SCIS 2014 The 31st Symposium on
Cryptography and Information Security
Kagoshima, Japan, Jan. 21 - 24, 2014
The Institute of Electronics,
Information and Communication Engineers

既約因子の次数に制限がある多項式の因数分解について

Factoring Polynomials Composed of Restricted Degree Irreducible Factors

小崎 俊二 * 松尾 和人 †
Shunji Kozaki Kazuto Matsuo

あらまし 有限体上の多項式の因数分解は暗号技術に対して様々な応用が知られている。特に超楕円曲線暗号に対しては、その構成法や攻撃の中で必須なアルゴリズムの一つである。効率的な因数分解アルゴリズムとして知られる Cantor-Zassenhaus アルゴリズムは多項式の既約因子の最高次数が小さいほど高速に動作する。応用においても多項式が可約であり既約因子の最高次数が既知である場合もある。例えば、種数 2 の超楕円曲線の等分多項式の既約因子の最高次数は等分多項式の次数の 3/4 乗程度になることが知られている。したがって、Cantor-Zassenhaus アルゴリズムような既約因子の最高次数が小さいとき高効率なアルゴリズムを利用することで、等分多項式の因数分解が高速化され安全な超楕円曲線暗号の構成が効率化される。Cantor-Zassenhaus アルゴリズムには、より効率的な改良がいくつか知られているが、これらは既約因子の最高次数がその効率に影響を与えない。そこで本論文では、Cantor-Zassenhaus アルゴリズムの改良アルゴリズムを既約因子の最高次数が与えられた場合に拡張し、既約因子の最高次数が小さいときに一層効率化するアルゴリズムを構成した。また、その計算量を評価し、拡張アルゴリズムが効果を持つ場合が多いことを示した。さらに、拡張アルゴリズムが現実的な問題に対して効率的に動作することを実装実験により示した。

キーワード 多項式の因数分解, Cantor-Zassenhaus アルゴリズム, Kaltofen-Shoup アルゴリズム, 超楕円曲線暗号, 等分多項式, 位数計算

1 はじめに

有限体上の（1 変数）多項式の因数分解は暗号技術に対して様々な応用が知られている。特に、超楕円曲線暗号に対しては、その構成手法や攻撃の中で必須なアルゴリズムの一つである。

有限体上の多項式の因数分解の確率的多項式時間アルゴリズムは Barlekamp [Ber67, Ber70] によって提案され、後に Barlekamp とは別のアルゴリズムが Cantor と Zassenhaus [Zas69, CZ81] によって提案された（Cantor-Zassenhaus アルゴリズム）。その後も両アルゴリズムの改良が行われ、現在のところ Cantor-Zassenhaus アルゴリズムの改良として、von zur Gathen と Shoup のアルゴリズム（Gathen-Shoup アルゴリズム）[GS92]、Kaltofen と Shoup のアルゴリズム（Kaltofen-Shoup アルゴリズム）[KS97, Sho95]、Kedlaya と Umans のアルゴリズム（Kedlaya-Umans アルゴリズム）[KU11] 等がよく知られている。

Cantor-Zassenhaus アルゴリズムは、Barlekamp のアルゴリズムと違い、多項式の既約因子の最高次数が小さいほど高速に動作するが、Kaltofen-Shoup アルