$GF(Q)$ 上の対数計算アルゴリズム†

八木沢 正博††

Pohlig と Hellman は，$Q-1$ が小さな素因数のみを持つ場合，$O(\log_2 Q)^2$ の計算量と，$O(\log_2 Q)$ bit の記憶容量を必要とする $GF(Q)$ 上の対数計算アルゴリズムを発表している．このアルゴリズムを改良し，$O(c(\log_2 Q)(\log_2\log_2 Q))$ の計算量と，$O(d(\log_2 Q)(\log_2\log_2 Q))$ bit の記憶容量で対数計算を実行できるアルゴリズムを提案する（c, d は定数）．

1. まえがき

Pohlig と Hellman は Q を素数として，$Q-1$ が小さな素因数のみを持つ場合，位数 Q のガロア体 $GF(Q)$ 上の対数計算が，$O(\log_2 Q)^2$ の計算量と $O(\log_2 Q)$ bit の記憶容量で行えるアルゴリズムを発表している[1]．

本論文では，このアルゴリズムを発展させ，計算量が $O(c(\log_2 Q)(\log_2\log_2 Q))$，記憶容量が $O(d(\log_2 Q)\times(\log_2\log_2 Q))$ bit で与えられるアルゴリズムを提案する．とくに，実用上の点から，$Q-1$ の素因数のべき指数が，2のべき乗-1，（つまり 2^g-1）となる場合を詳細に述べる．

2. Pohlig-Hellman 法

Pohlig と Hellman が発表したアルゴリズム（以下 P. H. 法と呼ぶ）について述べる．

Q は素数とする．$Q-1$ の素因数分解を

$$Q-1=P_1^{n_1}\cdot P_2^{n_2}\cdot\cdots\cdot P_k^{n_k} \quad P_i<P_{i+1} \tag{1}$$

とする．$GF(Q)$ の原始根の一つを α とし，y, α, Q を既知として

$$y\equiv\alpha^x \mod Q \tag{2}$$

を満足する $0\leqq x\leqq Q-2$ を求める．

x は次のように展開される．

$$x_i\equiv x \bmod P_i^{n_i}=\sum_{j=0}^{n_i-1} b_{ij}P_i^j \tag{3}$$

ここで

$$0\leqq b_{ij}\leqq P_i-1 \quad (i=1,2,\cdots,k)$$

である．

x は x_i から中国剰余定理（Chinese remainder theorem）により復元される．

$$x\equiv\sum_{i=1}^{k} c_i x_i \mod Q-1 \tag{4}$$

ここで，c_i は次のように表される．

$$q_i=(Q-1)/P_i^{n_i} \tag{5}$$

$$c_i\equiv d_i q_i \mod Q-1 \tag{6}$$

$$d_i q_i+e_i P_i^{n_i}=1 \tag{7}$$

さて，式(3)の b_{ij} を求めることを考える．

q_i と $P_i^{n_i}$ が互いに素であるとき，式(7)を満足する整数 d_i, e_i が存在することが知られている．

したがって，式(6)，(7)より

$$c_i\equiv d_i q_i\equiv 1 \mod P_i \tag{8}$$

が得られる．

P_i は，$Q-1$ の約数であるから，式(4)より

$$x \bmod P_i\equiv\sum_{j=1}^{k} c_j x_j \mod P_i$$
$$\equiv c_i x_i\equiv x_i \mod P_i\equiv b_{i0} \tag{9}$$
$$(i=1,2,\cdots,k)$$

が成立する．

次に，

$$\gamma_i=\alpha^{(Q-1)/P_i} \tag{10}$$

は，1の P_i 乗根であるから，この γ_i を用いて，

$$y^{(Q-1)/P_i}\equiv\alpha^{(Q-1)\cdot x/P_i}\equiv\gamma_i^{x \bmod P_i}\equiv(\gamma_i)^{b_{i0}}$$
$$\mod Q \tag{11}$$

と表現できる．b_{i0} は式(11)より，y を $(Q-1)/P_i$ 乗した値に，γ_i を $0, 1, 2, \cdots$ 乗した値が一致するまで繰り返すことにより求められる．前もって，γ_i のべき乗のテーブルを作成しておくことにより求めることもできる．

さらに，$n_i\geqq 2$ ならば

$$Z\equiv y\cdot\alpha^{-b_{i0}} \mod Q \tag{12}$$

$$\tilde{x}_{i1}=\sum_{h=1}^{n_i-1} b_{ih}P_i^h \tag{13}$$

と定義し，$i\neq j$ のとき $c_j\equiv 0 \bmod P_i^{n_i}$ に注目して，

$$Z^{(Q-1)/P_i^2}\equiv\alpha^{(Q-1)\tilde{x}_{i1}/P_i^2}\equiv(\gamma_i)^{\tilde{x}_{i1}/P_i}\equiv(\gamma_i)^{b_{i1}}$$

† Algorithms for Computing Logarithms over $GF(Q)$ by MASAHIRO YAGISAWA (Showa Engineering Corporation).

†† 昭和エンジニアリング(株)

$$\bmod Q \tag{14}$$

より，b_{i1} を求めることができる．

以下，同様に $b_{i2}, b_{i3}, \cdots, b_{in_i-1}$ を求めることができる．その手順を**図1**に示す．

$b_{ij}(j=0,1,\cdots,n_i-1)$ を求めるために必要な計算量 M_{PH_i} は，$\bmod Q$ 上の乗算の数として

$$\begin{aligned} M_{PH_i} &\leqq 2\log_2\{(Q-1)/P_i\cdot(Q-1)/P_i^2\cdot\cdots\cdot(Q-1)/P_i^{n_i}\}+2n_i+4(\log_2 P_i)n_i \\ &=2\log_2\{(Q-1)^{n_i}/P_i^{n_i(n_i+1)/2}\}+2n_i \\ &\quad +4n_i(\log_2 P_i) \end{aligned} \tag{15}$$

で与えられるから，すべての b_{ij} を求めるには

$$\begin{aligned} M_{PH} \leqq \sum_{i=1}^{k} &[2\log_2\{(Q-1)^{n_i}/P_i^{n_i(n_i+1)/2}\}+2n_i \\ &+4n_i\log_2 P_i] \end{aligned} \tag{16}$$

の計算量が必要となる．

Q として

$$Q=2^{n_1}\cdot P_2\cdot\cdots\cdot P_k+1 \tag{17}$$

$$P_2=\cdots=P_k\approx 10^2 \tag{18}$$

$$n_1 \gg k \tag{19}$$

の場合には，式(16)より

$$M_{PH}\leqq 2\log_2\{(Q-1)^{n_1}/2^{n_1(n_1+1)/2}\}$$

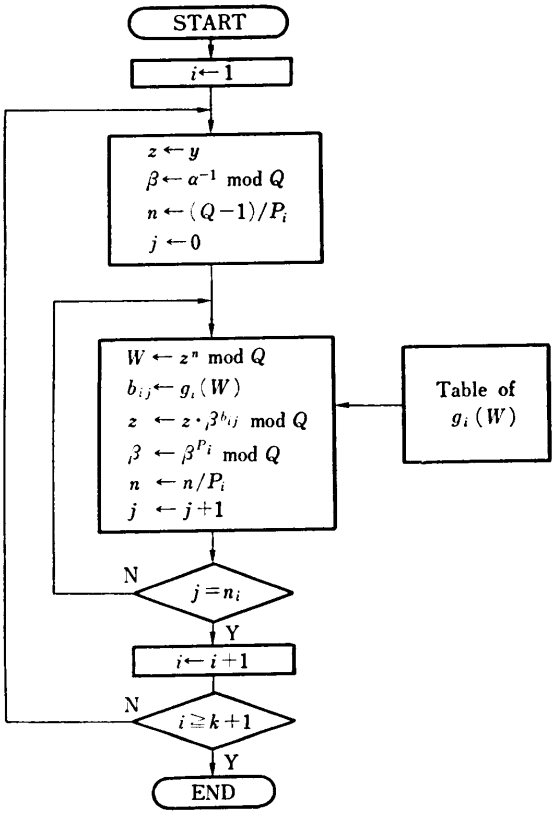

図 1 b_{ij} を求めるアルゴリズム
Fig. 1 Flowchart for algorithm to calculate b_{ij}.

$$\begin{aligned} &+\sum_{i=2}^{k}2\log_2\{(Q-1)/P_i\} \\ &+2n_1+2(k-1)+4n_1+4(k-1)\log_2 10^2 \end{aligned} \tag{20}$$

