BG P2. Cơ sở toán học của ATTT

Bài 2.2. Xác suất và thuật toán xác suất.

2.2.1. Khái niệm xác suất.

Ta xét một tập hợp Ω , được gọi là *không gian các sự kiện sơ cấp* (hay không gian mẫu). Các phần tử của Ω, tức các sự kiện sơ cấp hay các mẫu, có thể được xem như các kết quả có thể có (và loại trừ lẫn nhau) của một thực nghiệm nào đó. Về sau ta chỉ xét các không gian rời rạc, tức tập Ω là hữu hạn, giả sử $\Omega = \{s_1, s_2, \ldots, s_n\}$.

Một *phân bố xác suất* P trên Ω được định nghĩa là một tập các số thực không âm $P = \{ p_1, p_2, \ldots, p_n\}$ có tổng $\sum p_i = 1$. Số p_i được coi là xác suất của sự kiện sơ cấp s_i .

Một tập con $E \subseteq \Omega$ được gọi là một *sự kiện* . Xác suất của sự kiện E được định nghĩa bởi $p(E) = \sum_{s \in E} p(s)$.

Giả sử E là một sự kiện trong không gian xác suất Ω. Ta định nghĩa *sự kiện bù* của E, ký hiệu $\overline{E}$, là sự kiện gồm tất cả các sự kiện sơ cấp trong Ω mà không thuộc E . Dùng các thuật ngữ của lý thuyết tập hợp, ta có thể định nghĩa các*sự kiện hợp* $E_1 \cup E_2$ và *sự kiện giao* $E_1 \cap E_2$ của hai sự kiện E_1 và E_2 bất kỳ. Và ta có:

1) Giả sử E là một sự kiện. Khi đó $0 \leq p(E) \leq 1$ và $p(\overline{E}) = 1 - p(E)$. Ngoài ra, $p(\Omega) = 1$ và $p(\varnothing) = 0$.

2) Giả sử E_1 và E_2 là hai sự kiện. Nếu $E_1 \subseteq E_2$ thì $p(E_1) \leq p(E_2)$. Và có $p(E_1 \cup E_2) + p(E_1 \cap E_2) = p(E_1) + p(E_2)$. Do đó $p(E_1 \cup E_2) = p(E_1) + p(E_2)$ khi và chỉ khi $E_1 \cap E_2 = \varnothing$, tức là khi E_1 và E_2 là hai sự kiện loại trừ lẫn nhau.

Cho E_1 và E_2 là hai sự kiện, với $p(E_2) > 0$. Ta định nghĩa *xác suất có điều kiện* của E_1 khi có E_2 , ký hiệu $p(E_1|E_2)$, là

$$p(E_1|E_2) = \frac{p(E_1 \cap E_2)}{p(E_2)}.$$

Từ định nghĩa ta suy ra *công thức Bayes* :

$$p(E_1|E_2) = \frac{p(E_1).p(E_2|E_1)}{p(E_2)}. \quad .$$

Ta nói hai sự kiện E_1 và E_2 là *độc lập* với nhau, nếu $p(E_1 \cap E_2) = p(E_1).p(E_2)$. Khi đó ta có: $p(E_1|E_2) = p(E_1)$ và $p(E_2|E_1) = p(E_2)$.

Giả sử Ω là một không gian mẫu với một phân bố xác suất P . Ta gọi một *đại lượng ngẫu nhiên* ξ trên Ω là một ánh xạ gán cho mỗi $s \in \Omega$ một số thực $\xi(s)$. Hiển nhiên, nếu ξ và η là các đại lượng ngẫu nhiên trên Ω, thì $\xi + \eta$, $\xi.\eta$ được định nghĩa bởi :

$$\forall s \in \Omega: (\xi + \eta)(s) = \xi(s) + \eta(s) , (\xi.\eta)(s) = \xi(s).\eta(s).$$

cũng là các đại lương ngẫu nhiên trên Ω .

Giả sử ξ là một đại lượng ngẫu nhiên trên không gian mẫu Ω. Điều đó có nghĩa là với mọi $s \in \Omega$, ξ lấy giá trị bằng $\xi(s)$ với xác suất $p(s)$. Ta định nghĩa *giá trị kỳ vọng* (hay *trung bình*, hay *kỳ vọng toán học*) của ξ là

$$E(\xi) = \sum_{s\in\Omega} \xi(s).p(s).$$

Phương sai của đại lượng ngẫu nhiên ξ có giá trị trung bình μ được định nghĩa là Var (ξ) = $E((\xi - \mu)^2)$.

Căn bậc hai không âm của Var (ξ)được gọi là *độ lệch chuẩn* của ξ.

2.2.2. Tính bí mật hoàn toàn của một hệ mật mã.

Năm 1949, C. Shannon công bố công trình *Lý thuyết truyền thông của các hệ bí mật* , đưa ra nhiều quan niệm làm cơ sở cho việc đánh giá tính bí mật của các hệ mật mã, trong đó có khái niệm *tính bí mật hoàn toàn* của một hệ mật mã được định nghĩa như sau: Cho hệ mật mã $S = (P, C, K, E, D)$. Giả thử trên các tập P , C và K được xác định tương ứng các phân bố xác suất $p_P(.)$, $p_C(.)$ và $p_K(.)$. Như vậy, với mọi $x \in P$, $y \in C$ và $K \in K$, $p_P(x)$, $p_C(y)$ và $p_K(K)$ tương ứng là các xác suất để ký tự bản rõ là x, ký tự bản mã là y và khoá là K. Xác suất có điều kiện, chẳng hạn, xác suất của việc bản rõ là x khi bản mã là y, được ký hiệu là $p_P(x \mid y)$. Một hệ mật mã được gọi là *bí mật hoàn toàn*, nếu với mọi $x \in P$, $y \in C$ có $p_P(x \mid y) = p_P(x)$. Điều đó có nghĩa là việc biết xác suất bản rõ là x là như nhau dù biết hay không biết bản mã là y ; nói cách khác, có thông tin về bản mã không cho ta biết gì thêm về bản rõ; bản rõ và bản mã, với tư cách các biến ngẫu nhiên, là độc lập với nhau. Ta có định lý sau đây:

Định lý 2.2.1. *Giả sử* $S = (P, C, K, E, D)$ *là một hệ mật mã với điều kiện* $|P| = |C| = |K|$ *, tức các tập* P , C , K *có số các phần tử bằng nhau. Khi đó, hệ là bí mật hoàn toàn nếu và chỉ nếu mỗi khoá* $K \in K$ *được dùng với xác suất bằng nhau là* $1/|K|$ *, và với mọi* $x \in P$, $y \in C$ *có một khoá duy nhất* $K \in K$ *sao cho* $e_K(x) = y$.

