情報処理学会論文誌 Vol. 50 No. 9 1956–1967 (Sep. 2009)

GF(3^n) 上の関数体篩法の実装実験

林 卓也[†1] 白勢 政明[†1] 高木 剛[†1]

高速実装が可能な実用的ペアリングとして有限体 GF(3^n) 上の超特異曲線を用いた η_T ペアリングが知られている．この η_T ペアリングを利用したペアリング暗号の安全性は GF(3^n) 上の離散対数問題（DLP）の困難性を根拠とする．しかし，GF(3^n) 上の DLP の困難性に関する計算実験の報告は数例のみであり，その困難性の正確な評価を行うにはデータが不十分となっている．本稿では，小さな標数の有限体上の DLP に対する漸近的に最も高速な計算アルゴリズムとして知られている関数体篩法の高速実装を行った．関数体篩法の計算の中で最も計算量を必要とするのは篩処理であるため，GF(3)[x] 上の篩処理について window 法を用いた高速化を行った．その結果，GF(3^n) 上の DLP 計算実験において Granger らが実験を行った従来の世界記録である GF(3^{222})（352 ビット）を大きく上回る GF(3^{277})（440 ビット）上の DLP の計算に成功した．

An Experiment on Implementation of the Function Field Sieve over GF(3^n)

Takuya Hayashi,[†1] Masaaki Shirase[†1] and Tsuyoshi Takagi[†1]

The η_T pairing on supersingular curve over finite field GF(3^n) is known as one of the most efficient pairings. The security of pairing based cryptosystems using the η_T pairing is based on the difficulty of the discrete logarithm problems (DLP) over GF(3^n). The most efficient algorithm for solving the DLP over finite fields of small characteristic is the function field sieve. However, few experiments on the difficulty of the DLP over GF(3^n) have been reported. In this paper, we implemented the function field sieve and experimented the difficulty of the DLP over GF(3^n). We present a faster implementation of the sieving in GF(3)[x] which is the most time-consuming step in the function field sieve. From this improvement, we succeeded solving the DLP over GF(3^{277}) of 440 bits. This is currently the top-record bit-size of the function field sieve over GF(3^n) to the best of our knowledge.

1. はじめに

ペアリング暗号は，ID を公開鍵として利用できる ID ベース暗号[8] など，従来の公開鍵暗号では実現が困難であった様々な暗号プロトコルが提案されている．また，高速な実装が可能なペアリングとして知られている標数 3 の有限体上の超特異曲線を用いた η_T ペアリング[6] は，ソフトウェアやハードウェアなどで多くの実装が発表されている[7),14)]．

標数 3 の有限体上の超特異曲線の η_T ペアリングを利用したペアリング暗号の安全性は，標数 3 の有限体 GF(3^n) 上の離散対数問題（DLP）の困難性，およびその部分体上の楕円曲線上の DLP の困難性を根拠としている．本稿では GF(3^n) 上の DLP の困難性を扱う．この問題の漸近的に最も効率的な解法として関数体篩法[1),24),26)] が知られている．関数体篩法の計算量は鍵長に対して準指数関数時間であり，素因数分解における数体篩法と同等の計算量となっている．しかし，GF(3^n) 上の DLP の計算実験による困難性の解析は素因数分解などと比較してあまり行われていない．実際，2004 年に Granger らは，GF(3^n) 上の DLP 計算の世界記録となる GF(3^{222})（352 ビット）上の DLP の計算に成功した[13)] が，他の有限体上の DLP の計算世界記録[15),18)] や素因数分解の計算世界記録[3),5)] と比較してビット長に大きな差がある．

本稿では，有限体 GF(3^n) に特化した関数体篩法の高速実装およびその計算実験の結果を報告する．関数体篩法において最も計算量が大きい処理が篩処理である．篩処理には多項式篩（Polynomial sieve[12)]）と格子篩[4),22)] の 2 つがあるが，関数体篩法の実装においてこれら 2 つを詳細に比較した報告はない．よって，本稿では多項式篩と格子篩の両方を実装し，従来の世界記録である 350 ビット程度で実験による比較検討を行った．その結果，本実装では多項式篩が高速であったため実験には多項式篩を用いた．また，関数体篩法における篩処理で最も計算時間が必要な処理は，多項式環 GF(3)[x] 上での $k \cdot \mathfrak{p}_i$ ($k \in$ GF(3)[x]) の走査である．従来から知られている手法として文献 12) で提案された GF(2)[x] に対してグレイコードを用いる手法がある．一方，本稿ではグレイコードを用いない手法として window 法の事前計算部を利用する手法を考案し，グレイコードを GF(3)[x] に拡張した手法と比較して 1 割ほどの高速化を達成した．Granger らによる GF(3^n) 上の DLP 計算世界記録の実装と比較すると，約 5 倍ほど高速である．

†1 公立はこだて未来大学大学院システム情報科学研究科
Graduate School of Systems Information Science, Future University Hakodate

以上の結果をふまえて，GF(3^n) 上の DLP の計算世界記録となる GF(3^{277})（440 ビット）上の DLP の計算を行った．関数体篩法の多項式選択は Joux-Lercier の手法[16] を利用し，関係探索ステップにおける篩処理には window 法の事前計算部を利用した多項式篩を実装した．線形代数ステップでは，Structured Gaussian Elimination[19),23)] と Lanczos 法[20)] を実装した．その結果，517.2 時間（約 21 日）により GF(3^{277})（440 ビット）上の DLP の計算に成功した．このビット長は従来の GF(3^n) 上の DLP 計算世界記録である GF(3^{222})（352 ビット）を大きく上回る結果となっている．

2. 有限体上の離散対数問題

本章では，有限体上の離散対数問題およびその解法について説明する．標数 p，拡大次数 n の有限体を GF(p^n) とし，GF(p^n)* を GF(p^n) の乗法群とする．GF(p^n)* の生成元を g とし，$\alpha \in \mathrm{GF}(p^n)^*$ に対し $\alpha = g^\ell$ を満たす $\ell \in \{0, 1, \ldots, p^n - 2\}$ を求める問題を GF(p^n) 上の離散対数問題 (DLP) という．以降，$\ell = \log_g \alpha$ と表記する．

2.1 GF(p^n) 上の DLP 計算実験世界記録

GF(p^n) 上の DLP の鍵長に対する計算実験の世界記録を**表 1** に示す．標数 $p \in \{2, 3, p\}$（p は大きな素数）として記録を分類した．Granger らが行った GF(3^n) 上の DLP 計算実験[13)] はビット長が 352 ビットであり，Joux らによる GF(2^n) 上の DLP 計算実験[15)]（613 ビット）や Kleinjung らによる GF(p) 上の DLP 計算実験[18)]（532 ビット）と比較してビット長に大きな差がある．

2.2 小さな標数の有限体上の DLP の解法

有限体上の DLP の計算アルゴリズムには，準指数関数時間の計算量を持つ指数計算法のアルゴリズムが数多く提案されている．指数計算法に属するアルゴリズムとして数体篩法[11)] や関数体篩法[1),24),26)]，Coppersmith 法[9)] などが知られている．特に小さな標数の有限体上の DLP に関しては関数体篩法や Coppersmith 法が有効である．このアルゴリズムの計算量は，GF(q) ($q = p^n$)，$n \to \infty$ に対し，

$$L_q[1/3, c] = \exp((c + o(1))(\log q)^{1/3}(\log\log q)^{2/3})$$

である．ただし，c は定数，$o(1)$ は $n \to \infty$ に対し $o(1) \to 0$ となる関数である．

Coppersmith 法は 1984 年に Coppersmith によって提案された $p = 2$ に特化したアルゴリズムである．計算量は n によって異なり，最良ケースで $c = (32/9)^{1/3}$，最悪ケースで $c = 4^{1/3}$ となる．Thomé は Coppersmith 法を用いて，2001 年に当時の世界記録となる GF(2^{607}) 上の DLP の計算に成功した[25)]．

表 1　GF(p^n) 上の DLP 計算実験の世界記録
Table 1　The world records of solving the DLP over GF(p^n).