となるが，

$$\log_2 Q\approx\log_2(Q-1)=n_1+2(k-1)\log_2 10 \tag{21}$$

$$\therefore\ \log_2 Q\approx n_1 \tag{22}$$

式(20)，(22)より

$$M_{PH}=O(\log_2 Q)^2 \tag{23}$$

一般に，$Q-1$ の素因数がすべて小さければ，P_i とほぼ同程度の大きさの実数 t を用いて

$$Q-1\approx t^{\sum_{i=1}^{k} n_i} \tag{24}$$

と表せる．

$$\therefore\ \sum_{i=1}^{k} n_i\approx\log_2 Q/\log_2 t \tag{25}$$

式(16)より

$$\begin{aligned} M_{PH} &\leqq \sum_{i=1}^{k}[2n_i\log_2\{(Q-1)/P_i^{(n_i+1)/2}\} \\ &\quad +2n_i(1+2\log_2 P_i)] \\ &\approx \sum_{i=1}^{k}2n_i\log_2 Q \\ &=(\log_2 Q)\left(2\sum_{i=1}^{k}n_i\right) \\ &\approx(\log_2 Q)(2\log_2 Q/\log_2 t) \\ &\approx O(\log_2 Q)^2 \end{aligned} \tag{26}$$

となる．図1のフローチャートに従い，計算を実行するのに必要なテーブル容量は，$O(\log_2 Q)$ bit である．

b_{ij} と式(3)および中国剰余定理 (Chinese remainder theorem) から，x を求めることができるが，このとき $O(\log_2 Q)$ の計算量が必要となる．

3. 改良型アルゴリズム（1）

P. H. 法の計算手順を

$$Y_i\equiv y^{q_i} \quad \bmod Q \tag{27}$$

$$\delta_i\equiv\alpha^{-q_i} \quad \bmod Q \tag{28}$$

として，書き直す，

説明の便宜上，仮に

$$n_i=2^{g_i}-1 \quad (g_i \text{ は整数}) \tag{29}$$

の場合を考える．（以下，添字 i, $\bmod Q$ を略）

$$\begin{aligned} &\gamma^{b_0}\equiv Y^{P^{n-1}} \\ &\gamma^{b_1}\equiv Y^{P^{n-2}}(\delta^{P^{n-2}})^{b_0} \\ &\gamma^{b_2}\equiv Y^{P^{n-3}}(\delta^{P^{n-3}})^{b_0}(\delta^{P^{n-2}})^{b_1} \\ &\quad\vdots\qquad\vdots\qquad\qquad\ddots \\ &\gamma^{b_{n-1}}\equiv Y^{P^0}(\delta^{P^0})^{b_0}\cdots\qquad\cdots(\delta^{P^{n-2}})^{b_{n-2}} \end{aligned} \tag{30}$$

例えば，式(30)の上から3行目の

$Y^{P^{n-3}}(\delta^{P^{n-3}})^{b_0}$

を P 乗すると，2行目の

$Y^{P^{n-2}}(\delta^{P^{n-2}})^{b_0}$

が得られるから，b_0 が1行目で求まった時点で

$Y^{P^{n-2}}(\delta^{P^{n-2}})^{b_0}$

を一度計算しておけば，2，3行目で，その結果を利用することができる．改良型アルゴリズム(1)では，このように，同じ計算を重複して実行することをできるだけ避けることを主体に計算法を構築している．

以下，計算手順を具体的に示す（図2参照）．

$$y_j \overset{\text{def}}{\equiv} Y^{P^{n-j}} \tag{31}$$

$$[k] \overset{\text{def}}{\equiv} \delta^{P^{n-k}} \tag{32}$$

$$[k,j] \overset{\text{def}}{\equiv} (\delta^{P^{n-k}})^{b_j} \tag{33}$$

と略記する．

(1) $[k]$ テーブルの作成

$k=2,3,4,7,8,\cdots,2^s-1,2^s,\cdots,2^{g-1}-1,2^{g-1}$ に対する $[k]$ の値を格納する．

このテーブルの容量は，$(2g-3)\log_2 Q$ bit である．

(2) γ^b テーブルの作成

$b=1,2,\cdots,P-1$ に対する $\gamma^b \bmod Q$ の値を格納する．

このテーブルの容量は，$(P-1)\log_2 Q$ bit である．

$[k]$, γ^b テーブルは事前に作成しておくことができる．明記していない添字 $i=1,2,\cdots,k$ に対して，両テーブルを作成する必要がある．

次の(3)から(11)までは，明記していない添字 $i=1,2,\cdots,k$ に対して繰り返し実行する．

(3) 与えられた y から y_j $(j=n,n-1,\cdots,2,1)$ を式(27)，(31)から計算し，$j=2^g-1,2^{g-1}-1,\cdots,7,3,1$ に対する y_j の値を y_j テーブルに格納する（$n=2^g-1$ である）．