Chứng minh. a) Giả thử hệ S là bí mật hoàn toàn. Khi đó, với mọi $x \in P$ và $y \in C$ có $p_P(x \mid y) = p_P(x)$. Ngoài ra ta có thể giả thiết $p_C(y) > 0$ với mọi $y \in C$. Từ đó theo công thức Bayes ta có $p_C(y \mid x) = p_C(y) > 0$. Điều đó có nghĩa là có ít nhất một khoá K sao cho $e_K(x) = y$. Vì vậy, nếu cố định một $x \in P$ thì ta có

$$|C| = |\{ e_K(x): K \in K \}| \le |K| .$$

Theo giả thiết của định lý, $|C| = |K|$, do đó

$$|\{ e_K(x): K \in K \}| = |K| .$$

Nhưng điều này lại có nghĩa là không thể có hai khoá $K_1 \neq K_2$ sao cho $e_{K_1}(x) = e_{K_2}(x)$. Vậy ta đã chứng minh được với mọi $x \in P$ và $y \in C$ có đúng một khoá K sao cho $e_K(x) = y$.

Ký hiệu $n = |K|$ và đặt $K = \{K_1,..., K_n\}$. Cố định một $y \in C$ và giả thử $e_{K_i}(x_i) = y$ với $P = \{x_1,....., x_n\}$, $1 \leq i \leq n$. Dùng công thức Bayes ta lại có

$$p_P(x_i|y) = \frac{p_C(y|x_i).p_P(x_i)}{p_C(y)} = \frac{p_K(K_i).p_P(x_i)}{p_C(y)}.$$

Do giả thiết hệ là bí mật hoàn toàn, ta có $p_P(x_i \mid y) = p_P(x_i)$. Từ đó suy ra với mọi i, $1 \leq i \leq n$, $p_K(K_i) = p_C(y)$. Vậy các $p_K(K_i)$ $(1 \leq i \leq n)$ đều bằng nhau, và do đó đều bằng $1/|K|$.

b) Bây giờ ta chứng minh điều ngược lại. Giả thiết $p_K(K) = 1/|K|$ với mọi $K \in K$, và với mọi $x \in P$, $y \in C$ có đúng một khoá $K \in K$ sao cho $e_K(x) = y$. Ta tính:

$$p_C(y) = \sum_{K \in \mathcal{K}} p_K(K).p_P(d_K(y)) = \sum_{K \in \mathcal{K}} \frac{1}{|\mathcal{K}|} p_P(d_K(y)) = \\ = \frac{1}{|\mathcal{K}|} \sum_{K \in \mathcal{K}} p_P(d_K(y)).$$

Khi K chạy qua tập khoá K thì $d_K(y)$ chạy qua tập P, do đó

$$\sum_{K \in \mathcal{K}} p_P(d_K(y)) = \sum_{x \in \mathcal{P}} p_P(x) = 1,$$

và ta được $p_C(y) = 1/|K|$ với mọi $y \in C$.

Mặt khác, gọi K là khoá duy nhất mà $e_K(x) = y$, ta có

$$p_C(y \mid x) = p_K(K) = 1/|K|.$$

Dùng công thức B**ayes ta lại được** với mọi $x \in P$, $y \in C$:

$$p_P(x|y) = \frac{p_P(x).p_C(y|x)}{p_C(y)} = \frac{p_P(x).1/|\mathcal{K}|}{1/|\mathcal{K}|} = p_P(x).$$

Vậy hệ là bí mật hoàn toàn. Định lý được chứng minh.

2.2.3. Thuật toán xác suất:

Khái niệm *thuật toán* mà ta thường hiểu là thuật toán tất định, đó là một tiến trình thực hiện các phép toán trên dữ liệu đầu vào và cho kết quả ở đầu ra. Theo D.E. Knuth, thuật toán có 5 thuộc tính cơ bản: tính *hữu hạn*, thuật toán luôn kết thúc sau một số hữu hạn bước; tính *xác định*, mỗi bước của thuật toán phải được xác định một cách chính xác; tập hợp *đầu vào* và *đầu ra* của mỗi thuật toán cũng được xác định rõ ràng; và tính *hiệu quả*, mọi phép toán trong thuật toán phải là cơ bản, có thể được thực hiện chính xác trong một thời gian xác định. Thuật toán là khái niệm cơ bản đối với việc lập trình trên máy tính, và đã được sử dụng rất phổ biến. Nhưng

như ta biết, đối với nhiều bài toán trong thực tế, không phải bao giờ ta cũng tìm được thuật toán giải chúng với độ phức tạp tính toán chấp nhận được (ta sẽ xét qua vấn đề này trong một tiết sau). Vì vậy, cùng với các thuật toán tất định, đối với một số bài toán ta sẽ xét thêm các thuật toán xác suất, đó là những thuật toán mà cùng với dữ liệu đầu vào ta bổ sung thêm giá trị của một đại lượng ngẫu nhiên tương ứng nào đó, thường là các số ngẫu nhiên.

Các thuật toán xác suất thường được xây dựng cho các bài toán quyết định, tức các bài toán xác định trên một tập hợp dữ liệu sao cho ứng với mỗi dữ liệu bài toán có một trả lời *có* hoặc *không* . Người ta chia các thuật toán xác suất thành hai loại: loại thuật toán *Monte Carlo* và loại thuật toán *Las Vegas* . Thuật toán Monte Carlo luôn kết thúc với kết quả *có* hoặc *không* đối với mọi dữ liệu đầu vào bất kỳ; còn thuật toán Las Vegas tuy cũng kết thúc với mọi dữ liệu, nhưng có thể kết thúc với một thông báo *không có trả lời* có hoặc không. Thuật toán Monte Carlo được gọi là *thiên về có*, nếu nó cho trả lời *có* thì trả lời đó chắc chắn là đúng, còn nếu nó cho trả lời *không* thì trả lời đó có thể sai với một xác suất ε nào đó. Tương tự, một thuật toán Monte Carlo được gọi là *thiên về không*, nếu nó cho trả lời *không* thì trả lời đó chắc chắn là đúng, còn nếu nó cho trả lời *có* thì trả lời đó có thể sai với một xác suất ε nào đó. Còn với thuật toán Las Vegas, nếu nó kết thúc với trả lời *có* hoặc *không* , thì trả lời đó chắc chắn đúng, và nó có thể kết thúc với thông báo *không có trả lời* với một xác suất ε nào đó. Trong tiết 2.8 sau đây ta sẽ cho vài thí dụ cụ thể về một số thuật toán xác suất thuộc cả hai loại đó.

Bài 2.3. Độ phức tạp tính toán.

2.3.1. Khái niệm về độ phức tạp tính toán.

Lý thuyết thuật toán và các hàm số tính được ra đời từ những năm 30 của thế kỷ 20 đã đặt nền móng cho việc nghiên cứu các vấn đề "tính được", "giải được" trong toán học, đưa đến nhiều kết quả rất quan trọng và lý thú. Nhưng từ cái "tính được" một cách trừu tượng, hiểu theo nghĩa tiềm năng,đến việc tính được trong thực tế của khoa học tính toán bằng máy tính điện tử, là cả một khoảng cách rất lớn. Biết bao nhiêu thứ được chứng minh là tính được một cách tiềm năng, nhưng không tính được trong thực tế, dù có sự hỗ trợ của những máy tính điện tử ! Vấn đề là do ở chỗ những đòi hỏi về không gian vật chất và về thời gian để thực hiện các tiến trình tính toán nhiều khi vượt quá xa những khả năng thực tế. Từ đó, vào khoảng giữa những năm 60 (của thế kỷ trước), một lý thuyết về độ phức tạp tính toán bắt đầu được hình thành và phát triển nhanh chóng, cung cấp cho chúng ta nhiều hiểu biết sâu sắc về bản chất phức tạp của các thuật toán và các bài toán, cả những bài toán thuần tuý lý thuyết đến những bài toán thường gặp trong thực tế. Sau đây ta giới thiệu sơ lược một số khái niệm cơ bản và vài kết quả sẽ được dùng đến của lý thuyết đó.

