mãn (*).
- Sơ đồ này thể hiện một hàm one-way với việc sinh mã rất dễ dàng nhưng việc giải mã là rất khó ➔ cơ sở xây dựng một trapdoor

Hệ PKC Merkle - Hellman

- Merkle sử dụng một mẹo là áp dụng một vector mang đặc biệt là vector siêu tăng (super-increasing)
  - thành phần $i+1$ là lớn hơn tổng giá trị của các thành phần đứng trước nó $(1 \rightarrow i)$.
- Việc giải mã có thể diễn ra dễ dàng như ví dụ bằng số sau:

  Vector mang siêu tăng: $a = (1,2,4,8)$

  Cho $T = 11$, ta sẽ thấy việc tìm $X = (X_1, X_2, X_3, X_4)$ sao cho $T = \sum a_i X_i$ là dễ dàng:

  Đặt $T = T_0$

  $X_4 = 1$; $T_0 = T_0 - X_4 = 3$ ➔ $(X_1\ X_2\ X_3\ 1)$

  $X_3 = 0$; $T_2 = T_1 = 3$➔ $(X_1\ X_2\ 0\ 1)$

  $X_2 = 1$; $T_3 = T_2 - 2 = 1$➔ $(X_1\ 1\ 0\ 1)$

  $X_1 = 1$ ➔ $(1\ 1\ 0\ 1)$

Hệ PKC Merkle - Hellman

- Bài toán được giải quyết dần qua các bước.
  - Ở bước i, tổng đích là T_i (tức là phải tìm các a_j để tổng bằng T_i). Ta đem so sánh T_i với thành phần lớn nhất trong phần còn lại của vector, nếu lớn hơn thì thành phần này được chọn tức là X_i tương ứng bằng 1, còn ngược lại thì X_i tương ứng bằng 0. Sau đó tiếp tục chuyển sang bước sau với $T_{i+1} = T_i - Xi$.
- Cần chủ động "nguỵ trang" vector siêu tăng để chỉ có người chủ mới biết còn người ngoài không thể giải mã được.

Hệ PKC Merkle – Hellman: Cơ chế nguỵ trang

- *Tạo khoá*:

  Alice chọn một vector siêu tăng:

  $$a' = (a_1', a_2', \ldots, an')$$

  a' được giữ bí mật tức là một thành phần của khoá bí mật

  - Sau đó chọn một số nguyên $m > \sum ai'$, gọi là mo-dul đồng dư và một số nguyên ngẫu nhiên ω, gọi là nhân tử, sao cho nguyên tố cùng nhau với *m*.

  - Khoá công khai của Alice sẽ là vector *a* là tích của *a'* với nhân tử *ω*:

    $a = (a_1, a_2, \ldots, a_n)$

    $a_i = \omega \times a_i' \ (mod\ m);\ i = 1,2,3 \ldots n$

  - Còn khoá bí mật sẽ là bộ ba (a', m, ω)

Sơ đồ cụ thể Merkle-Hellman dựa trên bài toán đóng thùng.

- *Sinh mã:*
  - Khi Bob muốn gửi một thông báo X cho Alice, anh ta tính mã theo công thức:

$$T = a_i X_i$$

- *Giải mã*:
  - Alice nhận được T, giải mã như sau:

Để bỏ lớp nguỵ trang cô ta trước hết tính ω^{-1} (là giá trị nghịch đảo của ω, tức là $\omega \times \omega^{-1} = 1 \; mod \; m$, sẽ giới thiệu thuật toán tính sau), rồi tính

$$T' = T \times \omega^{-1} (mod \; m)$$

  - Alice biết rằng $T' = a'.X$ nên cô ta có thể dễ dàng giải ra được X theo siêu tăng a'.

- Chú thích: ở đây ta có

$$T' = T \times \omega^{-1} = \sum a_i X_i \, \omega^{-1} = \sum a_i \, ' \, \omega X_i \, \omega^{-1}$$
$$= \sum (a_i ' \omega \, \omega^{-1}) X_i = \sum a_i ' X_i = a '.X$$

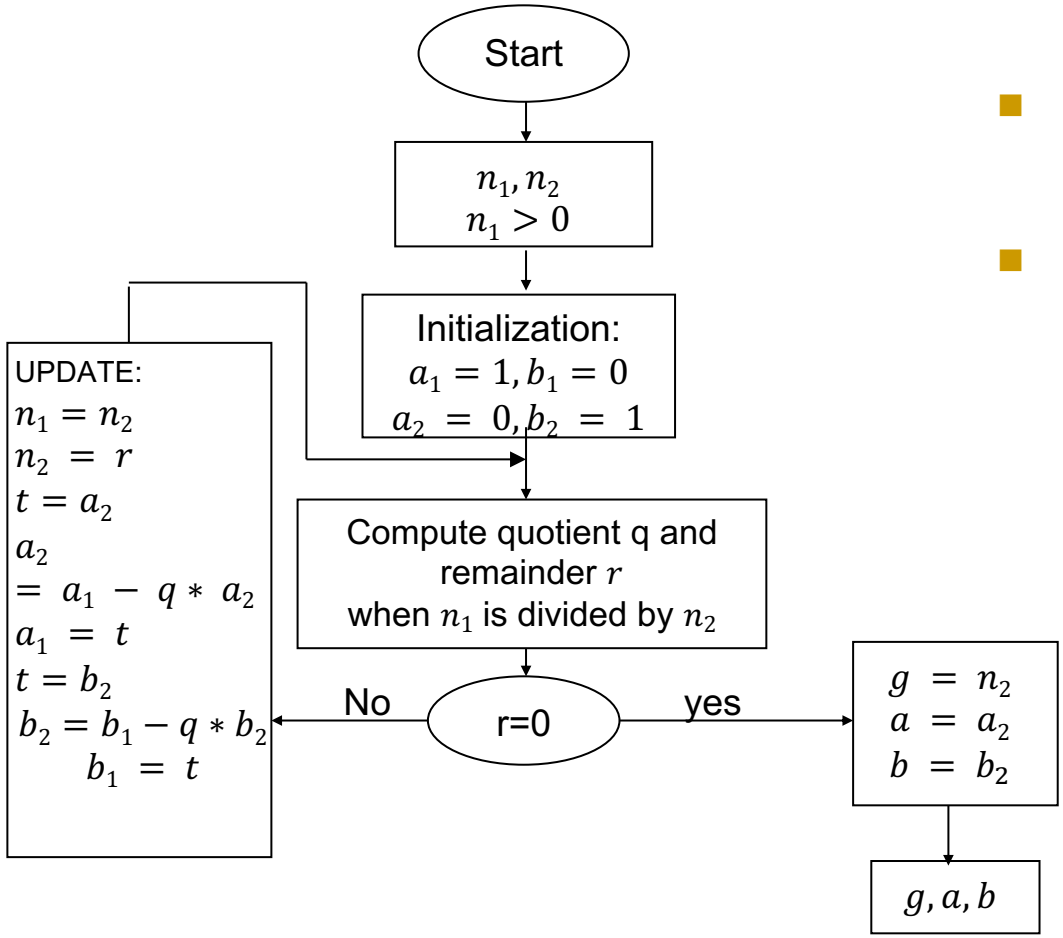
Điểm yếu của mật mã Knapsack

- **Brute Force Attack (tấn công vũ phu)**
  - Với những kẻ không biết trapdoor (a', m, ω), giải mã đòi hỏi phải tìm kiếm vét cạn qua 2^n khả năng của X.
- **Sự đổ vỡ của giải pháp dùng Knapsack (1982-1984).**
  - Shamir-Adleman đã chỉ ra chỗ yếu của GP này bằng cách đi tìm 1 cặp (ω',m') sao cho nó có thể biến đổi ngược a về a' (từ Public key về Private key).
  - 1984, Brickell tuyên bố sự đổ vỡ của hệ thống Knapsack với dung lượng tính toán khoảng 1 giờ máy Cray -1, với 40 vòng lặp chính và cỡ 100 trọng số.

Thuật toán tìm giá trị nghịch đảo theo modul đồng dư

- Việc xây dựng Knapsack với cửa bẫy đòi hỏi phải tính giá trị nghịch đảo của ω theo modul m.
- Thuật toán tìm $x = \omega^{-1} \ mod \ m$, sao cho $x \times \omega = 1 \ (mod \ m)$ được gọi là thuật toán GCD mở rộng hay Euclide mở rộng (GCD - Greatest common divisor - ước số chung lớn nhất).
  - Trong khi đi tìm ƯSCLN của hai số nguyên n_1 và n_2, người ta sẽ tính luôn các giá trị a, b sao cho $GCD(n_1, n_2) = a \times n_1 + b \times n_2$.
  - Từ đó suy ra nếu ta đã biết $(n_1, n_2) = 1$ thì thuật toán này sẽ cho ta tìm được a, b thoả mãn $a \times n_1 + b \times n_2 = 1$, tức là n_1 chính là nghịch đảo của a theo modulo n_2 (tức là m)