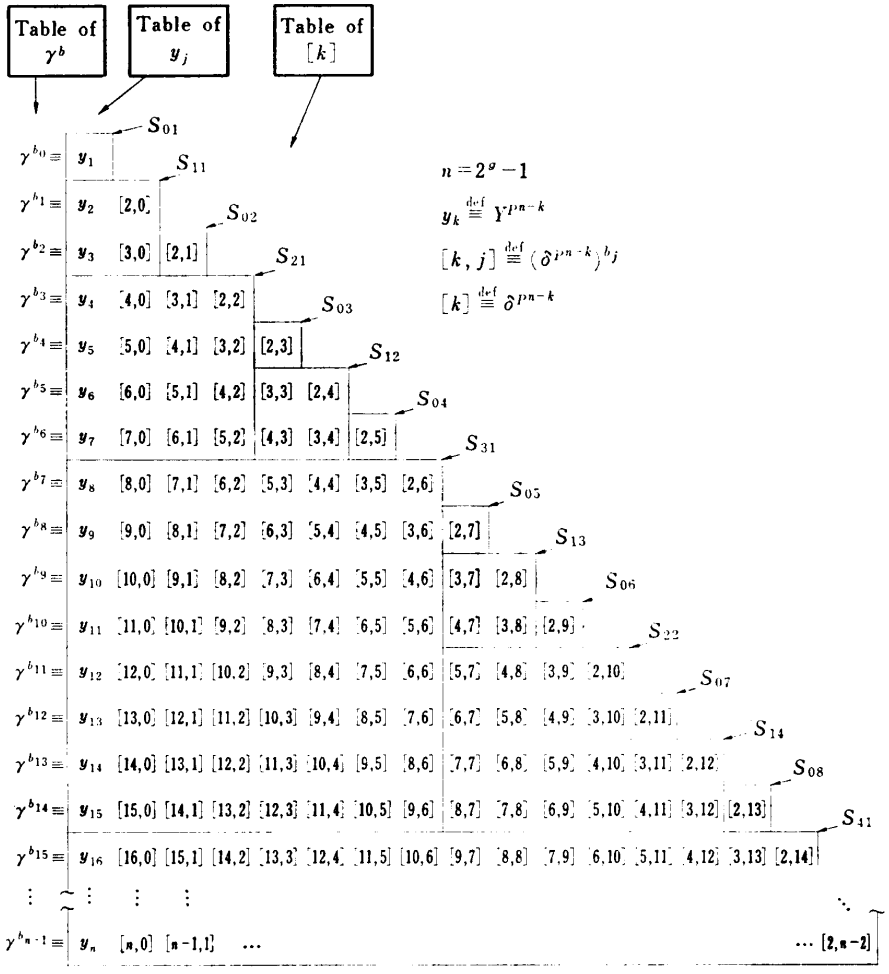

図 2 改良型アルゴリズム(1)の計算ブロック図
Fig. 2 Block-chart for calculating by improved algorithm (1).

このテーブルの容量は $g\log_2 Q$ bit である．

（4） 図2の最上段の式

$$\gamma^{b_0}\equiv y_1$$

に注目する．

y_1 の値は，y_j テーブルより求め，この値と γ^b テーブル上の値が一致する b を求め，b_0 とする．

（5） 図2のブロック S_{11} に注目し

$$a_{11}\overset{\text{def}}{\equiv} y_3[3,0]$$

を計算する．y_j テーブルより，y_3 を求める．[k] テーブルより，[3] を求め，b_0 乗することにより，[3, 0] を求め，a_{11} を計算し，一時記憶する．

$$\gamma^{b_1}\equiv a_{11}{}^P$$

より，右辺を計算し，γ^b テーブルより，b_1 を求める．

（6） 一時記憶した a_{11} と，[k] テーブルより求めた [2] と b_1 より

$$\gamma^{b_2}\equiv a_{11}\cdot[2,1]$$

の右辺を計算し，γ^b テーブルより，b_2 を求める．

（7） y_j テーブルより y_7 を，[k] テーブルより [7] を求め

$$a_{21}\overset{\text{def}}{\equiv} y_7\cdot[7]^{(b_0+P(b_1+Pb_2))}$$

の右辺を計算する．$a_{21}, a_{21}{}^P, a_{21}{}^{P^2}, a_{21}{}^{P^3}$ を計算し，$a_{21}, a_{21}{}^{P^2}$ を一時記憶する．