Trước hết, ta hiểu *độ phức tạp tính toán* (về không gian hay về thời gian) của một tiến trình tính toán là số ô nhớ được dùng hay số các phép toán sơ cấp được thực hiện trong tiến trình tính toán đó.

Dữ liệu đầu vào đối với một thuật toán thường được biểu diễn qua các từ trong một bảng ký tự nào đó. *Độ dài của một từ* là số ký tự trong từ đó.

Cho một thuật toán A trên bảng ký tự Σ (tức có đầu vào là các từ trong Σ) . *Độ phức tạp tính toán* của thuật toán A được hiểu là một hàm số $f_A(n)$ sao cho với mỗi số n , $f_A(n)$ là số ô nhớ, hay số phép toán sơ cấp tối đa mà A cần để thực hiện tiến trình tính toán của mình trên các dữ liệu vào có độ dài $\leq n$. Ta nói thuật toán A có độ phức tạp thời gian *đa thức* , nếu có một đa thức $P(n)$ sao cho với mọi n đủ lớn ta có $f_A(n) \leq P(n)$, trong đó $f_A(n)$ là độ phức tạp tính toán theo thời gian của A.

Về sau khi nói đến các bài toán, ta hiểu đó là các *bài toán quyết định* , mỗi bài toán P như vậy được xác định bởi:

- một tập các dữ liệu vào I (trong một bảng ký tự Σ nào đó),

- một câu hỏi Q trên các dữ liệu vào, sao cho với mỗi dữ liệu vào $x \in I$, câu hỏi Q có một trả lời *đúng* hoặc *sai*.

Ta nói bài toán quyết định P là *giải được* , nếu có thuật toán để giải nó, tức là thuật toán làm việc có kết thúc trên mọi dữ liệu vào của bài toán, và cho kết quả *đúng* hoặc *sai* tuỳ theo câu

hỏi *Q* trên dữ liệu đó có trả lời đúng hoặc sai. B**ài toán *P*** là *giải được trong thời gian đa thức* , nếu có thuật toán giải nó với độ phức tạp thời gian đa thức. Sau đây là vài thí dụ về các bài toán quyết định:

Bài toán *SATISFIABILITY* (viết tắt là *SAT*):

- mỗi dữ liệu vào là một công thức *F* của lôgich mệnh đề, được viết dưới dạng hội chuẩn tắc, tức dạng hội của một số các "clause".

- Câu hỏi là: công thức *F* có thoả được hay không ?

Bài toán *CLIQUE* :

- mỗi dữ liệu vào là một graph *G* và một số nguyên *k* .

- Câu hỏi là: Graph *G* có một clique với $\geq k$ đỉnh hay không ? (một clique của *G* là một graph con đầy đủ của *G*).

Bài toán *KNAPSACK* :

- mỗi dữ liệu là một bộ *n* +1 số nguyên dương $I = (s_1,...,s_n ; T)$.

- Câu hỏi là: có hay không một vectơ Boole $(x_1,...,x_n)$ sao cho

$$\sum_{i=1}^{n} x_i . s_i = T ?$$

(vectơ boole là vectơ có các thành phần là 0 hoặc 1).

Bài toán thặng dư bậc hai :

- mỗi dữ liệu gồm hai số nguyên dương (*a* , *n*).

- Câu hỏi là: *a* có là thặng dư bậc hai theo mod*n* hay không ?

Bài toán hợp số :

- mỗi dữ liệu là một số nguyên dương *N.*

- Câu hỏi: *N* là hợp số hay không ? Tức có hay không hai số *m, n* >1 sao cho *N* =*m* . *n* ?

Tương tự, nếu đặt câu hỏi là "*N* là số nguyên tố hay không?" thì ta được bài toán số nguyên tố .

Đối với tất cả các bài toán kể trên, trừ bài toán hợp số và số nguyên tố, cho đến nay người ta đều chưa tìm được thuật toán giải chúng trong thời gian đa thức.

2.3.2. Lớp phức tạp.

Ta xét một vài lớp các bài toán được xác định theo độ phức tạp tính toán của chúng. Trước hết, ta định nghĩa P là lớp tất cả các bài toán có thể giải được bởi thuật toán trong thời gian đa thức.

Giả sử cho hai bài toán P_1 và P_2 với các tập dữ liệu trong hai bảng ký tự tương ứng là Σ_1 và Σ_2 . Một thuật toán $f : \Sigma_1^* \to \Sigma_2^*$ được gọi là một *phép qui dẫn* bài toán P_1 về bài toán P_2 , nếu nó biến mỗi dữ liệu *x* của bài toán P_1 thành một dữ liệu *f* (*x*) của bài toán P_2 , và sao cho câu hỏi của P_1 trên *x* có trả lời đúng khi và chỉ khi câu hỏi của P_2 trên *f* (*x*) cũng có trả lời đúng. Ta nói bài toán P_1 *qui dẫn được* về bài toán P_2 *trong thời gian đa thức* , và ký hiệu $P_1 \propto P_2$, nếu có thuật

toán f với độ phức tạp thời gian đa thức qui dẫn bài toán P_1 về bài toán P_2 .Ta dễ thấy rằng, nếu $P_1 \propto P_2$ và $P_2 \in \Pi$, thì cũng có $P_1 \in \Pi$.

Một lớp quan trọng các bài toán đã được nghiên cứu nhiều là lớp các bài toán khá thường gặp trong thực tế nhưng cho đến nay chưa có khả năng nào chứng tỏ là chúng có thể giải được trong thời gian đa thức. Đó là lớp các bài toán $N\Pi$-*đầy đủ* mà ta sẽ định nghĩa sau đây:

Cùng với khái niệm thuật toán tất định thông thường (có thể mô tả chính xác chẳng hạn bởi máy Turing tất định), ta xét khái niệm thuật toán *không đơn định* với một ít thay đổi như sau: nếu đối với máy Turing tất định, khi máy đang ở một trạng thái q và đang đọc một ký tự a thì cặp (q,a) xác định duy nhất một hành động kế tiếp của máy, còn đối với máy Turing không đơn định, ta qui ước rằng (q,a) xác định không phải duy nhất mà là một tập hữu hạn các hành động kế tiếp; máy *có thể* thực hiện trong bước kế tiếp một trong các hành động đó. Như vậy, đối với một dữ liệu vào x , một thuật toán không đơn định (được xác định chẳng hạn bởi một máy Turing không đơn định) không phải chỉ có một tiến trình tính toán duy nhất, mà có thể có một số hữu hạn những tiến trình tính toán khác nhau. Ta nói thuật toán không đơn định A *chấp nhận* dữ liệu x , nếu với dữ liệu vào x thuật toán A có ít nhất một tiến trình tính toán kết thúc ở trạng thái chấp nhận (tức với kết quả *đúng*). Một bài toán P được gọi là *giải được bởi thuật toán không đơn định trong thời gian đa thức* nếu có một thuật toán không đơn định A và một đa thức $p(n)$ sao cho với mọi dữ liệu vào x có độ dài n , $x \in P$ (tức câu hỏi của P có trả lời đúng trên x) khi và chỉ khi thuật toán A chấp nhận x bởi một tiến trình tính toán có độ phức tạp thời gian $\leq p(n)$. Ta ký hiệu lớp tất cả các bài toán giải được bởi thuật toán không đơn định trong thời gian đa thức là $N\Pi$.