有限体	GF(p)	GF(2^n)	GF(3^n)
計算者/著者	Kleinjung, et al.[18)]	Joux, et al.[15)]	Granger, et al.[13)]
計算日/発表日	2007 年 2 月 5 日	2005 年 9 月 22 日	2004 年 6 月 16 日
アルゴリズム	数体篩法	関数体篩法	関数体篩法
実験環境（関係探索）	Many PCs	Itanium2（1.3 GHz）16×4 台	Pentium4 100 台
実験環境（線形代数）	Xeon64（3.2 GHz）24×8 台	Itanium2（1.3 GHz）16×4 台	Ultra SPARC III 1 台
時間	約 33 日	約 17 日	約 5 日
ビット長	532 ビット	613 ビット	352 ビット

関数体篩法は 1994 年に Adleman によって提案され，$p \le n^{o(\sqrt{n})}$ に対して $c = (64/9)^{1/3}$ の計算量である[1)]．1999 年には Adleman-Huang によってアルゴリズムの詳細が改良され，$p^6 \le n$ に対して $c = (32/9)^{1/3}$ の計算量となった[2)]．この結果は Schirokauer によって一般化され，$p \le n^{o(\sqrt{n})}$ に対し Adleman-Huang の関数体篩法の計算量と同等となった[24)]．また，2002 年に Joux-Lercier によってより効率の良い多項式選択方法を用いた関数体篩法が考案された[16)]．これは Adleman-Huang のものと同様に計算量は $c = (32/9)^{1/3}$ であるが，多項式選択法の違いから実装上では Joux-Lercier の関数体篩法がより高速であることが指摘されている．Granger らは Joux-Lercier の関数体篩法を利用し GF(3^{222}) 上の DLP の計算に成功した[13)]．さらに，2006 年に Joux-Lercier によってある条件を満たす p，n に対して $c = 3^{1/3}$ の計算量となることが示された[17)]．

3. Adleman の関数体篩法

本章では，Adleman の関数体篩法[1)] の概要について説明する．有限体の標数を p（p は小さい素数），拡大次数を n とし，f を n 次モニック既約多項式とする．このとき，GF(p^n) $\cong$ GF(p)[x]/(f) である．GF(p^n)* の生成元を g とする．

$H(x, y)$ を y に関する次数が d，x に関する次数が $d' - 1$ の 2 変数多項式

$$H(x, y) = \sum_{i=0}^{d} \sum_{j=0}^{d'-1} h_{i,j} y^i x^j \in \mathrm{GF}(p)[x, y]$$

とする．また，$H(x, y)$ が次の 8 つの条件を満たすとする．

(1)　H は絶対既約である．
(2)　$h_{d,d'-1} = 1$.
(3)　$h_x = \sum_{i=0}^{d} h_{i,d'-1} y^i$ は $\bar{F}[y]$ 上で平方因子を持たない（$\bar{F}$ は GF(p) の代数的閉包）．
(4)　$h_{0,d'-1} \neq 0$.
(5)　$h_y = \sum_{j=0}^{d'-1} h_{d,j} x^j$ は $\bar{F}[x]$ 上で平方因子を持たない（$\bar{F}$ は GF(p) の代数的閉包）．
(6)　$h_{d,0} \neq 0$.
(7)　$m \in \mathrm{GF}(p)[x]$ $s.t.$ $H(x,m) \equiv 0 \bmod f$ が存在する．
(8)　$\gcd(h_L, (p^n-1)/(p-1)) = 1$ である．ただし，h_L は多項式 $H(x,y)$ で定義される関数体 $L = \mathrm{GF}(p)(x)[y]/(H)$ の類数．

このとき，全射準同形写像

$$\phi : \begin{cases} \mathrm{GF}(p)[x,y]/(H) & \to & \mathrm{GF}(p^n) \cong \mathrm{GF}(p)[x]/(f) \\ y & \mapsto & m \end{cases}$$

が存在する．以降，$H(x,y) = \sum_{i=0}^{d} h_i y^i \in F[x,y]$ $(h_i \in \mathrm{GF}(p)[x])$ と書く．

因子基底の次数の上界である smooth bound B に対し，有理側の因子基底を B_R，代数側の因子基底を B_A とし，次のように定義する．

$B_R = \{\mathfrak{p} \in \mathrm{GF}(p)[x] \mid \deg(\mathfrak{p}) \leq B, \mathfrak{p}$ は既約多項式 $\}$

$B_A = \{\langle \mathfrak{p}, y - t\rangle \in \mathrm{Div}(\mathrm{GF}(p)[x,y]/(H)) \mid \mathfrak{p} \in B_R, t \equiv m \bmod \mathfrak{p}\}$

探索範囲の次数の上界である degree bound D に対し，r，$s \in \mathrm{GF}(p)[x]$ の次数が D 以下で $\gcd(r,s) = 1$ を満たし，かつ，

$$rm + s = \prod_{\mathfrak{p}_i \in B_R} \mathfrak{p}_i^{a_i} \tag{1}$$

$$\langle ry + s\rangle = \sum_{\langle \mathfrak{p}_j, t_j\rangle \in B_A} b_j \langle \mathfrak{p}_j, y - t_j\rangle \tag{2}$$

を満たすとき，r，s の組を double smooth pair という．

今，r，s が double smooth pair であるとする．このとき，$h_L\langle \mathfrak{p}_j, y - t_j\rangle$ は主因子であるため，$h_L\langle \mathfrak{p}_j, y - t_j\rangle = \langle \lambda_j\rangle$ となる $\lambda_j \in (\mathrm{GF}(p)[x,y]/(H))^*$ が一意に存在し，$(ry+s)^{h_L} = \mu \prod \lambda_j^{b_j}$ となる $\mu \in \mathrm{GF}(p)^*$ が存在する．$(ry+s)^{h_L} = \mu \prod \lambda_j^{b_j}$ の両辺に準同形写像 ϕ を適用すると，$(rm+s)^{h_L} = \phi(\mu) \prod \phi(\lambda_j)^{b_j}$ となる．式 (1) より，左辺は，$\prod_{\mathfrak{p}_i \in B_R} \mathfrak{p}_i^{a_i h_L} = \phi(\mu) \prod \phi(\lambda_j)^{b_j}$ となる．$\phi(\mu) \in \mathrm{GF}(p)^*$ より，$(p^n-1)/(p-1)$ を法とする離散対数の関係式は，

$$\sum_{\mathfrak{p}_i \in B_R} a_i h_L \log_g \mathfrak{p}_i \equiv \sum b_j \log_g \phi(\lambda_j) \bmod (p^n-1)/(p-1)$$

である．H の条件 8 より，両辺を $1/h_L$ 倍し $\kappa_j = (\log_g \phi(\lambda_j))/h_L$ とすると，関係式（relation）は，

$$\sum_{\mathfrak{p}_i \in B_R} a_i \log_g \mathfrak{p}_i \equiv \sum b_j \kappa_j \bmod (p^n-1)/(p-1) \tag{3}$$

となる．ここで，$\log_g \mathfrak{p}_i$ および κ_j について式 (3) を解く．各 κ_j と b_j の対応は因子 $\langle \mathfrak{p}_j, y - t_j\rangle$ とその係数 b_j を考えればよい．

十分な個数の double smooth pair を得ることができれば，式 (3) の関係を用いた連立 1 次方程式により $\log_g \mathfrak{p}_i \bmod (p^n-1)/(p-1)$ を計算できる．

4. 関数体篩法の実装要素技術

本章では従来から知られている関数体篩法の高速実装手法について説明する．関数体篩法を以下の 4 つのステップに大別し，各ステップについて説明する．

(i)　多項式選択ステップ（Polynomial selection）
(ii)　関係探索ステップ（Collection of relations）
(iii)　線形代数ステップ（Linear algebra）
(iv)　特定の元の離散対数計算ステップ（Individual logarithms）

4.1 多項式選択ステップ

多項式選択ステップでは $H(x,y)$ が 3 章で述べた 8 つの条件を満たすように構成する．$H(x,y)$ の構成手法は GF(2^n) 上の DLP 計算アルゴリズムである Coppersmith 法の拡張となる Adleman-Huang が提案した手法[2] と，C_{ab} 曲線を用いた Joux-Lercier が提案した手法[16] がある．$H(x,y)$ に C_{ab} 曲線（文献 21) Proposition-Definition 1）を用いると前述した 8 つの条件のうち条件 1–6 を満たす[21]．また，条件 8 については以下の手法により考慮しなくてよい[24]．

標数 p，$H(x,y)$ で定義される関数体 L の種数 g に対して，$h_L \leq (\sqrt{p}+1)^{2g}$ である[24]．(p^n-1) の素因数のうち h_L の上界以下の数の積を N とし，式 (3) を $\bmod (p^n-1)/N$ で解くことで，条件 8 を考慮せずに計算できる．ただし，$\log_g \mathfrak{p}_i \bmod (p^n-1)$ を計算する際に Pohlig-Hellman 法などで GF(p^n)* の位数 N の部分群上の DLP を解く必要がある．

以上のことから，C_{ab} 曲線を用いることで $H(x,y)$ の条件が 8 つの条件から条件 7 のみに緩和される．

4.2 関係探索ステップ

本節で説明する関係探索ステップは素因数分解における数体篩法と類似している．しかし，r，s は自然数ではなく多項式環 GF(p)[x] 上の元であるため，篩処理において関数体篩法に特有の処理が必要となる．

関係探索ステップは double smooth pair となる r，$s \in \mathrm{GF}(p)[x]$ を求め，関係式（式 (3)）を集めるステップである．

$$N_R(r,s) = rm + s,\ N_A(r,s) = N(\langle ry + s \rangle) = r^d H(x, -s/r)$$

とする．また，因子基底 B_R，B_A を変形し，

$B'_R = \{(\mathfrak{p}, t) \mid \deg(\mathfrak{p}) \le B,\ \mathfrak{p}$ は既約多項式，$t \equiv m \bmod \mathfrak{p}\}$