Chứng minh tính đúng đắn của thuật toán GCD mở rộng



- Ví dụ tính bằng số: Tìm ngịch đảo của 11 theo modulo 39
- Đặt $n_1 = 39, n_2 = 11$ ta có bảng tính minh họa các bước như sau:

n_1	n_2	r	q	a_1	b_1	a_2	b_2
39	11	6	3	1	0	0	1
11	6	5	1	0	1	1	-3
6	5	1	1	1	-3	-1	4

Nhận xét chung về PKC

- Kể từ năm 1976, nhiều giải pháp cho PKC đã được nêu ra nhưng khá nhiều đã bị phá vỡ: chứng minh được là không an toàn.
- Một hệ thống PKC có thể đáp ứng 2 mục đích:
  - Bảo mật thông tin và truyền tin.
  - Chứng thực và chữ ký điện tử.
- Hai thuật toán đáp ứng các ứng dụng trên thành công nhất là RSA và El-Gamal.
- Nói chung PKC chậm, không thích hợp cho on-line encryption
  - Cần khi yêu cần tính an toàn cao và chấp nhận tốc độ chậm.
  - Ngoài ra người ta thường sử dụng kết hợp PKC và SKC:
    - dùng PKC để tạo khóa bí mật thống nhất chung giữa hai bên truyền tin để thực hiện pha truyền tin chính bằng SKC sau đó.

Hệ mã công khai RSA

- Phát minh năm **1978** bởi **R**ivest, Adi **S**hamir and Leonard **A**dleman
  - Được công bố bởi R L Rivest, A Shamir, L Adleman, "*On Digital Signatures and Public Key Cryptosystems*",Communications of the ACM, vol 21 no 2, pp120-126, Feb 1978
  - Tính an toàn được dựa trên độ khó của bài toán phân tích thừa số nguyên tố của một số rất lớn

Ý tưởng chính

- Thuật toán mã hoá và giải mã là các hàm đồng dư của các luỹ thừa trong trường $Z_n = \{0,1,2,..n-1\}$
  - Mã hoá : $Y \equiv X^e \ (mod\ n)$
  - Decryption: $X \equiv Y^d \ (mod\ n)$
  - Để đảm bảo tính đúng đắn của thuật toán
    - e & d phải thoả mãn: $X^{ed} \equiv X \ (mod\ n)$

Ý tưởng chính

- Định lý Euler: nếu $(X, n) = 1$ thì
  - $\varphi(n)$: số lượng các số $k: 0 < k < n | (k, n)$ $X^{\varphi(n)} \equiv 1 \ (mod\ n) = 1$
  - Nếu $n = p \times q$ (p, q nguyên tố) → $\varphi(n) = (p-1)(q-1)$
- Chọn e và tìm d sao cho $ed \equiv 1 (mod\ \varphi(n))$
  - $d \equiv e^{-1} (mod\ \varphi(n))$
  - $X^{ed} \equiv X^{k\varphi(n)+1} \equiv (X^{\varphi(n)})^k \times X \equiv X \ (mod\ n)$
- Chú ý: để giải mã được → cần biết $\varphi(n)$ → cần biết p, q → vì n rất lớn nên việc phân tích n để tìm p, q là không khả thi

Hệ mã RSA

- **Tạo khoá:**
  - Chọn 2 số nguyên tố rất lớn và có độ lớn tương đương (~512 bit): p, q
  - Tính $n = pq$, và $\varphi(n) = (q-1)(p-1)$
  - Chọn 1 số tự nhiên e tuỳ ý, sao cho $1 < e < \varphi(n)$, và $\gcd(e, \varphi(n)) = 1$
  - Tìm d, sao cho $1 < d < \varphi(n)$ và $ed \equiv 1 \, mod \varphi(n)$
  - **Khoá công khai : (e, n); khoá bí mật : d**
    - **Chú ý: p và q phải giữ bí mật**

Hệ mã RSA

- **Mã hoá**
  - Cho trước bản rõ M biểu diễn dưới dạng nhị phân → convert M sang hệ cơ số 10: $0 < M < n$
  - Dùng khoá công khai (e, n) và mã hoá:

    $$C = M^e \ (mod\ n)$$
- **Giải mã**
  - Cho bản mã C, sử dụng khoá bí mật (d) và giải mã:
    - $M = C^d \ (mod\ n)$
- Tính đúng đắn
  - $C^d \ (mod\ n) \equiv M^{ed} (mod\ n) \equiv M (mod\ n)$