Người ta đã chứng tỏ được rằng tất cả những bài toán trong các thí dụ kể trên và rất nhiều các bài toán tổ hợp thường gặp khác đều thuộc lớp $N\Pi$, dù rằng hầu hết chúng đều chưa được chứng tỏ là thuộc Π. Một bài toán P được gọi là $N\Pi$.-*đầy đủ*, nếu $P \in N\Pi$ và với mọi $Q \in N\Pi$ đều có $Q \propto P$.

Lớp $N\Pi$ có một số tính chất sau đây:

1) $\Pi \subseteq N\Pi$,
2) Nếu $P_1 \propto P_2$ và $P_2 \in N\Pi$, thì $P_1 \in N\Pi$.
3) Nếu P_1 ,$P_2 \in N\Pi$, $P_1 \propto P_2$, và P_1 là $N\Pi$-đầy đủ, thì P_2 cũng là $N\Pi$ -đầy đủ.
4) Nếu có P sao cho P là $N\Pi$-đầy đủ và $P \in \Pi$, thì $\Pi = N\Pi$.

Từ các tính chất đó ta có thể xem rằng trong lớp $N\Pi$, Π là lớp con các bài toán " dễ " nhất, còn các bài toán $N\Pi$-đầy đủ là các bài toán " khó " nhất; nếu có ít nhất một bài toán $N\Pi$-đầy đủ được chứng minh là thuộc Π , thì lập tức suy ra $\Pi = N\Pi$, dù rằng cho đến nay tuy đã có rất nhiều cố gắng nhưng toán học vẫn chưa tìm được con đường nào hy vọng đi đến giải quyết vấn đề [$\Pi = N\Pi$?], thậm chí vấn đề đó còn được xem là một trong 7 vấn đề khó nhất của toán học trong thiên niên kỷ mới!

2.3.3. Hàm một phía và cửa sập một phía.

Khái niệm *độ phức tạp tính toán* cung cấp cho ta một cách tiếp cận mới đối với vấn đề *bí mật* trong các vấn đề bảo mật và an toàn thông tin. Dù ngày nay ta đã có những máy tính điện tử có tốc độ tính toán cỡ hàng tỷ phép tính một giây đồng hồ, nhưng với những thuật toán có độ phức tạp tính toán cỡ $f(n) = 2^n$, thì ngay với những dữ liệu có độ dài khoảng $n = 1000$, việc thực hiện các thuật toán đó đã không thể xem là khả thi, vì nó đòi hỏi thực hiện khoảng 10^{300} phép tính! Như vậy, một giải pháp mật mã chẳng hạn có thể xem là có độ bảo mật cao, nếu để giải mã cần phải thực hiện một tiến trình tính toán có độ phức tạp rất lớn. Do đó, việc phát hiện và sử dụng các hàm số có độ phức tạp tính toán rất lớn là có ý nghĩa hết sức quan trọng đối với việc xây dựng các giải pháp về mật mã và an toàn thông tin.

Hàm số số học $y = f(x)$ được gọi là *hàm một phía* (one-way function), nếu việc tính thuận từ x ra y là "dễ", nhưng việc tính ngược từ y tìm lại x là rất "khó", ở đây các tính từ "dễ" và "khó" không có các định nghĩa chính xác mà được hiểu một cách thực hành, ta có thể hiểu chẳng hạn dễ là tính được trong thời gian đa thức (với đa thức bậc thấp), còn khó là không tính được trong thời gian đa thức! Thực tế thì cho đến hiện nay, việc tìm và chứng minh một hàm số nào đó là không tính được trong thời gian đa thức còn là việc rất khó khăn, cho nên "khó" thường khi chỉ được hiểu một cách đơn giản là chưa tìm được thuật toán tính nó trong thời gian đa thức! Với cách hiểu tương đối như vậy về "dễ" và "khó", người ta đã đưa ra một số thí dụ sau đây về các hàm một phía:

Thí dụ 1. Cho p là một số nguyên tố, và α là một phần tử nguyên thuỷ mod$p$. Hàm số $y = \alpha^x \bmod p$ (từ Z_p^* vào Z_p^*) là một hàm một phía, vì hàm ngược của nó, tính từ y tìm x mà ta ký hiệu $x = \log_a(y)$, là một hàm có độ phức tạp tính toán rất lớn.

Thí dụ 2. Cho $n = p.q$ là tích của hai số nguyên tố lớn. Hàm số $y = x^2 \bmod n$ (từ Z_n vào Z_n) cũng được xem là một hàm một phía.

Thí dụ 3. Cho $n = p.q$ là tích của hai số nguyên tố lớn, và a là một số nguyên sao cho $\gcd(a, \phi(n)) = 1$. Hàm số $y = x^a \bmod n$ (từ Z_n vào Z_n) cũng là một hàm một phía, nếu giả thiết là biết n nhưng không biết p,q.

Hàm $y = f(x)$ được gọi là *hàm cửa sập một phía* (trapdoor one-way function), **nếu việc tính thuận từ *x* ra *y*** là "dễ", việc tính ngược từ y tìm lại x là rất "khó", nhưng có một cửa sập z để với sự trợ giúp của cửa sập z thì việc tính x từ y và z lại trở thành dễ.

Thí dụ 4 (tiếp tục thí dụ 3). Hàm số $y = x^a \bmod n$ khi biết p và q là hàm cửa sập một phía. Từ x tính y là dễ, từ y tìm x (nếu chỉ biết n, a) là rất khó, nhưng vì biết p và q nên biết $\phi(n) = (p-1)(q-1)$, và dùng thuật toán Euclide mở rộng tìm được b sao cho $a.b \equiv 1 \pmod{\phi(n)}$, từ đó dễ tính được $x = y^b \bmod n$. ở đây, có thể xem b là cửa sập.

Bài 2.4. Số nguyên tố. Phân tích thành thừa số. Logarit rời rạc.

Trong tiết này ta sẽ xét ba bài toán có vai trò quan trọng trong lý thuyết mật mã, đó là ba bài toán: thử tính nguyên tố của một số nguyên, phân tích một số nguyên thành tích của các thừa số nguyên tố, và tính logarit rời rạc của một số theo một môđuyn nguyên tố.