（8） $\gamma^{b_3}\equiv a_{21}{}^{P^3}$ より，γ^b テーブルから b_3 を求める．

（9） $\gamma^{b_4}\equiv a_{22}{}^{P^2}\cdot[2,3]$

[k] テーブルより，[2] を求め，一時記憶した $a_{21}{}^{P^2}$ と b_3 より右辺を計算し，γ^b テーブルより b_4 を求める．

(10) $a_{12}\overset{\text{def}}{\equiv} a_{21}\cdot[4]^{(b_3+b_4P)}$

一時記憶した a_{21} と [k] テーブルの [4] より，右辺を計算し，a_{12} として一時記憶する．$a_{12}{}^P$ を計算し

$$\gamma^{b_5}\equiv a_{12}{}^P$$

より，γ^b テーブルから，b_5 を求める．

(11) $\gamma^{b_6}\equiv a_{12}\cdot[2,5]$

一時記憶した a_{12} と [k] テーブルの [2] と b_5 より右辺を計算し，γ^b テーブルより，b_6 を求める．

以下，同様にして，逐次 b_j を求めていくことができる．

(12) b_{n-1} まで求められたならば，明記していない添字 $i=i+1$ として，（3）へ戻り，$i=k+1$ となるまで繰り返す．

以上の計算手順に必要な計算量と記憶容量を求める．

ブロック S_{0j} $(j=1,2,\cdots,(n+1)/2)$ 内部の計算量

⋮

ブロック $S_{g-1,1}$ 内部の計算量

が，それぞれほぼ等しい．したがって，明記していない添字 i に対して，必要とする計算量は，

$$\sum_{j=1}^{g} n\left(\frac{1}{2}+2\log_2 P\right)=g\cdot n\cdot\left(\frac{1}{2}+2\log_2 P\right) \quad (34)$$

で与えられる．$i=1,\cdots,k$ に対して必要な計算量 M_Y は

$$\begin{aligned}M_Y&=\sum_{i=1}^{k} g_i n_i\left(\frac{1}{2}+2\log_2 P_i\right)\\&\approx\sum_{i=1}^{k} n_i\log_2 n_i\left(\frac{1}{2}+2\log_2 P_i\right)\\&\leqq\left(\sum_{i=1}^{k} n_i\right)\left(\log_2\left(\sum_{i=1}^{k} n_i\right)\right)\left(\frac{1}{2}+2\log_2 P_{\max}\right)\\&\approx O(c(\log_2 Q)(\log_2\log_2 Q)) \quad (35)\end{aligned}$$

ここで $c=\dfrac{1}{2}+2\log_2 P_{\max}$, $P_{\max}=\max_i P_i$

となる．

$$\left(\because\quad n_1^{n_1}n_2^{n_2}\cdots n_k^{n_k}\leqq\left(\sum_{i=1}^{k} n_i\right)^{n_1}\cdots\left(\sum_{i=1}^{k} n_i\right)^{n_k}=\left(\sum_{i=1}^{k} n_i\right)^{n_1+n_2+\cdots+n_k}\right.$$

両辺の対数をとると

$$\sum_{i=1}^{k}(n_i\log_2 n_i)\leqq\left(\sum_{i=1}^{k} n_i\right)\log_2\left(\sum_{i=1}^{k} n_i\right) \quad (36)$$

となる． $\Big)$.

式(35)は n_i が式(29)のような形をしていない一般の場合にも，あてはまる．

次に，明記していない添字 i に対して必要な記憶容量は，

$$(2g-3+P-1+g)\log_2 Q$$

で与えられる．y_j テーブルは，一時記憶でよいので，$i=1,2,\cdots,k$ に対する必要な記憶容量は，

$$\begin{aligned}&\sum_{i=1}^{k}(2g_i-3+P_i-1)\log_2 Q+\max_i g_i\log_2 Q\\&\approx\sum_{i=1}^{k} 2g_i\log_2 Q+\max_i g_i\log_2 Q\\&\approx O(d(\log_2\log_2 Q)(\log_2 Q))\ \text{bit} \quad (37)\end{aligned}$$