Ví dụ

- Parameters:
  - Select $p = 11$ và $q = 13$
  - $n = 11 * 13 = 143$; $m = (p-1)(q-1) = 10 * 12 = 120$
  - Chọn $e = 37$ ➔ $\gcd(37,120) = 1$
  - Tìm d sao cho: $e \times d \equiv 1 \ (mod\ 120)$ ➔ $d = 13$ $(e \times d = 481)$
- Mã hoá
  - Cắt bản rõ thành các đoạn u bits, $2^u \leq 142$ ➔ $u = 7$
    - Mỗi đoạn sẽ là 1 số tự nhiên từ 1 đến 127
  - Tính $Y = X^e (mod\ n)$

  Ví dụ: $X = (0000010) = 2$, ta có $Y \equiv X^{37} \equiv 12 \ (mod\ 143) \rightarrow Y = (00001100)$
- Giải mã: $X \equiv 12^{13} (mod\ 143) = 2$

Cách tính nghịch đảo đồng dư

- $ed \equiv 1(mod\varphi(n))$
- Định lý Bézout: nếu $d = GCD(a,b)$ thì tồn tại 2 số x, y sao cho $d = xa + yb$ (đồng nhất thức Bézout), x, y được gọi là hệ số của a, b
- Nếu $1 = GCD(e, \varphi(n)) \rightarrow 1 = xe + y\varphi(n) \rightarrow xe \equiv 1(mod\varphi(n)) \rightarrow x \equiv e^{-1}(mod\varphi(n))$
- Phương trình Diophantine: $ax+by=c$
  - Chỉ có nghiệm khi $c : d = \gcd(a,b)$
  - $da_1x + db_1y = dc_1$
  - $a_1x + b_1y = c_1$

Cách tính nghịch đảo đồng dư

- Thuật toán Ơclit tìm ước số chung lớn nhất của 2 số r_0, r_1

$$
\begin{aligned}
r_0 &= q_1 r_1 + r_2, & 0 < r_2 < r_1 \\
r_1 &= q_2 r_2 + r_3, & 0 < r_3 < r_2 \\
&\vdots \\
r_{m-2} &= q_{m-1} r_{m-1} + r_m, & 0 < r_m < r_{m-1} \\
r_{m-1} &= q_m r_m.
\end{aligned}
$$

  - Chứng minh được: $\gcd(r_0, r_1) = \gcd(r_1, r_2) = \cdots = \gcd(r_{m-1}, r_m) = r_m$

Cách tính nghịch đảo đồng dư

- Ví dụ
  - Tìm ước số chung lớn nhất của (252, 198)

$$252 = 198 \times 1 + 54$$
$$198 = 54 \times 3 + 36$$
$$54 = 36 \times 1 + 18$$
$$36 = 18 \times 2 + 0$$

⇒ Gcd(252, 198) = 18

Cách tính nghịch đảo đồng dư

- Ví dụ
  - Giải phương trình nghiệm nguyên: $252x+198y=18$

$$252 = 198 \times 1 + 54$$
$$198 = 54 \times 3 + 36$$
$$54 = 36 \times 1 + 18$$
$$36 = 18 \times 2 + 0$$

$$18 = 54 - 36$$
$$18 = 54 - (198 - 54 \times 3)$$
$$18 = 54 \times 4 - 198$$
$$18 = (252 - 198) \times 4 - 198$$
$$18 = 252 - 198 \times 5$$

$$(x, y) = 1, -5$$

Cách tính đồng dư luỹ thừa

- Tính $x^a (mod\ n)$
- Cách đơn giản nhất:
  - $x^a\ (mod\ n) = x\ (mod\ n) \times x\ (mod\ n) \times$ ...$\times$ $x\ (mod\ n)$
  - → thực hiện phép lấy đồng dư và phép nhân a lần
- Sử dụng phương pháp **bình phương và nhân**

Cách tính đồng dư luỹ thừa
PP: bình phương và nhân

- Biểu diễn a dưới dạng nhị phân: $a = \sum_{i=0}^{l} a_i 2^i$

$z \leftarrow 1$
For $i = l$ down to 0
  $z \leftarrow z^2 \ mod \ n$
  if $a_i = 1$ then
    $z \leftarrow (z \times x) \ (mod \ n)$
  end if
End for
Return z