$B'_A = \{(\mathfrak{p}, t) \mid \deg(\mathfrak{p}) \le B,\ \mathfrak{p}$ は既約多項式，$H(x,t) \equiv 0 \bmod \mathfrak{p}\}$

とし，B'_R に属するすべての $\mathfrak{p}$ の集合を P_R，B'_A に属するすべての $\mathfrak{p}$ の集合を P_A とする．また，多項式 $a \in \mathrm{GF}(p)[x]$ のすべての素因子が B 次以下であるとき，a は B-smooth であるとする．このとき，N_R のすべての素因子が P_R に属するとき式 (1) を満たし，N_A のすべての素因子が P_A に属するとき文献 11) の分解方法から式 (2) を満たす．このため，次数が D 以下で互いに素である r，s に対して $N_R(r,s)$ のすべての素因子が P_R に，$N_A(r,s)$ のすべての素因子が P_A に属するか調べることで，divisor の分解をせずに関係式を集めることができる．ここで，N_R が B-smooth であることは N_R のすべての素因子が P_R に属することの必要十分条件であり，また，N_A が B-smooth であることは N_A のすべての素因子が P_A に属することの必要条件であるため，N_R，N_A が B-smooth であるかを調べればよい．

関係探索ステップの単純な方法は，次数が D 以下のすべての r，$s \in \mathrm{GF}(p)[x]$ に対して $N_R(r,s)$，$N_A(r,s)$ が B-smooth となるかを調べることである．しかし，この方法では gcd の計算や既約多項式分解を $p^{D+1} \times p^{D+1}$ 回行う必要があり効率的ではない．そこで，篩処理を行い篩に残ったペアを B-smooth の候補（candidate）とし，候補に対して gcd の計算や既約多項式分解を行うことで数倍程度の速度向上が望める．

以下篩処理について説明する．$N_\bullet(r,s)$ を $N_R(r,s)$ または $N_A(r,s)$ とし，$P_\bullet$ を P_R または P_A とする．式 (1)，(2) を満たす $N_\bullet(r,s)$ は $N_\bullet(r,s) = \prod_{\mathfrak{p}_i \in P_\bullet} \mathfrak{p}_i^{e_i}$ である．ここで $N_\bullet(r,s)$ を割る大きな次数の素因子 $\mathfrak{p}_i$ の指数 e_i は，GF(p)[x] の既約多項式の分布から高い確率で $e_i = 1$ となる．このため，$N_\bullet$ の次数は，

$$\deg(N_\bullet(r,s)) = \sum_{\mathfrak{p}_i \in P_\bullet} e_i \deg(\mathfrak{p}_i) \approx \sum_{\mathfrak{p}_i \in P_\bullet} \deg(\mathfrak{p}_i)$$

と近似することができる．(r,s) に対応する変数 $v(r,s) \in \mathbb{N}$（初期値 0）を用意し，$N_\bullet(r,s)$ を割り切る $\mathfrak{p}_i \in P_\bullet$ が判明した際に $v(r,s)$ に $\deg(\mathfrak{p}_i)$ を加える．この操作をすべての $\mathfrak{p}_i \in P_\bullet$ に対して行い，ある閾値 $t(r,s) \in \mathbb{N}$ を上回る $v(r,s)$ を B-smooth の候補とし，gcd の計算や既約多項式分解を行う．

以上が篩処理の原理である．篩処理の具体的なアルゴリズムには多項式篩や格子篩がある．

4.2.1 多項式篩（Polynomial sieve）

多項式篩は数体篩法における線篩の篩対象を多項式に拡張したアルゴリズムである．多項式篩は $N_\bullet(r,s)$ が $\mathfrak{p}_i$ で割り切れるならば $N_\bullet(r, s + \mathfrak{p}_i)$ もまた $\mathfrak{p}_i$ で割り切れるという性質を用いて，$\mathfrak{p}_i$ で割り切れる $N_\bullet(r,s_0)$ に対し，

$$s = s_0 + \mathfrak{p}_i,\ s_0 + 2\mathfrak{p}_i,\ s_0 + 3\mathfrak{p}_i,\ \ldots$$

のように，s_0 に $\mathfrak{p}_i$ を繰り返し足しこむ操作のみで $\mathfrak{p}_i$ で割り切れる $N_\bullet(r,s)$ を生成できることを利用している．ここで標数 p が小さい素数の場合，$\mathfrak{p}_i$ の足しこみのみでは $\mathfrak{p}_i$ で割り切れる $N_\bullet(r,s)$ を p 個しか生成できない．このため，

$$s = s_0 + \mathfrak{p}_i,\ s_0 + x \cdot \mathfrak{p}_i,\ s_0 + (x+1) \cdot \mathfrak{p}_i,\ \ldots,\ s_0 + 2x \cdot \mathfrak{p}_i,\ \ldots,\ s_0 + x^2 \cdot \mathfrak{p}_i,\ \ldots$$

のように s_0 に $\mathfrak{p}_i$ を足しこむだけでなく，多項式 $k \in \mathrm{GF}(p)[x]$ に対して多項式乗算 $k \cdot \mathfrak{p}_i$ を行う必要があり，すべての $k \cdot \mathfrak{p}_i$ を高速に計算する方法を考える必要がある．以下，この処理を $k \cdot \mathfrak{p}_i$ の走査とよぶ．

Gordon-McCurley は $p = 2$ に対して多項式篩を利用しており，$k \cdot \mathfrak{p}_i$ の走査方法にグレイコードを用いる走査手法を提案している[12]．$G_1, G_2, \ldots, G_{2^d}$ を d 次 2 進グレイコードとする．また，i のビット列で 1 であるビット中の最下位ビットを $l(i)$ と表す．このとき，$i \in \{1, 2, \ldots, 2^d - 1\}$ に対し G_i と G_{i+1} で異なるビットは $l(i)$ である．この性質から，$\sum_{i=1}^{2^d} \mathfrak{p}_i x^{l(i)}$ は各 i の足しこみの過程で $k \cdot \mathfrak{p}_i$ ($k \in \mathrm{GF}(2)[x]$, $\deg(k) \le (d-1)$) をすべて計算することができる．$l(i)$, $\mathfrak{p}_i x^{l(i)}$ の計算は，すべてシフト演算のみで行うことができるため，高速に $k \cdot \mathfrak{p}_i$ の走査を行うことができる．

また，Thomé は $p = 2$ に特化し，複数の篩対象の処理を 1 度に行う Grouping sieves という高速化手法を提案している[25]．

4.2.2 格　子　篩

格子篩[4),22)] は $N_\bullet$ がある既約多項式 $q \in P_\bullet$ で割り切れる (r,s) のみを篩の対象とする．この q に対応する $(q,u) \in B'_\bullet$ を special-q とよぶ．ただし，$B'_\bullet$ は B'_R または B'_A とする．

$N_\bullet$ が special-q で割り切れる格子点 (r,s) は，ある格子基底 v_1，$v_2 \in (\mathrm{GF}(p)[x])^2$ を用いて $av_1 + bv_2$ $(a,b \in \mathrm{GF}(p)[x])$ と表現できる．この ab 平面に対して篩処理を行うことで，$N_\bullet$ が B-smooth という条件が $N_\bullet/q$ が B-smooth という条件に緩和され，より効率的に関係式を集めることができる．ここで，Gauss 縮約を用いて v_1，v_2 を最小基底に変換することで，さらに効率的に関係式を集めることができる．一方で，各 special-q に対して Gauss 縮約の計算やそれにともなう因子基底の変換などの多項式篩にはない計算量の大きい処理が必要になる．

4.3　線形代数ステップ

線形代数ステップでは，関係探索ステップで得た式 (3) を行列計算を用いて解く．今，R 個の関係式を関係探索ステップで集めたとする．このとき，$n_1 = \#B_R$，$n_2 = \#B_A$ とし，$R \times (n_1 + n_2)$ 行列 A，$(n_1 + n_2)$ 次ベクトル $\vec{x}$ を次のように定義する．

$$A = \begin{pmatrix} a_{1,1} & \dots & a_{n_1,1} & b_{1,1} & \dots & b_{n_2,1} \\ \vdots & & \vdots & \vdots & & \vdots \\ a_{1,R} & \dots & a_{n_1,R} & b_{1,R} & \dots & b_{n_2,R} \end{pmatrix}, \vec{x} = \begin{pmatrix} \log_g \mathfrak{p}_1 \\ \vdots \\ \log_g \mathfrak{p}_{n_1} \\ \kappa_1 \\ \vdots \\ \kappa_{n_2} \end{pmatrix}.$$