となる（d は定数）．

d の大きさは，$k=1$ の場合は $d\approx 3$

各 g_i が同じ値の場合は $d\approx 2k+1$

となる．

4. 改良型アルゴリズム（2）

改良型アルゴリズム（1）とは独立に，P. H. 法を改良し，計算量を減少させるアルゴリズムを述べる．改

良型アルゴリズム(1), (2)を同時に適用すれば，さらに計算量は減少することになる．

式(30)を次のように変形する（添字 i, mod Q は略）．

$$\begin{aligned}
&Y^{P^{n-4}} && \equiv(\delta^{-P^{n-4}})^{b_0}(\delta^{-P^{n-3}})^{b_1}(\delta^{-P^{n-2}})^{b_2}\gamma^{b_3}\\
&Y^{P^{n-8}}(\delta^{P^{n-8}})^{b_0}(\delta^{P^{n-7}})^{b_1}(\delta^{P^{n-6}})^{b_2}(\delta^{P^{n-5}})^{b_3} && \equiv(\delta^{-P^{n-4}})^{b_4}(\delta^{-P^{n-3}})^{b_5}(\delta^{-P^{n-2}})^{b_6}\gamma^{b_7}\\
&\quad\vdots && \qquad\qquad\vdots\\
&Y^{P^0}(\delta^{P^0})^{b_0} \quad \cdots && (\delta^{P^{n-5}})^{b_{n-5}} \equiv(\delta^{-P^{n-4}})^{b_{n-4}}(\delta^{-P^{n-3}})^{b_{n-3}}(\delta^{-P^{n-2}})^{b_{n-2}}\gamma^{b_{n-1}}
\end{aligned} \tag{38}$$

n が 4 の倍数でないときは，始めの部分を次のようにする（例えば，$n=4M+2$ のとき）．

$$Y^{P^{n-2}}$$
$$\equiv(\delta^{-P^{n-4}})^{0}(\delta^{-P^{n-3}})^{0}(\delta^{-P^{n-2}})^{b_0}\gamma^{b_1}$$
$$Y^{P^{n-4}}(\delta^{P^{n-6}})^{b_0}(\delta^{P^{n-5}})^{b_1}$$
$$\equiv(\delta^{-P^{n-4}})^{b_2}(\delta^{-P^{n-3}})^{b_3}(\delta^{-P^{n-2}})^{b_4}\gamma^{b_5}$$

（以下，同様）

各右辺は指数部の b_j を除けば，同じ形をしているので，$l_0l_1l_2l_3$ に対する

$$\delta(l_0l_1l_2l_3)\overset{\text{def}}{=}(\delta^{-P^{n-4}})^{l_0}(\delta^{-P^{n-3}})^{l_1}(\delta^{-P^{n-2}})^{l_2}\gamma^{l_3} \tag{39}$$

のテーブルを作成しておけば，式(38)の左辺のみを計算することにより，$b_j, b_{j+1}, b_{j+2}, b_{j+3}$ を求めることができる．このとき，テーブルの容量は，

$$P_i{}^4\log_2 Q \text{ bit} \tag{40}$$

となる．

式(38)を模式的に描くと，**図3**のようになる．

〈データ圧縮〉

一般に，式(38)の右辺を $(l+1)$ 項の積とすると，計算量はほぼ $1/l$ に減少するが，必要とするテーブルの容量は $P_i{}^l$ に比例して増大する（P. H. 法に比較して）．

ただし，この場合，Q の代りに R ($R\ll Q$) を用いて例えば

$$\delta(l_0l_1l_2l_3)\bmod Q$$

の代りに

$$(\delta(l_0l_1l_2l_3)\bmod Q)\bmod R \tag{41}$$

をテーブル上に格納すれば，記憶容量を減らすことができる．

しかし，異なる値に，法 R のモード演算 (mod R) を施した結果が，同じ値にならないように注意すべきである．つまり，R 個の中から，ランダムに N_i ($=P_i{}^{l+1}$) 個選んだとき，すべてが異なる値となる確率 P_{P_i} は

$$P_{P_i}=\frac{R(R-1)\cdots(R-N_i+1)}{R^{N_i}} \tag{42}$$

で与えられる．この確率が大きくなるように，R, l を選ぶべきである．

$P_{P_i}>0.9$ となる R, l の例を示す．

$$Q=2^{100}\cdot q+1 \tag{43}$$

$P_1=2$, q は素数で，十分小さい数とする． (44)

$$N_1=2^{l+1}$$
$$P_{P_1}=1\cdot\left(1-\frac{1}{R}\right)\cdots\left(1-\frac{N_1-1}{R}\right)$$
$$\geqq 1-\frac{1}{R}\cdot\frac{N_1(N_1-1)}{2}$$

$l=9$, $R=10^7$ とすると

$$N_1=2^{10},\ P_{P_1}\geqq 1-\frac{1}{10^7}\cdot\frac{2^{10}(2^{10}-1)}{2}$$
$$=0.94 \tag{45}$$

したがって，Q としては $\log_2 Q\approx 100$ bit であるが，各 $\delta(l_0l_1\cdots l_9)$ は，$\log_2 R\approx 24$ bit