Tìm số dư của x^{19} khi chia cho n
$19 = 16 + 2 + 1 = 2^4 + 2^1 + 2^0 = 10011$
$z \leftarrow 1$
$i = 4: a_4 = 1; z \leftarrow z^2 \times x \equiv 1^2 \times x \equiv x$
$i = 3; a_3 = 0; z \leftarrow z^2 \equiv x^2$
$i = 2; a_2 = 0; z \leftarrow z^2 \equiv x^4$
$i = 1; a_1 = 1; z \leftarrow z^2 \times x \equiv x^8 \times x \equiv x^9$
$i = 0; a_1 = 1; z \leftarrow z^2 \equiv x^{18} \times x \equiv x^{19}$

Tìm số dư của 3^{19} khi chia cho 5
$19 = 10011$
$z \leftarrow 1$
$i = 4: a_4 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
$i = 3; a_3 = 0; z \leftarrow 3^2 \equiv -1$
$i = 2; a_2 = 0; z \leftarrow (-1)^2 \equiv 1$
$i = 1; a_1 = 1; z \leftarrow 1^2 \times 3 \equiv 3$
$i = 0; a_1 = 1; z \leftarrow 3^2 \times 3 \equiv -3 \equiv 2$

- $a = \sum_{i=0}^{l} a_i 2^i$
- $x^a (mod\ n) = x^{\sum_{i=0}^{l} a_i 2^i} = x^{a_0 + a_1 \times 2 + a_2 \times 2^2 + \cdots + a_l \times 2^l}$
- $x^{a_0} \left(x^{a_1 + a_2 \times 2 + \cdots + a_l \times 2^{l-1}} \right)^2$

$z \leftarrow 1$
For $i = l$ down to 0
  $z \leftarrow z^2\ mod\ n$
  if $a_i = 1$ then
    $z \leftarrow (z \times x)\ (mod\ n)$
  end if
End for
Return z

$z \leftarrow (z \times x^{a_i})\ (mod\ n)$

$z \leftarrow (z^2 \times x^{a_i})\ (mod\ n)$

$$i = l \rightarrow x^{a_l}$$
$$i = l - 1 \rightarrow (x^{a_l})^2 x^{a_{l-1}} = x^{a_l \times 2 + a_{l-1}}$$
$$i = l - 2 \rightarrow (x^{2a_l + a_{l-1}})^2 x^{a_{l-2}} = x^{a_l \times 2^2 + a_{l-1} \times 2 + a_{l-2}}$$

Bài tập

1. Tính
   1. $28^{-1} \bmod 75$
   2. $17^{-1}\ mod\ 101$
   3. $357^{-1}\ mod\ 1234$
   4. $3125^{-1}\ mod\ 9987$
2. Chứng minh: $X^{(p-1)(q-1)} \equiv 1\ (mod\ pq)$ với p, q nguyên tố
3. Viết đoạn giả mã của thuật toán tính nghịch đảo đồng dư
4. Chứng minh tính đúng đắn của phương pháp bình phương và nhân
5. Tính
   1. $9726^{3533}\ (mod\ 11413)$

Bài tập lớn

1. Viết chương trình phá mã thế (1 bảng thế) bằng phương pháp thống kê
2. Viết chương trình phá mã vigenere (mã đã bảng thế)
3. Viết chương trình mã hoá và phá mã RSA như sau
    1. Mã hoá:
        1. Input: bản rõ, khoá công khai (d, n)
        2. Bản rõ là 1 văn bản tiếng anh. Mỗi từ được encode theo bảng chữ cái, ví dụ như sau:
            - DOG → $3\times26^2 + 14\times26 + 6 = 2398$
            - CAT → $2\times26^2 + 0\times26 + 6 = 19$
        - Mỗi số (tương ứng với 1 word) trong bản rõ được mã hoá bằng mã RSA với khoá công khai (d, n)
            - Áp dụng giải thuật bình phương và nhân để tính đồng dư luỹ thừa
    2. Phá mã
        1. Phân tích n thành tích của 2 thừa số nguyên tố
        2. Tính $\varphi(n)$
        3. Tìm khoá bí mật e
            - Áp dụng thuật toán Ơclit mở rộng
        4. Tìm bản rõ
            - Áp dụng giải thuật bình phương và nhân để tính đồng dư luỹ thừa