ただし，$\mathfrak{p}_i \in P_R$ とする．これに対し，

$$A\vec{x} \equiv \vec{0} \bmod (p^n - 1)/(p-1) \tag{4}$$

を解き，ベクトル $\vec{x}$ を求める．この際，行列 A は疎行列となる性質がある．

式 (4) の解法として，Lanczos 法[20] や Wiedemann 法[27] が用いられる．また，前処理として，列ごとの非零濃度の違いを利用して行列のサイズを小さくする Structured Gaussian Elimination [19),23)] が知られている．

4.4　特定の元の離散対数計算ステップ

線形代数ステップで得られた各因子基底の離散対数を用いて求めたい元の離散対数を計算する．特定の元の離散対数計算の手法には，Adleman の手法[1] や Coppersmith の手法[9] が知られている．Adleman の手法は関係探索ステップ，線形代数ステップを繰り返し行う手法であり，計算コストが大きい．このため，実装上では Coppersmith の手法が用いられる．

Coppersmith の手法は以下の手順で行われる．GF(p^n)* の生成元を g とし，$\alpha \in \mathrm{GF}(p^n)^*$ に対し $\log_g \alpha$ を求める．ただし，B 次以下の既約多項式の離散対数 $\log_g \mathfrak{p}_i$（$\mathfrak{p}_i \in P_R$）および式 (3) の κ_j を求めたとする．$\mathfrak{p}_g \in P_R$ を選び，$\gamma \in \{1, 2, \dots, (p^n-1)/(p-1)\}$ をランダムに選ぶ．$z = \mathfrak{p}_g^\gamma \alpha$ を計算し，拡張ユークリッド互除法により $z \equiv z_1/z_2 \bmod f$ となる多項式 z_1，z_2 $(\deg(z_1), \deg(z_2) < n/2)$ を求める．また，パラメータ $B' > B$ に対し z_1，z_2 がともに B'-smooth であるかを調べる．ここで，z_1，z_2 が持つ B 次より大きく B' 次以下の素因子の離散対数を求める．このために，格子篩と同様に special-q を用いる．離散対数を求めたい z_i（$i \in \{1,2\}$）の素因子 d_j（$B < \deg(d_j) \le B'$）を special-q とし，$N_\bullet$ が d_j で割れて，かつ d_j 以外の $N_\bullet$ の素因子 d_k の次数が $\deg(d_j)$ より小さい関係式を探索する．$\deg(d_k) > B$ ならば d_k を special-q とし，再度，関係式を探索する．この操作を special-q 以外の素因子の次数が B 以下になるまで再帰的に実行することで，素因子 d_j の離散対数を計算することができる．

5.　GF(3^n) に特化した関数体篩法の実装

本章では，本稿で扱う GF(3^n) に特化した関数体篩法の実装について詳述する．特に，4.2.1 項で述べた $k \cdot \mathfrak{p}_i$（$k \in \mathrm{GF}(3)[x]$）の走査方法は，GF(3^n) 上の関数体篩法における特徴的な問題であるため，篩処理に重点をおいて説明する．

5.1　パラメータ選択

smooth bound B，degree bound D や，多項式 $H(x,y)$ の変数 y に関する次数 d は，Joux-Lercier の heuristic analysis [16] に基づいて選択した．以下にその式を示す．

$$B = \lceil (4/9)^{1/3} n^{1/3} (\log_3 n)^{2/3} \rfloor,\ D = B,\ d = \left\lfloor \sqrt{n/(B+1)} \right\rfloor \tag{5}$$

また，$H(x,y)$ の多項式の形は Granger らのものを参考に選択した[13]．

5.2　多項式選択法

多項式選択ステップで用いる多項式選択法として，本稿では Joux-Lercier の多項式選択法[16]（**Algorithm 1**）を採用した．

この手法で根 u_1，u_2 および定義多項式 f を構成すると u_1，u_2 の次数はたかだか $\lfloor n/d \rfloor$ となる．一方，Adleman-Huang の手法では，根 m の次数は $\lceil n/d \rceil$ である．このため，d が n を割り切らなければ Joux-Lercier の手法では Adleman-Huang の手法よりも $N_R(r,s)$ の次数が小さくなる．これにより，$N_R(r,s)$ が B-smooth となる確率が上がるため，Adleman-Huang の手法と比較して Joux-Lericer の手法では関係探索ステップの計算量を小さくすることができる．また，Joux-Lercier の手法では $N_R(r,s) = su_2 - ru_1$ となるため，関係式（式 (3)）は以下の式になる．

Algorithm 1　Joux-Lercier の多項式選択法
Algorithm 1　Joux-Lercier's polynomial selection

入力：y に関して次数 d の多項式 $H(x,y)=\sum_{i=0}^{d}h_iy^i\in\mathrm{GF}(3)[x,y]$，拡大次数 n
出力：n 次既約多項式 $f\in\mathrm{GF}(3)[x]$，$u_1,u_2\in\mathrm{GF}(3)[x]$ $s.t.$ $H(x,-u_1/u_2)\equiv 0 \bmod f$

1: **repeat**
2:　たかだか $\lfloor n/d\rfloor$ 次の多項式 $u_1,u_2\in\mathrm{GF}(3)[x]$ をランダムに選ぶ．
3:　$f\leftarrow u_2^dH(x,-u_1/u_2)=\sum_{i=0}^{d}h_i(-u_1)^iu_2^{d-i}$
4: **until** f が n 次既約多項式．
5: **return** f, u_1, u_2

$$\sum_{\mathfrak{p}_i\in P_R}a_i\log_g\mathfrak{p}_i-\log_g u_2\equiv\sum b_j\kappa_j \bmod (3^n-1)/2$$

5.3　篩処理の実装

本節では篩処理の実装方法について詳述する．篩処理ではまず有理側で篩処理を行う．そして，篩に残った候補に対して gcd の計算や有理側および代数側での B-smooth テストを行う．本稿では，有理側でのみ篩処理を行い代数側での篩処理は行っていない．これは，代数側での篩処理では $\mathfrak{p}\in P_A$ が割り切る N_A の計算に d 乗根を計算する必要があり，計算コストが大きいためである．

5.3.1　$k\cdot\mathfrak{p}_i$ の走査方法

関数体篩法において，$k\cdot\mathfrak{p}_i$（$k\in\mathrm{GF}(3)[x]$）の走査は篩処理の計算の大部分を占めるため，この高速化は非常に重要である．

$p=3$ において，4.2.1 項で述べたグレイコードを用いた $k\cdot\mathfrak{p}_i$ の走査方法を考える．$G_1,G_2,\ldots,G_{3^d}$ を d 次 3 進グレイコードとし，$l(i)$ を 3 進展開された i の桁列の中で非零桁となる桁列中の最下位桁とする．このとき，3 進グレイコードにおいても 2 進グレイコードと同様の性質を満たす．すなわち，$i\in\{1,2,\ldots,3^d-1\}$ に対し G_i と G_{i+1} で異なる桁は $l(i)$ である．このことから $p=2$ と同様に，$\sum_{i=1}^{3^d}\mathfrak{p}_ix^{l(i)}$ を計算する際に行われる $\mathfrak{p}_ix^{l(i)}$ の足しこみで $k\cdot\mathfrak{p}_i$（$k\in\mathrm{GF}(3)[x]$）の走査を行うことができる．しかし，$p=2$ と異なり $l(i)$ を求める過程で 3 のべき乗や剰余算を多数行わなければならない．一方，グレイコードを利用しない $k\cdot\mathfrak{p}_i$ の走査方法として window 法の事前計算部を利用する方法を考える．window 法の事前計算部では，計算結果を利用するために元を保存する必要があるが，代入命令 3 回で計算できる負数の計算を多用することで高速化が可能である．

Algorithm 2　GF(3)[x] 上の有理側における window 法を用いた多項式篩
Algorithm 2　Polynomial sieve with window method in rational part over GF(3)[x]

入力：smooth bound $B\in\mathbb{N}$，degree bound $D\in\mathbb{N}$，有理側の因子基底 P_R，$u_1,u_2\in\mathrm{GF}(3)[x]$ $s.t.$ $H(x,-u_1/u_2)\equiv 0 \bmod f$，
出力：候補の集合 $S=\{(r,s)\}$

1: **for all** $r\in\mathrm{GF}(3)[x]$ $s.t.$ $\deg(r)\le D$ **do**
2:　**for** $i=0$ **to** $3^{D+1}-1$ **do**
3:　$v[i]\leftarrow 0$
4:　**for all** $\mathfrak{p}\in P_R$ **do**
5:　$s_0\leftarrow ru_1/u_2 \bmod \mathfrak{p}$
6:　$d\leftarrow D-\deg(\mathfrak{p})$
7:　**if** $d\ge 0$ **then**
8:　$window=\{(k\cdot\mathfrak{p})\}$ $s.t.$ $k\in\mathrm{GF}(3)[x]$, $\deg(k)\le d$ を計算．
9:　**if** $\deg(s_0)\le D$ **then**
10:　**for all** $w\in window$ **do**
11:　$s\leftarrow s_0+w$
12:　$v[\xi(s)]\leftarrow v[\xi(s)]+\deg(\mathfrak{p})$　/* $\xi:\mathrm{GF}(3)[x]\to\mathbb{N}$ */
13:　**for all** $s\in\mathrm{GF}(3)[x]$ $s.t.$ $\deg(s)\le D$ **do**
14:　**if** $v[\xi(s)]\ge t(r,s)$ **then**
15:　$S\leftarrow S\cup\{(r,s)\}$
16: **return** S