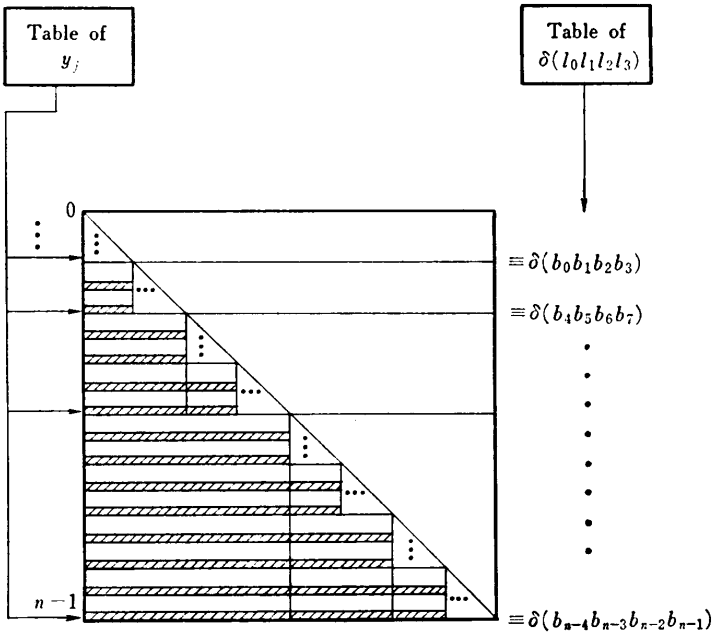

図 3 改良型アルゴリズム(2)の計算ブロック図
Fig. 3 Block-chart for calculating by improved algorithm (2).

で十分である．テーブル容量としては，24 bit×2^{10}=24k bit となる．

5. 数 値 例

$Q=2^8\cdot3+1=769$

原始根 $\alpha=11$, $P_1=2$, $P_2=3$, $n_1=8$, $n_2=1$

理解を助けるため，改良型アルゴリズム(1)，(2)を同時に適用した例を示す．また，式(38)の右辺は2項の積から成る場合を採用した．

$y\equiv11^x \mod 769$

$0\leqq x\leqq768$

与えられた y の値から，上式を満たす x を求める．

x は，次のように x_1, x_2 に分解される．

$x_1\equiv x \mod 2^8$

$x_2\equiv x \mod 3$

$x\equiv-255x_1+256x_2 \mod 768$

と，x は，x_1, x_2 から復元される．

y として

$y=276$

の場合を考える

(1) x_1 を求める．

式(3)より，x_1 は次のように展開される．

$$x_1=\sum_{j=0}^{7}b_{1j}2^j$$

式(27)，(28)より

$Y_1\equiv y^{(Q-1)/P_1^{n_1}}\equiv276^3\equiv116 \mod 769$

$\delta_1\equiv\alpha^{-(Q-1)/P_1^{n_1}}\equiv11^{-3}\equiv26 \mod 769$

式(10)より

$\gamma_1\equiv\alpha^{(Q-1)/P_1}\equiv11^{384}\equiv-1 \mod 769$

式(38)の右辺は2項の積としたので

$\delta(l_0l_1)=(\delta_1^{-P_1^{n_1-1}})^{l_0}\gamma_1^{l_1}$

のテーブルを作成する．この際，式(41)の R として

$R=67$

を採用する．したがって

$\delta(l_0l_1)\equiv\{(26^{-2^6})^{l_0}(-1)^{l_1} \mod 769\} \mod 67$

を用いて，作成する（**表1**参照）．

$y_k\equiv Y_1^{2^{8-k}}$, $[k]\equiv\delta_1^{2^{8-k}}$, $[k,j]\equiv(\delta_1^{2^{8-k}})^{b_{1j}}$

と表記して，式(38)を記述する．

y_2

$\equiv(\delta_1^{-2^6})^{b_{10}}\gamma^{b_{11}}$ ①

$y_4[4,0][3,1]$

$\equiv(\delta_1^{-2^6})^{b_{12}}\gamma^{b_{13}}$ ②

$y_6[6,0][5,1][4,2][3,3]$

$\equiv(\delta_1^{-2^6})^{b_{14}}\gamma^{b_{15}}$ ③

表1 式(38)右辺の値
$\delta(l_0l_1)\equiv\{(\delta_1^{-2^6})^{l_0}(-1)^{l_1} \mod Q\} \mod R$

Table 1 $\delta(l_0l_1)\equiv\{(\delta_1^{-2^6})^{l_0}(-1)^{l_1} \mod Q\} \mod R$, values of rightside of (38).

l_0l_1	$\delta(l_0l_1)$
00	1
10	62
01	31
11	37

表2 $y_k\equiv Y_1^{2^{8-k}} \mod Q$

Table 2 $y_k\equiv Y_1^{2^{8-k}} \mod Q$.

k	y_k
2	707
4	311
8	116

表3 $[k]\equiv\delta_1^{2^{8-k}} \mod Q$

Table 3 $[k]\equiv\delta_1^{2^{8-k}} \mod Q$.