2.4.1. Thử tính nguyên tố của một số.

Bài toán đặt ra rất đơn giản: Cho một số nguyên dương *n* bất kỳ. Hãy thử xem *n* có là số nguyên tố hay không? Bài toán được đặt ra từ những buổi đầu của số học, và trải qua hơn 2000 năm đến nay vẫn là một bài toán chưa có được những cách giải dễ dàng. Bằng những phương pháp đơn giản như phương pháp sàng Euratosthène, từ rất sớm người ta đã xây dựng được các bảng số nguyên tố đầu tiên, rồi tiếp tục bằng nhiều phương pháp khác tìm thêm được nhiều số nguyên tố lớn. Tuy nhiên, chỉ đến giai đoạn hiện nay của lý thuyết mật mã hiện đại, nhu cầu sử dụng các số nguyên tố và thử tính nguyên tố của các số mới trở thành một nhu cầu to lớn và phổ biến, đòi hỏi nhiều phương pháp mới có hiệu quả hơn. Trong mục này ta sẽ lược qua vài tính chất của số nguyên tố, sau đó giới thiệu môt vài phương pháp thử tính nguyên tố của một số nguyên bất kỳ. Ta đã biết một số tính chất sau đây của các số nguyên tố và hợp số (trong các phát biểu dưới đây, ký hiệu $|A|$ chỉ cho số phần tử của tập hợp *A*):

1. <u>Tiêu chuẩn Euler-Solovay-Strassen:</u>

a) Nếu *n* là số nguyên tố, thì với mọi số nguyên dương *a* [*n* -1:

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \, mod \, n.$$

b) Nếu *n* là hợp số , thì

$$\left|\left\{a : 1 \le a \le n-1, \left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \, mod \, n\right\}\right| \le \frac{n-1}{2}.$$

2. <u>Tiêu chuẩn Solovay-Strassen-Lehmann :</u>

a) Nếu *n* là số nguyên tố, thì với mọi số nguyên dương *a* [*n* -1:

$$a^{(n-1)/2} \equiv \pm 1 (mod \, n).$$

b) Nếu *n* là hợp số, thì

$$\left|\left\{a : 1 \le a \le n-1, a^{(n-1)/2} \equiv \pm 1 \, mod \, n\right\}\right| \le \frac{n-1}{2}.$$

3. <u>Tiêu chuẩn Miller-Rabin :</u>

a) Cho *n* là số nguyên lẻ, ta viết *n* - 1 = 2^e .*u*, với *u* là số lẻ. Nếu *n* là số nguyên tố, thì với mọi số nguyên dương *a* [*n* -1:

$$(a^u \equiv 1 \, mod \, n) \vee \exists k < e (a^{2^k.u} \equiv -1 \, mod \, n).$$

b) Nếu *n* là hợp số, thì $\left|\left\{a : 1 \le a \le n-1, (a^u \equiv 1 \, mod \, n) \vee \exists k < e(a^{2^k.u} \equiv -1 \, mod \, n)\right\}\right| \le \frac{n-1}{4}$.

Các tiêu chuẩn kể trên là cơ sở để ta xây dựng các thuật toán xác suất kiểu Monte-Carlo thử tính nguyên tố (hay hợp số) của các số nguyên. Chẳng hạn, từ tiêu chuẩn thứ nhất ta có ***thuật toán Euler-Solovay-Strassen*** sau đây:

Dữ liệu vào: số nguyên dương *n* và *t* số ngẫu nhiên $a_1,...,a_t$ (1[a_i[*n* -1),

1. **for** *i* = 1 **to *t*** do
2. **if** $\left(\frac{a_i}{n}\right) \equiv a_i^{(n-1)/2} \, mod \, n$, **then**
3. **answer** "*n* là số nguyên tố "
4. else
5. **answer** "*n* là hợp số " and **quit**

Thuật toán này nếu cho trả lời "*n* là hợp số " thì đúng *n* là hợp số, nhưng nếu nó cho trả lời "*n* là số nguyên tố " thì trả lời đó có thể sai với một xác suất ε nào đó. Như vậy, thuật toán đó là một thuật toán xác suất Monte-Carlo *thiên về có* nếu xem nó là thuật toán thử tính *là hợp số* ; còn nó là một thuật toán xác suất *thiên về không* nếu xem nó là thuật toán thử tính *nguyên tố* của các số nguyên.

Tương tự như vậy, dựa vào các tiêu chuẩn 2 và 3 ta cũng có thể xây dựng các thuật toán xác suất Solovay-Strassen-Lehmann và Miller-Rabin kiểu Monte-Carlo để thử tính nguyên tố (hay là hợp số) của các số nguyên. Hai thuật toán đó chỉ khác thuật toán Euler-Solovay-Strassen kể trên ở chỗ công thức trong hàng lệnh thứ 2 cần được thay tương ứng bởi

$$a^{(n-1)/2} \equiv \pm 1 \, \boldsymbol{mod} \, n$$

hay

$$(a^u \equiv 1 \, mod \, n) \vee \exists k < e(a^{2^k.u} \equiv -1 \, mod \, n)$$

trong đó *u* và *e* được xác định bởi: *n* - 1 = $2^e.u$, *u* là số lẻ.

Xác suất sai lầm ε khi nhận được kết quả "*n* là số nguyên tố " trong các thuật toán đó được tính như sau: Giả sử *n* là một số lẻ trong khoảng *N* và 2*N* , tức *N* <*n* < 2*N* . Gọi A là sự kiện "*n* là hợp số ", và B là sự kiện "thuật toán cho kết quả trả lời *n* là số nguyên tố ". Ta phải tính xác suất ε =*p* (A | B). Theo tính chất b) của tiêu chuẩn Euler-Solovay-Strassen, nếu *n* là hợp số, thì sự kiện

$$\left(\frac{a}{n}\right) \equiv a^{(n-1)/2} \, mod \, n$$

đối với mỗi *a* ngẫu nhiên (1[*a* [*n* - 1) có xác suất [1/2, vì vậy ta có

$$p\left(\frac{B}{A}\right) \leq \frac{1}{2^t}.$$

Theo công thức Bayes ta có

$$p\left(\frac{A}{B}\right) = \frac{p\left(\frac{B}{A}\right).p(A)}{p(B)} = \frac{p\left(\frac{B}{A}\right).p(A)}{p\left(\frac{B}{A}\right).p(A)+p\left(\frac{B}{\bar{A}}\right).p(\bar{A})}.$$

Theo định lý về số nguyên tố, số các số nguyên tố giữa *N* và 2*N* xấp xỉ $\frac{N}{\ln N} \approx \frac{n}{\ln n}$, số các số lẻ là $\frac{N}{2} \approx \frac{n}{2}$, do đó $p(\bar{A}) \approx \frac{2}{\ln n}$, và $p(A) \approx 1 - \frac{2}{\ln n}$. Dĩ nhiên ta có $p(B/\bar{A}) = 1$. Thay các giá trị đó vào công thức trên, ta được

$$p\left(\frac{A}{B}\right) \leq \frac{2^{-t}(1-\frac{2}{\ln n})}{2^{-t}(1-\frac{2}{\ln n})+\frac{2}{\ln n}} = \frac{\ln n-2}{\ln n-2+2^{t+1}}. \qquad (5)$$

Đánh giá đó cũng đúng đối với thuật toán Solovay-Strassen-Lehmann, còn đối với thuật toán Miler-Rabin thì ta được một đánh giá tốt hơn, cụ thể là

$$p\left(\frac{A}{B}\right) = \frac{\ln n-2}{\ln n-2+2^{2t+1}}. \qquad (6)$$

Chú ý rằng khi *t* =50 thì đại lượng ở vế phải của (5) $\approx 10^{-13}$, và vế phải của (6) $\approx 10^{-28}$; do đó nếu chọn cho dữ liệu vào thêm khoảng 50 số ngẫu nhiên a_i thì các thuật toán Euler-Solovay-Strassen và Solovay-Strassen-Lehmann sẽ thử cho ta một số là nguyên tố với xác suất sai lầm [10^{-13} và thuật toán Miller-Rabin với xác suất sai lầm [10^{-28} !