グレイコードを利用した走査方法および window 法を利用した走査方法を実装し比較を行った結果，window 法を利用した走査方法が 1 割ほど高速であった．このため，本稿では window 法を用いた走査方法を利用し実験を行った．

5.3.2　多項式篩の実装

window 法を用いた多項式篩のアルゴリズムを **Algorithm 2** に示す．ステップ 8 では，window 幅 d の *window* を window 法の事前計算部を利用して計算する．ステップ 11 では，*window* を用いて s_0 を起点とした走査を行い，$N_R(r,s)$ が $\mathfrak{p}$ で割り切れる s を生成する．ステップ 12 では，配列変数 v に $\deg(\mathfrak{p})$ を足す際，全単射の写像 $\xi:\mathrm{GF}(3)[x]\to\mathbb{N}$, $x\mapsto 3$ を用いて s に対応する自然数を配列 v のインデックスとしている．ステップ 13–15 では，固定した r に対して閾値 $t(r,s)=\deg(N_R(r,s))-B$ を超える $v[\xi(s)]$ に対応する (r,s) を B-smooth の候補として出力する．ここで，$\deg(N_R(r,s))=\max(\deg(ru_1),\deg(su_2))$ と容易に計算できる．

5.3.3　格子篩の実装

格子篩のアルゴリズムを **Algorithm 3** に示す．格子篩は文献 4) を参考に実装した．また，多項式ベクトルに対する Gauss 縮約は文献 16) を参考に実装した．

Algorithm 3　GF(3)[x] 上の有理側における格子篩
Algorithm 3　Lattice sieve in rational part over GF(3)[x]

入力：smooth bound $B \in \mathbb{N}$, degree bound D_a, $D_b \in \mathbb{N}$,
有理側の因子基底 B'_R, special-q の集合 $Q \subseteq (B'_R \cup B'_A)$,
u_1, $u_2 \in \mathrm{GF}(3)[x]$ $s.t.$ $H(x, -u_1/u_2) \equiv 0 \bmod f$
出力：候補の集合 $S = \{(r, s)\}$

1: **for all** $(q, u) \in Q$ **do**
2:　$w_1 \leftarrow (-u, 1)$, $w_2 \leftarrow (q, 0)$
3:　Gauss 縮約で w_1, w_2 の最小基底 $v_1 = (a_1, b_1)$, $v_2 = (a_2, b_2)$ を計算.
4:　**for** $i = 1$ **to** $3^{D_a+1} - 1$ **do**
5:　　**for** $j = 1$ **to** $3^{D_b+1} - 1$ **do**
6:　　$v[i][j] \leftarrow 0$
7:　**for all** $(\mathfrak{p}, t) \in B'_R$ **do**
8:　　$T \leftarrow a_1 + b_1 t$, $U \leftarrow a_2 + b_2 t$
9:　　**if** $\deg(\mathfrak{p}) \le D_a$ **then**
10:　　$aT + bU \equiv 0 \bmod \mathfrak{p}$ とし ab 平面上で多項式篩 (Algorithm 2) を行う.
11:　　**else**
12:　　$w_3 \leftarrow (-T^{-1}U \bmod \mathfrak{p}, 1)$, $w_4 \leftarrow (\mathfrak{p}, 0)$
　　$T \equiv 0 \bmod \mathfrak{p}$ ならば $v_3 \leftarrow (1, 0)$, $v_4 \leftarrow (0, \mathfrak{p})$ とし，ステップ 14 へ.
　　$U \equiv 0 \bmod \mathfrak{p}$ ならば $v_3 \leftarrow (0, 1)$, $v_4 \leftarrow (\mathfrak{p}, 0)$ とし，ステップ 14 へ.
13:　　Gauss 縮約で w_3, w_4 の最小基底 $v_3 = (a_3, b_3)$, $v_4 = (a_4, b_4)$ を計算.
14:　　**for all** c, $d \in \mathrm{GF}(3)[x]$ $s.t.$
　　$\deg(ca_3 + da_4) \le D_a$, $\deg(cb_3 + db_4) \le D_b$ **do**
15:　　$v[\xi(a)][\xi(b)] \leftarrow v[\xi(a)][\xi(b)] + \deg(\mathfrak{p})$　/* $\xi : \mathrm{GF}(3)[x] \to \mathbb{N}$ */
16:　**for all** a, $b \in \mathrm{GF}(3)[x]$ $s.t.$ $\deg(a) \le D_a$, $\deg(b) \le D_b$ **do**
17:　　**if** $v[\xi(a)][\xi(b)] \ge t(a, b)$ **then**
18:　　$S \leftarrow S \cup \{(r, s)\}$ $s.t.$ $r = ab_1 + bb_2$, $s = aa_1 + ba_2$
19: **return** S

格子篩では，$N_\bullet$ が special-q で割れる格子点 (r, s) は格子基底 v_1，v_2 を用いて $(r, s) = av_1 + bv_2$ と表現される．$\deg(\mathfrak{p}) \le D_a$ ($(\mathfrak{p}, t) \in B'_R$) の場合は ab 平面に対して多項式篩を実行するほうが効率が良いことから，ステップ 10 で多項式篩を行う．また，$\deg(\mathfrak{p}) > D_a$ となる $\mathfrak{p}$ に対しては ab 平面上で N_R が $\mathfrak{p}$ で割り切れる格子点を格子基底 v_3，v_4 を用いて求める（ステップ 12–13）．格子基底 v_3，v_4 が作る格子点 $(a, b) = cv_3 + dv_4$ に対応する $N_R(r, s)$ は $\mathfrak{p}$ で割り切れるため，$\deg(a) \le D_a$，$\deg(b) \le D_b$ の点に対して，ステップ 14–15 で二次元配列変数 $v[\xi(a)][\xi(b)]$ に $\deg(\mathfrak{p})$ を足す．この際，多項式篩と同様に全単射の写像 $\xi : \mathrm{GF}(3)[x] \to \mathbb{N}$，$x \mapsto 3$ で a，b に対応するインデックスを計算する．ステップ 17 の閾値 $t(a, b)$ は，$d = \max(\deg(u_1 a\, b_1), \deg(u_1 b\, b_2), \deg(u_2 a\, a_1), \deg(u_2 b\, a_2))$ に対して，$q \in B'_R$ ならば $t(a, b) = d - \deg(q) - B$ で，$q \in B'_A$ ならば $t(a, b) = d - B$ で求める．$t(a, b)$ を超える $v[\xi(a)][\xi(b)]$ は rs 平面に変換し，B-smooth の候補として (r, s) を出力する．

5.3.4　候補に対する処理

篩に残った候補に対しては $\gcd(r, s) = 1$ となるか，$N_\bullet(r, s)$ が B-smooth となるかの計算を行う．

本稿では，拡張ユークリッド互除法を用いて gcd を計算し互いに素か判定を行っている．しかし，小さい次数の多項式は高い確率で r，s を割り切ることから，小さい次数の多項式については r，s に対する被整除性を調べるほうが効率が良い．本稿では，被整除性の判定が容易であることから gcd の計算を行う前に x，$x + 1$，$x + 2$ についてのみ被整除性の判定をしている．これにより，1 割程の高速化を達成した．

$N_\bullet(r, s)$ が B-smooth となるか調べる方法として smooth テスト[9),12)] を用いた．smooth テストは既約多項式分解をせずに B-smooth かどうか判定できるため，直接既約多項式分解を行うよりも高速に判定できる．

$w = N_\bullet(r, s)$ とし，w を関数としたときの微分を w' とする．このとき，$t = w' \prod_{i=\lceil B/2 \rceil}^{B} (x^{3^i} - x) \bmod w$ を計算する．ここで，w' は w の平方因子を含み，$(x^{3^i} - x)$ は i の約数を次数とするすべての既約多項式の積であるため，$t = 0$ であるとき w は高い確率で B-smooth である．既約多項式分解を行う前に smooth テストを行うことで，1 割ほどの高速化を達成した．

5.4　連立一次方程式の解法

本稿では式 (4) の解法に前処理として Structured Gaussian Elimination を，その解法として Lanczos 法を実装した．Lanczos 法は行列とベクトルの積，ベクトルの内積を主演算とした反復法の一種である．行列とベクトルの積は行列が疎行列ならば効率的に演算が可能であるため，疎行列に適したアルゴリズムである．Wiedemann 法との比較としては，

- ともに n 次正方行列に対して計算量が $O(n^2)$ である，
- DLP 計算実験世界記録である文献 15) では Lanczos 法が用いられている一方，同様に世界記録である文献 18) では Wiedemann 法が使われている，

の 2 点から実装上においても有意な差はないようである．

6.　GF(3^n) 上の DLP 計算実験結果

本章では，関数体篩法の計算実験の結果とその考察について述べる．また，GF(3^n) 上の DLP 計算実験世界記録となる GF(3^{277}) 上の DLP 計算結果について詳述する．実装は C

表 2　各篩処理で 10,000 個の関係を得るのに要した時間
Table 2　Timing for finding 10,000 relations using the polynomial or lattice sieve.