k	$[k]$
4	311
8	26

$y_8[8,0][7,1][6,2][5,3][4,4][3,5]$

$\equiv(\delta_1^{-2^6})^{b_{16}}\gamma^{b_{17}}$ ④

まず，y_8, y_6, y_4, y_2 を計算する（**表2**参照）．

ただし，テーブルに格納するのは，y_8, y_4, y_2 のみである．次に，[4]，[8] を計算し，テーブルに格納する（**表3**参照）．

以上で準備ができた．

まず，式①の左辺より

$y_2=(707 \mod 769) \mod 67\equiv37$

表1より，$l_0l_1=(11)$ $\therefore$ $b_{10}=b_{11}=1$

次に，式②の左辺より

$y_4\cdot[4]\cdot[3]\equiv311\cdot26^{1+2}\equiv(707 \mod 769) \mod 67$

$\equiv37$

表1より，$l_0l_1=(11)$ $\therefore$ $b_{12}=b_{13}=1$

ここで，b_{10}, b_{11}, b_{12}, b_{13} が求まったから，次の a_1 を計算できる．

$a_1\equiv y_8[8][7][6][5]$

$\equiv116\cdot26^{1+2(1+2(1+2))}\equiv27 \mod 769$

式③の左辺より

$y_6[6][5][4][3]\equiv a_1^4\equiv(62 \mod 769) \mod 67\equiv62$

表1より，$l_0l_1=(10)$ $\therefore$ $b_{14}=1$, $b_{15}=0$

式④の左辺より

$y_8[8][7][6][5][4]\equiv a_1\cdot[4]\equiv27\cdot311$

表 4 $A=(\gamma_2^{b_{20}} \bmod Q) \bmod R$
Table 4 $A=(\gamma_2^{b_{20}} \bmod Q) \bmod R$.

b_{20}	A
0	1
1	25
2	6

$\equiv(707 \bmod 769) \bmod 67 \equiv 37$

表1より，$l_0 l_1=(11)$ ∴ $b_{16}=b_{17}=1$

∴ $x_1(11111011)_2=223$

が得られる．

（2） x_2 を求める．

$P_{2^{n_2}}=3^1=3$

であるから，x_2 の展開した形は，

$x_2=b_{20}$

となる．

$\gamma_2 \equiv \alpha^{(Q-1)/P_2} \equiv 11^{256} \equiv 360 \bmod 769$

$\gamma_2^{b_{20}}$ のテーブルを作成する（表4参照）．

これで，x_2 を求める準備ができた．

式(30)より，

$Y_2 \equiv \gamma_2^{b_{20}} \quad \bmod Q$

だから，$y=276$ のとき，x_2 は

$Y_2 \equiv y^{256} \equiv 276^{256} \equiv (408 \bmod 769) \bmod 67 \equiv 6$

表4より，$b_{20}=2$ ∴ $x_2=2$

x_1, x_2 より

$x \equiv -255 \cdot 223 + 256 \cdot 2 \equiv 479 \bmod 768$

が得られる．

6. あとがき

Pohlig と Hellman が提案した $GF(Q)$ 上の対数計算アルゴリズムに改良を加え，計算量が $O(c(\log_2 Q) \times (\log_2 \log_2 Q))$，記憶容量が $O(d(\log_2 Q)(\log_2 \log_2 Q))$ bit となるアルゴリズムを提案した．

さらに，計算量と記憶容量のトレードオフにより，計算量を小さくする方法も示した．実用上は，このトレードオフを利用しても，十分効果が得られると思われる．

計算に必要なデータを Table に格納する際，十分小な数を法としたモード計算を行うことにより，データを圧縮して記憶する方法も提示した．

最後に，簡単な数値例を挙げた．

今後は，このアルゴリズムの応用を考えていきたい．

参考文献

1) Pohlig, S. C. and Hellman, M. E.: An Improved Algorithm for Computing Logarithms over $GF(P)$ and Its Cryptographic Significance, *iEEE Trans. Inf. Theory*, Vol. IT-24, No. 1, pp. 106-110 (1978).

（昭和61年 8 月 13 日受付）
（昭和61年11月 5 日採録）

八木沢正博（正会員）

昭和25年生．昭和49年東京大学工学部計数工学科卒業．昭和51年同大学院修士課程修了．同年昭和電工(株)入社．川崎工場勤務．昭和61年昭和エンジニアリング(株)に出向．現在に至る．化学プラントの計装エンジニアとして，プラントの設計，保全に従事．公開鍵暗号法に興味を持つ．