Ta có thể tính được rằng độ phức tạp tính toán về thời gian của các thuật toán xác suất kể trên là vào cỡ đa thức của log*n* , tức là đa thức của độ đài biểu diễn của dữ liệu vào (là số *n*), tuy nhiên các thuật toán đó chỉ cho ta thử tính nguyên tố của một số với một xác suất sai lầm ε nào đó, dù ε là rất bé. Trong nhiều ứng dụng, ta muốn có được những số nguyên tố với độ chắc chắn 100% là số nguyên tố. Do đó, dù đã có các thuật toán xác suất như trên, người ta vẫn không ngừng tìm kiếm những thuật toán tất định để thử tính nguyên tố với độ chính xác tuyệt đối. Trong mấy chục năm gần đây, một số thuật toán đã được đề xuất, trong đó có những thuật toán đặc sắc như thuật toán thử tổng Jacobi, được phát hiện bởi Adleman, Pomerance và Rumely, sau đó được đơn giản hoá bởi Cohen và Lenstra; thuật toán thử bằng đường cong elliptic, được đề xuất bởi Goldwasser, Kilian, Adleman và Huang, được tiếp tục hoàn thiện bởi Atkin và Morain, các thuật toán này đã được dùng để tìm nhiều số nguyên tố rất lớn, thí dụ dùng thuật toán Atkin-Morain đã chứng tỏ được số $(2^{3539}+1)/3$ có 1065 chữ số thập phân là số nguyên tố. Gần đây, vào tháng 8/2002, các nhà toán học Ân độ Agrawal, Kayal và Saxena đã đưa ra một thuật toán tất định mới thử tính nguyên tố có độ phức tạp tính toán thời gian đa thức khá đơn giản, thuật toán đó được mô tả như sau:

<u>Thuật toán Agrawal-Kayal-Saxena:</u>

Input: integer $n > 1$

1. if (n is of the form a^b, $b > 1$) ouput COMPOSITE;
2. r =2;
3. while ($r < n$) {
4. if (gcd(n , r)≠ 1) ouput COMPOSITE;
5. if (r is prime)
6. let q be the largest prime factor of r - 1;

7. if $(q \geq 4\sqrt{r}\log n)$ and $(n^{\frac{r-1}{q}} \neq 1 (\mathrm{mod}\, r))$

8. break;

9. $r \leftarrow r + 1$;

10. }

11. for $a = 1$ to $2\sqrt{r}\log n$

12. if $((x-a)^n \neq (x^n - a) (\mathrm{mod}\, x^r - 1, n))$ ouput COMPOSITE;

13. output PRIME;

Thuật toán này đã được một số nhà toán học kiểm nghiệm , đánh giá cao và xem là một thuật toán đẹp, có thể dùng cho việc kiểm thử tính nguyên tố của các số nguyên.

Trong thực tiễn xây dựng các giải pháp mật mã, thường có nhu cầu có các số nguyên tố rất lớn. Để tìm được các số như vậy, người ta thường chọn ngẫu nhiên một số rất lớn, rồi dùng trước cho nó một thuật toán xác suất chẳng hạn như thuật toán Miller-Rabin; nếu thuật toán cho ta kết quả "là số nguyên tố" với một xác suất sai ε nào đó, thì sau đó ta dùng tiếp một thuật toán tất định (chẳng hạn như thuật toán trên đây) để bảo đảm chắc chắn 100% rằng số đó là số nguyên tố. Thuật toán Agrawal-Kayal-Saxena trên đây được chứng tỏ là có độ phức tạp thời gian đa thức cỡ $O((\log n)^{12})$ khi thử trên số n ; và nếu số nguyên tố được thử có dạng Sophie Germain, tức dạng $2p+1$, thì độ phức tạp thời gian sẽ chỉ là cỡ $O((\log n)^6)$.

2.4.2. Phân tích thành thừa số nguyên tố.

Bài toán phân tích một số nguyên > 1 thành thừa số nguyên tố cũng được xem là một bài toán khó thường được sử dụng trong lý thuyết mật mã. Biết một số n là hợp số thì việc phân tích n thành thừa số mới là có nghĩa; do đó thường khi để giải bài toán phân tích n thành thừa số, ta thử trước n có là hợp số hay không (chẳng hạn bằng một trong các thuật toán ở mục trước); và bài toán phân tích n thành thừa số có thể dẫn về bài toán *tìm một ước số của n,* vì khi biết một ước số d của n thì tiến trình phân tích n được tiếp tục thực hiện bằng cách phân tích d và n/d.

Bài toán phân tích thành thừa số, hay bài toán tìm ước số của một số nguyên cho trước, đã được nghiên cứu nhiều, nhưng cũng chưa có một thuật toán hiệu quả nào để giải nó trong trường hợp tổng quát; do đó người ta có khuynh hướng tìm thuật toán giải nó trong những trường hợp đặc biệt, chẳng hạn khi n có một ước số nguyên tố p với

$p-1$ là ***B*-mịn** với một cận ***B*** >0 nào đó, hoặc khi n là số Blum, tức là số có dạng tích của hai số nguyên tố lớn nào đó ($n = p.q$).

Ta xét trường hợp thứ nhất với (p -1)-thuật tóan Pollard như sau: Một số nguyên n được gọi là ***B*-mịn**, nếu tất cả các ước số nguyên tố của nó đều $\leq B$. ý chính chứa trong (p-1)- thuật toán Pollard là như sau: Giả sử n là B-mịn. Ký hiệu Q là bội chung bé nhất của tất cả các luỹ thừa của các số nguyên tố $\leq B$ mà bản thân chúng $\leq n$. Nếu $q^l \leq n$ thì $l \ln q \leq \ln n$, tức $l \leq \left\lfloor \frac{\ln n}{\ln q} \right\rfloor$ ($\lfloor x \rfloor$là số nguyên bé nhất lớn hơn x). Ta có

$$Q = \prod_{q \leq B} q^{\lfloor \ln n / \ln q \rfloor},$$

trong đó tích lấy theo tất cả các số nguyên tố khác nhau $q \leq B$. Nếu p là một thừa số nguyên tố của n sao cho p -1 là B-mịn, thì p -1 $\mid Q$, và do đó với mọi a bất kỳ thỏa mãn gcd(a,p) = 1, theo định lý Fermat ta có

$a^Q \equiv 1 \pmod{p}$. Vì vậy, nếu lấy d =gcd(a^Q - 1, n) thì $p \mid d$. Nếu $d = n$ thì coi như thuật toán không cho ta điều mong muốn, tuy nhiên điều đó chắc không xẩy ra nếu n có ít nhất hai thừa số nguyên tố khác nhau. Từ những lập luận đó ta có:

(p - 1)-thuật toán Pollard phân tích thành thừa số :

INPUT: một hợp số n không phải là luỹ thừa của một số nguyên tố.