篩処理	時間（分）
多項式篩	21.46
格子篩	26.13

言語を用いて行い，コンパイラは gcc，多倍長整数ライブラリは gnu mp [10] を用いた．

1 章で述べた η_T ペアリングを用いると拡大次数 n は 6 の倍数となるが，関数体篩法の計算量は n の約数によらないため，実験ではそのような制約を持たせていない．

6.1　篩処理の選択

DLP 計算の世界記録である文献 13)，15)，18) の篩処理の実装はすべて格子篩を用いている．また，2001 年当時の GF(2^n) 上の DLP 計算記録を達成した文献 25) では格子篩を用いず，多項式篩のみを利用している．しかし，これらの文献にはどちらが実装上高速であるか明示されていない．このため，本実験で使用する篩処理の選択のために，従来の GF(3^n) 上の DLP 計算世界記録と同等の 350 ビット程度で実験的に両方の篩処理を行い，どちらがより高速であるかについて考察する．

GF(3^{223}) 上の DLP 計算の関係探索ステップを実行し，10,000 個の関係を得るのに要する時間を計測した．実験は Quad-Core Xeon E5440（2.83 GHz）× 2 CPUs，16 GB RAM を搭載した計算機 1 台（計 8 コア）で行った．パラメータは式 (5) で求め，smooth bound $B = 13$，多項式は $H(x, y) = y^4 + x^3 + x$ とした．degree bound D は多項式篩では $D = 13$ としたが，格子篩では二次元配列を用いるため RAM の制限により $D_a = D_b = 9$ とした．また，格子篩で用いる special-q は $q \in B'_R$，$\deg(q) = B - 1$ とした．

表 2 に，各篩処理に対する 10,000 個の関係を得るのに要した時間を示す．多項式篩と比較して格子篩が 2 割ほど遅い結果となった．格子篩は $N_\bullet$ が special-q で割れる点のみを篩の対象としているため，多項式篩と比較して篩の対象が少ない．また，$N_\bullet$ が special-q で割れることから，篩対象が B-smooth となる確率は多項式篩より高い．しかし，格子篩では，多項式篩にはない Gauss 縮約や基底変換にともなう処理が必要となる．350 ビット程度では，格子篩特有の処理コストが格子篩の効果を上回ったため格子篩よりも多項式篩が高速となったと考えられる．このため，本稿では篩処理に多項式篩を採用した．

6.2　拡大次数 n に対する GF(3^n) 上の DLP の計算時間の推移

本節では $n = 109$，127，137，157，173，191 について GF(3^n) 上の DLP の計算実験を行い，拡大次数 n の増加に対する関数体篩法の計算時間について考察する．n は GF(3^n)* の位数が大きな素因数を持つように選択した．実験は Quad-Core Xeon E5440（2.83 GHz）× 2 CPUs，16 GB RAM を搭載した計算機 1 台（計 8 コア）で行い，篩処理は 6.1 節の考察から多項式篩を用いた．

表 3　本実験に使用したパラメータ
Table 3　The parameters in our experiments.

拡大次数 n	B, D	$H(x, y) \in \mathrm{GF}(3)[x, y]$
109	10	$y^4 + x$
127	10	$y^4 + x^3 + x$
137	11	$y^4 + x$
157	11	$y^4 + x$
173	12	$y^4 + x$
191	12	$y^4 + x^3 + x$

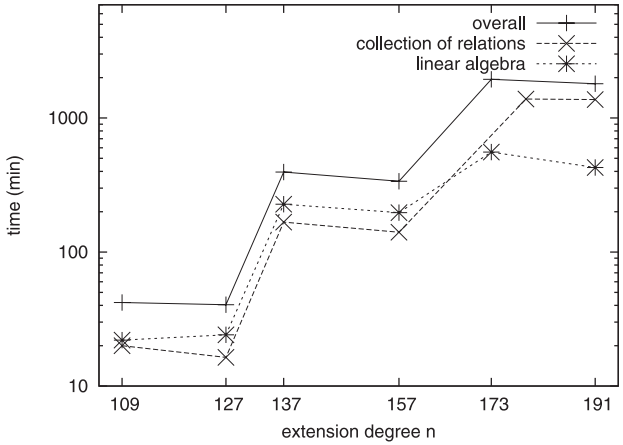

図 1　n に対する GF(3^n) 上の DLP の計算時間
Fig. 1　Timing for solving the DLP over GF(3^n) with different n.

実験に用いるパラメータは式 (5) で計算した．**表 3** に実験に用いた smooth bound B，degree bound D，y に関する次数 d の二変数多項式 $H(x, y)$ を示す．また，**図 1** に各拡大次数 n に対する関係探索ステップ，線形代数ステップおよび関数体篩法全体の計算に要した時間（分：対数スケール）を示す．

式 (5) でパラメータを計算すると $n = 109$，127，137 で $d = 3$ となるが，H が 3 重根を

表 4　関係探索ステップで得られた候補の数および関係の数
Table 4　The number of candidates and relations in the collection of relations.

n	B, D	候補の数	関係の数	因子基底の要素数
109	10	73,348,588	160,398	18,665
127	10	30,383,548	55,859	18,769
137	11	291,623,502	534,362	50,873
157	11	95,780,258	180,537	50,873
173	12	625,637,508	1,327,728	139,253
191	12	270,673,900	486,175	138,989

持つため，$d = 4$ で実験を行った．図 1 において，パラメータ B（$= D$）の増加にともない，関数体篩法全体の計算時間が増加している．これは，degree bound D と関係探索ステップの計算量，smooth bound B と線形代数ステップの計算量の関係に起因する．関係探索ステップにおいて，篩処理を行う篩領域は $3^{D+1} \times 3^{D+1}$ である．よって，degree bound D が増加すると篩領域が 9 倍増加し，篩処理の計算量が約 9 倍になる．また，これにともなって候補の数が増加するため，候補に対する既約多項式分解などの計算量の増加から関係探索ステップの計算量が増加する．線形代数ステップにおいて，扱う行列の列数は因子基底の要素数である．因子基底の要素数（$\#B_R + \#B_A$）は $2 \cdot 3^B/B$ で近似でき，smooth bound B の増加にともない約 3 倍になる．Lanczos 法の計算量は n 次正方行列に対して $O(n^2)$ であるため，smooth bound B の増加とともに Lanczos 法の計算量は約 9 倍になる．

一方で，パラメータ B（$= D$）が同じ拡大次数において，拡大次数が大きいほうが計算時間が短い結果となっている．これは，関係探索ステップで得られた候補の数，関係の数に関係する．**表 4** に各拡大次数 n で得られた候補の数，関係の数および因子基底の要素数を示す．同パラメータにおいて拡大次数 n が大きいと候補の数，関係の数が少ないことが分かる．候補の数が少ないと，gcd の計算や既約多項式分解の回数が少なくなるため，関係探索ステップの計算量が小さくなる．また，関係の数が少ないと行列の行数が少なくなるため，Structured Gaussian Elimination の計算量が小さくなることから線形代数ステップの計算量が小さくなる．

6.3　パラメータの最適性について

表 4 を見ると因子基底の要素数よりも関係探索ステップで得られた関係の数は数倍ほど多い．関係の数が十分にないと線形代数ステップで扱う行列のランクが小さくなってしまい，連立 1 次方程式を解くことができない．しかし，関係の数が過剰だと関係探索ステップで必要以上の計算を行う必要があるため計算量が大きくなる．因子基底の要素数，得られる関係の数はパラメータ B，D によって異なるため，パラメータの最適性について考察を行う必要がある．

表 5　パラメータ B，D に対する GF(3^{137}) 上の DLP 計算時間
Table 5　Timing for solving the DLP over GF(3^{137}) with parameters B, D.

	$B = 9$	$B = 10$	$B = 11$
$D = 9$	No	No	No
$D = 10$	No	44.13	137.6
$D = 11$	42.53	104.3	337.6

単位：（分）

表 5 に GF(3^{137}) 上の DLP 計算時間をパラメータ B，D ごとに計測した結果を示す．実験は Quad-Core Xeon E5440（2.83 GHz）× 2 CPUs，16 GB RAM を搭載した計算機 1 台（計 8 コア）で行った．また，表中 “No” は関係の数が足りなかったため，連立 1 次方程式を解くことができなかったことを指す．

$(B, D) = (11, 11)$ が式 (5) で計算した値であるが，関係の数が因子基底の要素数を上回ったすべてのパラメータにおいて $(B, D) = (11, 11)$ よりも高速であるという結果が得られた．特に，$(B, D) = (9, 11)$ が最も高速であった．このことから，式 (5) で計算した値は最適値と比較して大きいことが分かる．特に smooth bound B に関しては式 (5) で計算した値から 1–2 ほど減じた値，degree bound D に関しては 0–1 ほど減じた値が最適と思われる．

6.4　GF(3^{277}) 上の DLP の計算

本節では，標数 3 の有限体上の DLP 計算世界記録となる，GF(3^{277}) 上の DLP の計算について詳細に記述する．

パラメータは，6.3 節の考察から，式 (5) で計算した値（$B = D = 15$）から調整し，$B = 13$，$D = 14$ とした．また，関数体の定義多項式は $H(x, y) = y^4 + x$ を用いた．乗法群 GF(3^{277})* の位数 $3^{277} - 1$ の非自明な素因数は，以下の 24 ビット，39 ビット，376 ビットの素数である．ただし，`0x` 以降の表記は 16 進数を表す．

$$(3^{277} - 1)/2 = \texttt{0xf11d8f} \times \texttt{0x6f7ae4ee19}$$
$$\times \texttt{0x9fd45e 8d564da1 17d7e227 03db7c4e a9195761}$$
$$\texttt{62396f94 2999f4a4 021595de 9d9f0aca 44d59c9f}$$
$$\texttt{dc8ea9ed 63f7122f}$$

有限体 GF(3^{277}) の定義多項式 f は Algorithm 1 で計算した．以下に f および GF(3^{277}) $\cong$ GF(3)$[x]/(f)$ の生成元 g を示す．ただし，全単射の写像 ξ : GF(3)$[x] \to \mathbb{N}$, $x \mapsto 3$ に対し写像 $\nu = \xi^{-1}$ とし，GF(3)$[x]$ の元を ν を用いて 16 進数で表現する．たとえば，$x = \nu(\texttt{0x3})$,

$x+1=\nu(\texttt{0x4})$，$x+2=\nu(\texttt{0x5})$ となる．

$$f=\nu(\texttt{0xbde83b 323a35f1 2fa25b28 01908fe3 a2d27cc5 51715559 d2d52740 f0d0f588 99667312 62d9d328 5aaba2d1 4675fa4a e1da3f7c 2e23513c})$$

$$g=\nu(\texttt{0x3})$$

離散対数を求める元を $e(x)$，$\pi(x)$ として，以下のように定義する．ネイピア数 $e=2.71\ldots$ の上位 277 桁の各桁の数を上から順に $e_{276}, e_{275}, \ldots, e_0$ とし，$e(x)=\sum_{i=0}^{276}(e_i \bmod 3)x^i$ とする．$\pi(x)$ についても同様とする．

関係探索ステップでは，Quad-Core Xeon E5440（2.83 GHz）×2 CPUs，16 GB RAM を搭載した計算機が 5 台，Quad-Core Xeon X5355（2.66 GHz）×2 CPUs，16 GB RAM を搭載した計算機が 1 台の，計 6 台（計 48 コア）で計算を行った．6.1 節の結果から，篩処理は多項式篩を用いた．次数が 14 次以下のすべての r，s に篩処理を行い，篩空間 $3^{15}\times 3^{15}$ から得られた候補は 402,021,492 個，その中から，523,100 個の関係が得られた．重複する関係を除去した結果，353,448 個となり，必要な個数である 384,533 個を 1 割ほど下回った．このため，次数が 15 次の r，s についても一部篩処理を行い，126,339,538 個の候補が得られ，その中から 60,202 個の関係が得られた．再度，重複する関係の除去を行った結果，関係は全部で 402,697 個となった．関係探索ステップで要した時間は 243.8 時間（約 10 日）だった．

線形代数ステップでは Quad-Core Xeon E5440（2.83 GHz）×2 CPUs，16 GB RAM を搭載した計算機 1 台（計 8 コア）で計算を行った．8 コアで RAM を共有する構成のため，データ転送などによる遅延なしに並列計算を行うことできた．また，行列の元をすべて RAM に保存することができたため，RAM-HDD 間のスワップといった遅延なしに計算を行うことができた．

線形代数ステップは $\bmod((3^{277}-1)/2)$ で計算を行う．前述したように，$((3^{277}-1)/2)$ は 24 ビット，39 ビット，376 ビットの素数の積に分解できるため，中国剰余定理を利用して線形代数ステップの高速化を行うことができる．しかし，線形代数ステップの結果の検証が困難になる点などから，本実験では行っていない．

線形代数ステップで扱う行列は，サイズが 402,697×384,533，非零元の数が平均 17.7 個/行の行列だった．Structured Gaussian Elimination で約 1 時間，前処理を行い，行列のサイズを 317,650×317,650 に削減した．Lanczos 法を 260.6 時間（約 11 日）実行し，因子基底の離散対数を解くことに成功した．以下に因子基底の離散対数の例を 16 進数で示す．

$$\log_g(\nu(\texttt{0x4}))=\texttt{0x6967cd ff11d92c cf944de5 62f05bb0 9a28076f 02230c58 1fb1c26d 7b13ef36 077dde2a e2da59fc e6605502 f69b8db3 b6730947 40aee6f4}$$

$$\log_g(\nu(\texttt{0x5}))=\texttt{0x69aff 1e8f913d 31de275e d3222e9d d310979e 3fbe2d7d 4689d8d4 2d29d27b 612c4c6d d07a8a00 270a4365 ca6c430f fb883197 1631267c}$$

特定の元の離散対数計算ステップは，Quad-Core Xeon E5440（2.83 GHz）×2 CPUs，16 GB RAM を搭載した計算機 1 台（計 8 コア）のみで計算を行った．$e(x)$，$\pi(x)$ に対して，4.4 節で述べた Coppersmith の手法を用いた．以下，Coppersmith の手法の処理については $e(x)$ のみ記述する．約 100 分の計算を行った結果，以下の z_1，$z_2\in \mathrm{GF}(3)[x]$ が得られた．

$$z_1=\nu(\texttt{0x5})\times\nu(\texttt{0x11})\times\nu(\texttt{0x22})\times\nu(\texttt{0x61})\times\nu(\texttt{0x64d8})\times\nu(\texttt{0x906d})\times\nu(\texttt{0xbb586})\times\nu(\texttt{0x879bbc33})\times\nu(\texttt{0x5e f167b4da})\times\nu(\texttt{0x94f 11e8020e})\times\nu(\texttt{0x26b3 5f504b20})$$

$$z_2=\nu(\texttt{0x3})^2\times\nu(\texttt{0x8b})\times\nu(\texttt{0x2950})\times\nu(\texttt{0x1b190c})\times\nu(\texttt{0x448cd1d})\times\nu(\texttt{0x1 154d210e})\times\nu(\texttt{0x3 60cb9a77})\times\nu(\texttt{0x130 7f7519d1})\times\nu(\texttt{0x1f1a 6bef7161})$$

$$\gamma=\texttt{0x21cd41 ca470ab0 8d79e5b2 7eb38e3c b16357c7 83db7e74 3e14b949 4bf9a3a9 9889030e 762e3b9f 800d2832 70bb0dbe c3573f7e 4b1c5cb3}$$

$$s.t.\ x^{\gamma}e(x)\equiv z_1/z_2 \bmod f$$

13 次より大きな次数の素因子を special-q として，それぞれの素因子の離散対数を計算した．各素因子の離散対数の計算に要した時間は平均 1.12 時間だった．以下に $e(x)$，$\pi(x)$ の離散対数を 16 進数で示す．

$$\log_g e(x)=\texttt{0x5748fc 413ea1c6 0cbbcafb e975544a 0d2b9548 4c1ab555 a89275a2 2a31cf65 9a3608bd f3161fcb b243d33a 76cf8e2a e76da8e3 88f5d3fb}$$

$$\log_g \pi(x)=\texttt{0x342b5a d4cede7d 20869605 b2cc2d8c a7ac1cb1 e12ccdb6 6e5e2f38 9120f030 1a3cb850 356ce925 5099bcd1 0dcf8249 13fb9360 86cbcd5d}$$

全体で 517.2 時間（約 21 日）かけて，GF(3^{277})（440 ビット）上の DLP を解くことに

表 6　GF(3^{277}) 上の DLP 計算に要した計算時間の内訳
Table 6　The details of timing for solving the DLP over GF(3^{277}).

関係探索	線形代数	特定の元の離散対数計算	全体
243.8 時間	261.6 時間	11.80 時間	517.2 時間

表 7　Granger らの実験データとの比較
Table 7　Comparison with the experiment by Granger, et al.