OUTPUT: một thừa số không tầm thường của n .

1. Chọn một cận cho độ mịn B.
2. Chọn ngẫu nhiên một số nguyên a , $2 \leq a \leq n - 1$, và tính d = gcd(a,n). Nếu $d \geq 2$ thì cho ra kết quả (d).
3. Với mỗi số nguyên tố $q \leq B$ thực hiện:

   Tính $l = \left\lfloor \frac{\ln n}{\ln q} \right\rfloor$.

   Tính $a \leftarrow a^{q^l} \bmod n$.
4. Tính d = gcd(a -1, n).
5. Nếu $1 < d < n$ thì cho ra kết quả (d). Nếu ngược lại thì thuật toán coi như không có kết quả.

Thí dụ: Dùng thuật toán cho số n = 19048567. Ta chọn B =19, và a =3, và tính được gcd(3,n) =1. Chuyển sang thực hiện bước 3 ta được bảng sau đây (mỗi hàng ứng với một giá trị của q) :

q	l	a
2	24	2293244
3	15	13555889
5	10	16937223
7	8	15214586
11	6	9685355
13	6	13271154
17	5	11406961
19	5	554506

Sau đó ta tính d =gcd(554506-1,19048567) = 5281. Vậy ta được một thừa số p = 5281, và do đó một thừa số nữa là $q = n/p$ = 3607. Cả hai thừa số đó đều là nguyên tố.

Chú ý rằng ở đây p -1 = 5280 = 2^5.3.5.11 , có tất cả các ước số nguyên tố đều $\leq$ 19, do đó chắc chắn thuật toán sẽ kết thúc có kết quả. Thuật toán sẽ kết thúc không có kết quả khi độ mịn B

được chọn quá bé để không một thừa số nguyên tố *p* nào của *n* mà *p* -1 chỉ chứa các ước số nguyên tố ≤ *B*. Như vậy, có thể xem (*p* -1)-thuật toán Pollard phân tích *n* thành thừa số nguyên tố là có hiệu quả đối với những số nguyên *n* là *B*-mịn, người ta tính được thời gian cần để thực hiện thuật toán đó là cỡ *O*(*B* ln*n* /ln*B*) phép nhân theo môđuyn.

Bây giờ ta xét trường hợp các số nguyên Blum, tức là các số có dạng *n* = *p.q* , tích của hai số nguyên tố lớn. Trước hết ta chú ý rằng nếu ta biết hai số nguyên khác nhau *x* và *y* sao cho $x^2 \equiv y^2 (mod\, n)$ thì ta dễ tìm được một thừa số của *n* . Thực vậy, từ $x^2 \equiv y^2 (mod\, n)$ ta có $x^2 - y^2 = (x+y)(x-y)$ chia hết cho *n* , do *n* không là ước số của *x* + *y* hoặc *x* - *y*, nên gcd(*x* - *y*, *n*) phải là một ước số của *n*, tức bằng *p* hoặc *q* .

Ta biết nếu *n* = *p.q* là số Blum, thì phương trình đồng dư

$$x^2 \equiv a^2 (mod\, n)$$

có 4 nghiệm, hai nghiệm tầm thường là *x* = *a* và *x* = -*a* . Hai nghiệm không tầm thường khác là $\pm b$, chúng là nghiệm của hai hệ phương trình đồng dư bậc nhất sau đây:

$$\begin{cases} x \equiv a(mod\, p) \\ x \equiv -a(mod\, q) \end{cases} \quad \begin{cases} x \equiv -a(mod\, p) \\ x \equiv a(mod\, q) \end{cases}$$

Bằng lập luận như trên, ta thấy rằng nếu *n* là số Blum, *a* là một số nguyên tố với *n*, và ta biết một nghiệm không tầm thường của phương trình $x^2 \equiv a^2 (mod\, n)$, tức biết một $x \neq \pm a$ sao cho $x^2 \equiv a^2 (mod\, n)$ thì gcd(*x* - *a* , *n*) sẽ là một ước số của *n* . Những điều trên đây là căn cứ cho một số phương pháp tìm ước số nguyên tố của một số nguyên dạng Blum; ý chung của các phương pháp đó là dẫn về việc tìm một nghiệm không tầm thường của một phương trình dạng $x^2 \equiv a^2 (mod\, n)$, chẳng hạn như phương trình $x^2 \equiv 1 (mod\, n)$.

Một trường hợp khá lý thú trong lý thuyết mật mã là khi ta biết hai số *a* ,*b* là nghịch đảo của nhau theo modϕ(*n*) (nhưng không biết ϕ(*n*)), và tìm một phân tích thành thừa số của *n.* Bài toán được đặt ra cụ thể là: Biết *n* có dạng B**lum**, biết *a* và *b* sao cho *ab* ≡ 1(modϕ(*n*)). Hãy tìm một ước số nguyên tố của *n* , hay tìm một nghiệm không tầm thường của phương trình $x^2 \equiv 1 (mod\, n)$. Ta giả thiết *ab* - 1 = 2^s. *r* với *r* là số lẻ. Ta phát triển một thuật toán xác suất kiểu Las Vegas như sau: Ta chọn một số ngẫu nhiên *v* (1≤ *v* ≤ *n* - 1). Nếu may mắn *v* là bội số của *p* hay *q*, thì ta được ngay một ước số của *n* là gcd(*v*,*n*). Nếu *v nguyên tố với n* , thì ta tính các bình phương liên tiếp kể từ v^r, được $v^r, v^{2r}, v^{4r}, \ldots$ cho đến khi được $v^{2^t.r} \equiv 1 (mod\, n)$ với một *t* nào đó. Số *t* như vậy bao giờ cũng đạt được, vì có 2^s. *r* ≡ 0 (modϕ(*n*)) nên có $v^{2^s.r} \equiv 1 (mod\, n)$. Như vậy, ta đã tìm được một số $x = v^{2^{t-1}.r}$ sao cho $x^2 \equiv 1 (mod\, n)$. Tất nhiên có *x*≠1 mod*n* . Nếu cũng có *x*≠-1 (mod*n*) thì *x* là nghiệm không tầm thường của $x^2 \equiv 1 (mod\, n)$, từ đó ta có thể tìm ước số

của *n* . Nếu không thì thuật toán coi như thất bại, cho ta kết quả *không đúng.* Người ta có thể ước lượng xác suất cho kết quả *không đúng* với một lần thử với một số *v* là < 1/2, do đó nếu ta thiết kế thuật toán với *m* số ngẫu nhiên $v_1,...v_m$, thì sẽ có thể đạt được xác suất cho kết quả *không đúng* là < $1/2^m$!