	Granger, et al. [13]	本実験のデータ
有限体	GF(3^{222})	GF(3^{277})
実験環境 （関係探索）	Pentium 4 100 台	Quad-Core Xeon 6 台（計 48 コア）
実験環境 （線形代数）	Ultra SPARC III 1 台	Quad-Core Xeon 1 台（計 8 コア）
計算時間	約 5 日	約 21 日

成功した．**表 6** に計算時間の内訳を示す．

表 7 に Granger らの実験環境との比較を示す．計算機のスペックの違いや台数の違いなどあるが，おおむね同規模の計算環境である．次に，実装の比較を行う．Granger らの実装を用いて GF(3^{277}) 上の DLP を解くのに要する時間を簡単に見積もると，$L_{3^{277}}[1/3, (32/9)^{1/3}]/L_{3^{222}}[1/3, (32/9)^{1/3}] \approx 22.8$ であるから，およそ 114 日かかることが分かる．同規模の計算環境であることを考慮すると，我々の実装は Granger らの実装と比較して 5 倍ほど高速である．

7. ま と め

本稿では，有限体 GF(3^n) に特化した関数体篩法の高速実装を行い，GF(3^n) 上の DLP 計算実験を行った．また，関数体篩法の中で最も計算量の大きい篩処理について実装実験による考察を行った．関係探索ステップの篩処理には window 法を利用した多項式篩を実装し，線形代数ステップでは Structured Gaussian Elimination および Lanczos 法を実装した．この実装を用いて，Granger らによる GF(3^n) 上の DLP 計算実験の世界記録である GF(3^{222})（352 ビット）を大幅に超える，GF(3^{277})（440 ビット）上の DLP の計算に成功した．

謝辞　本稿での関数体篩法実装実験の一部は，CRYPTREC（http://www.cryptrec.go.jp/）の支援を受け実施された．また，多くの貴重なコメントをくださいました NTT の青木和麻呂様，富士通研究所の下山武司様，伊豆哲也様に感謝いたします．

参 考 文 献

1) Adleman, L.M.: The Function Field Sieve, *Proc. Algorithmic Number Theory Symposium* (*ANTS-I*), LNCS 877, pp.108–121 (1994).
2) Adleman, L.M. and Huang, M.-D.A.: Function Field Sieve Method for Discrete Logarithms over Finite Fields, *Information and Computation*, Vol.151, pp.5–16 (1999).
3) Aoki, K., Franke, J., Kleinjung, T., Lenstra, A.K. and Osvik, D.A.: A Kilobit Special Number Field Sieve Factorization, *Proc. Advances in Cryptology – ASIACRYPT 2007*, LNCS 4833, pp.1–12 (2007).
4) 青木和麻呂，木田祐司，植田広樹：一般数体篩法実装実験（6）—格子篩，電子情報通信学会技術研究報告，ISEC, 情報セキュリティ，Vol.104, No.315, pp.9–14 (2004).
5) Bahr, F., Boehm, M., Franke, J. and Kleinjung, T.: RSA-200, Announcement (2005).
6) Barreto, P.S.L.M., Galbraith, S., Ó hÉigeartaigh, C. and Scott, M.: Efficient Pairing Computation on Supersingular Abelian Varieties, *Designs, Codes and Cryptography*, Vol.42, No.3, pp.239–271 (2007).
7) Beuchat, J.-L., Brisebarre, N., Detrey, J., Okamoto, E., Shirase, M. and Takagi, T.: Algorithms and Arithmetic Operators for Computing the η_T Pairing in Characteristic Three, *IEEE Trans. Comput.*, Vol.57, No.11, pp.1454–1468 (2008).
8) Boneh, D. and Franklin, M.: Identity Based Encryption from the Weil Pairing, *SIAM Journal on Computing*, Vol.32, No.3, pp.586–615 (2003).
9) Coppersmith, D.: Fast Evaluation of Logarithms in Fields of Characteristic Two, *IEEE Trans. Information Theory*, Vol.IT-30, No.4, pp.587–594 (1984).
10) Free Software Foundation: GNU MP – GNU Multiple Precision Arithmetic Library. http://gmplib.org/
11) Gordon, D.M.: Discrete Logarithms in GF(p) using the Number Field Sieve, *SIAM Journal on Discrete Mathematics*, Vol.6, No.1, pp.124–138 (1993).
12) Gordon, D.M. and McCurley, K.S.: Massively Parallel Computation of Discrete Logarithms, *Proc. Advances in Cryptology – CRYPTO'92*, LNCS 740, pp.312–323 (1992).
13) Granger, R., Holt, A.J., Page, D., Smart, N.P. and Vercauteren, F.: Function Field Sieve in Characteristic Three, *Proc. Algorithmic Number Theory Symposium* (*ANTS-VI*), LNCS 3076, pp.223–234 (2004).
14) Granger, R., Page, D. and Stam, M.: Hardware and Software Normal Basis Arithmetic for Pairing-based Cryptography in Characteristic Three, *IEEE Trans. Comput.*, Vol.54, No.7, pp.852–860 (2005).

15) Joux, A., et al.: Discrete Logarithms in GF(2^{607}) and GF(2^{613}), Posting to the Number Theory List (2005). http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0509&L=nmbrthry&T=0&P=3690
16) Joux, A. and Lercier, R.: The Function Field Sieve is Quite Special, *Proc. Algorithmic Number Theory Symposium* (*ANTS-V*), LNCS 2369, pp.431–445 (2002).
17) Joux, A. and Lercier, R.: The Function Field Sieve in the Medium Prime Case, *Proc. Advances in Cryptology – EUROCRYPT 2006*, LNCS 4004, pp.254–270 (2006).
18) Kleinjung, T., et al.: Discrete Logarithms in GF(p) – 160 digits, Posting to the Number Theory List (2007). http://listserv.nodak.edu/cgi-bin/wa.exe?A2=ind0702&L=nmbrthry&T=0&P=194
19) LaMacchia, B.A. and Odlyzko, A.M.: Solving Large Sparse Linear Systems over Finite Fields, *Proc. Advances in Cryptology – CRYPTO' 90*, LNCS 537, pp.109–133 (1991).
20) Lanczos, C.: Solution of Systems of Linear Equations by Minimized Iterations, *Journal of Research of the National Bureau of Standards*, Vol.49, No.1, pp.33–53 (1952).
21) Matsumoto, R.: Using C_{ab} Curves in the Function Field Sieve, *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences*, Vol.E82-A, No.3, pp.551–552 (1999).
22) Pollard, J.: The Lattice Sieve, *The Development of the Number Field Sieve*, pp.43–49 (1991).
23) Pomerance, C. and Smith, J.W.: Reduction of Huge, Sparse Matrices over Finite Fields via Created Catastrophes, *Experimental Mathematics*, Vol.1, No.2, pp.89–94 (1992).
24) Schirokauer, O.: The Special Function Field Sieve, *SIAM Journal on Discrete Mathematics*, Vol.16, No.1, pp.81–98 (2003).
25) Thomé, E.: Computation of Discrete Logarithms in $\mathbb{F}_{2607}$, *Proc. Advances in Cryptology – ASIACRYPT 2001*, LNCS 2248, pp.107–124 (2001).
26) 内山成憲：有限体上の離散対数問題—数体ふるい法，関数体ふるい法，日本応用数理学会論文誌，Vol.13, No.2, pp.245–256 (2003).
27) Wiedemann, D.H.: Solving Sparse Linear Equations over Finite Fields, *IEEE Trans. Information Theory*, Vol.IT-32, No.1, pp.54–62 (1986).

(平成 20 年 12 月 1 日受付)
(平成 21 年 6 月 4 日採録)

林　卓也（学生会員）

2008 年公立はこだて未来大学システム情報科学部卒業．現在，同大学大学院システム情報科学研究科博士前期（修士）課程在学中．公開鍵暗号の安全性解析に関する研究に従事．電子情報通信学会学生会員．

白勢　政明（正会員）

1994 年茨城大学理学部数学科卒業．1996 年同大学大学院理学研究科修了．同年より JTB 出版にて時刻表システムの開発に従事．2006 年北陸先端科学技術大学院大学情報科学研究科博士課程修了．博士（情報科学）．同年より公立はこだて未来大学ポスドク，2008 年より同大学助教，現在に至る．公開鍵暗号の高速実装および情報セキュリティに関する研究に従事．電子情報通信学会会員．

高木　剛（正会員）

1993 年名古屋大学理学部数学科卒業．1995 年同大学大学院理学研究科修士課程修了．同年 NTT 情報流通プラットフォーム研究所入社．2001 年理学博士（ダルムシュタット工科大学）．その後，ダルムシュタット工科大学情報科学部助教授を経て，2005 年公立はこだて未来大学システム情報科学部准教授，2008 年より同大学教授，現在に至る．2009 年より九州大学大学院数理学研究院客員教授．第 8 回船井情報科学振興賞受賞．暗号および情報セキュリティに関する研究に従事．電子情報通信学会，IACR 各会員．