2.4.3. Tính logarit rời rạc theo môđuyn nguyên tố.

Cho *p* là một số nguyên tố, và α là một phần tử nguyên thuỷ theo mod*p*, tức là phần tử nguyên thuỷ của nhóm Z_p^*. *Bài toán tính logarit rời rạc* theo mod*p* là bài toán tìm, với mỗi số $\beta \in Z_p^*$,một số *a* ($1 \le a \le p-1$) sao cho $\beta = \alpha^a \bmod p$, tức là $a = \log_\alpha \beta \pmod{p-1}$. Một thuật toán tầm thường để giải bài toán này là thuật toán *duyệt toàn bộ* các số *a* từ 1 đến *p* - 1, cho đến khi tìm được *a* thoả mãn $\beta = \alpha^a \bmod p$. Tất nhiên, thuật toán này là không hiệu quả nếu *p* là số nguyên tố rất lớn. Một biến dạng của thuật toán đó với ít nhiều hiệu quả hơn là <u>*thuật toán Shanks*</u> sau đây:

Đặt $m = \left[\sqrt{p-1}\right]$. Ta tìm *a* dưới dạng $a = mj + i, 0 \le j, i \le m-1$. Rõ ràng $\beta = \alpha^a \bmod p$ khi và chỉ khi $\alpha^{mj} \equiv \beta\alpha^i (mod\ p)$. Ta lập hai danh sách gồm các cặp (j, α^{mj}) và các cặp $(i, \beta\alpha^{-i})$ với *j* và *i* chạy từ 0 đến m - 1. Khi phát hiện ra có hai cặp từ hai danh sách đó có hai phần tử thứ hai bằng nhau là ta được kết quả $a = mj + i$, đó chính là giá trị $\log_\alpha \beta$ mà ta cần tìm. Thuật toán Shanks có độ phức tạp cỡ *O*(*m*) phép toán nhân và *O*(*m*) bộ nhớ (chưa kể *O*(m^2) phép so sánh).

Một thuật toán khác, <u>*thuật toán Polig-Hellman*</u>, thường được dùng có hiệu quả trong trường hợp *p* -1 chỉ có các thừa số nguyên tố bé, có nội dung như sau: Giả thiết rằng *p* - 1 có dạng phân tích chính tắc là

$$p - 1 = \prod_{i=1}^{k} p_i^{c_i}.$$

Để tìm $a = \log_\alpha \beta \pmod{p-1}$, ta tìm các số a_i sao cho $a_i \equiv a\ mod p_i^{c_i}$ với *i* = 1,...,*k.* Sau khi tìm được các a_i như vậy, thì hệ phương trình $x \equiv a_i\ mod p_i^{c_i} \quad (i = 1, \ldots, k)$,được giải theo định lý số dư Trung quốc, sẽ cho ta lời giải $x \equiv a(mod\ p-1)$ cần tìm. Vậy, vấn đề là xác định các $a_i \bmod p_i^{c_i}$ $(i = 1,...,k)$. Vấn đề này được phát biểu lại như sau: Giả sử *q* là một ước số nguyên tố của *p* - 1, và $q^c \mid p-1$ nhưng không còn $q^{c+1} \mid p-1$. Ta cần tìm *x* = *a* modq^c . Ta biểu diễn *x* dưới dạng số *q* - phân như sau:

$$x = \sum_{i=0}^{c-1} x_i q_i \ (0 \le x_i \le q-1).$$

Vì *x* = *a* modq^c nên *a* viết được dưới dạng $a = x + q^c . s$, và vì $a^{p-1} \equiv 1 (\bmod p)$, nên ta có

$$\beta^{\frac{p-1}{q}} \equiv \alpha^{a\frac{p-1}{q}} \equiv (\alpha^{p-1})^{\frac{a}{q}} \equiv \alpha^{\frac{(p-1)x_0}{q}} (mod\ p).$$

Ta đặt $g = a^{(p-1)/q}$, và tính lần lượt $g^0, g^1, g^2, \ldots$, đồng thời so sánh với $b^{(p-1)/q} \bmod p$, ta sẽ tìm được *i* sao cho $g^i \equiv b^{(p-1)/q} \bmod p$. Ta lấy số *i* đó là x_0, tức $x_0 = i$. Nếu *c* = 1 thì *x* = x_0 , ta tìm xong *x* . Nếu *c* >1 thì bằng cách đặt $b' = ba^{-x_0}$ và $x' = \log_a b' \bmod q^c$ ta dễ thấy rằng

$$x' = \sum_{i=1}^{c-1} x_i q_i.$$

Từ đó ta suy ra

$$\beta'^{(p-1)/q^2} \equiv \alpha^{(p-1)x_1/q} \, mod \, p.$$

Tương tự như ở bước trên, tính lần lượt $g^0, g^1, g^2, \ldots$, đồng thời so sánh với $b'^{(p-1)/q^2}$, ta sẽ tìm được x_1.

Cứ làm như vậy, ta sẽ tìm được dần tất cả các giá trị x_i với i = 0,1,...,c -1, tức là tính được x. Sau khi tìm được tất cả các giá trị x ứng với mọi ước số nguyên tố q của p , thì theo một nhận xét ở trên, chỉ cần giải tiếp một hệ phương trình đồng dư bậc nhất theo các môđuyn từng cặp nguyên tố với nhau (bằng phương pháp số dư Trung quốc), ta sẽ tìm được số a cần tìm, $a = \log_\alpha \beta$ theo mod$p$.

Thí dụ: Cho p = 29 và α = 2. Hãy tính $a = \log_2 18$ theo mod29. Ta có $p - 1 = 28 = 2^2 . 7^1$. Theo thuật toán Polig-Hellman, ta tìm lần lượt a mod 4 và a mod 7. Theo các bước tính toán như mô tả ở trên, ta sẽ tìm được a mod 4 = 3 và a mod 7 =4 .Từ đó giải hệ phương trình

$$\begin{cases} x \equiv 3 (mod \, 4), \\ x \equiv 4 (mod \, 7) \end{cases}$$

ta được nghiệm $x \equiv 11$ (mod28), tức được $11 = \log_2 18$ theo mod29. Thuật toán Polig-Hellman cho ta một cách tính logarit rời rạc khá hiệu quả, nhưng chỉ khi p -1 chỉ có các thừa số nguyên tố bé. Vì vậy, nếu p -1 có ít nhất một thừa số nguyên tố lớn thì thuật toán đó khó được thực hiện có hiệu quả, tức trong trường hợp đó bài toán tính logarit rời rạc theo mod$p$ vẫn là một bài toán khó. Một lớp các số nguyên tố p mà p - 1 có ít nhất một ước số nguyên tố lớn là lớp các số nguyên tố dạng $p = 2q + 1$, trong đó q là nguyên tố. Những số nguyên tố dạng đó được gọi là số nguyên tố Sophie Germain, có vai trò quan trọng trong việc xây dựng một lớp khá thông dụng các hệ mật mã có khoá công khai.

Người ta cũng đã nghiên cứu phát triển nhiều thuật toán khác, cả thuật toán tất định, cả thuật toán xác suất, để tính logarit rời rạc, nhưng chưa có thuật toán nào được chứng tỏ là có độ phức tạp tính toán với thời gian đa thức.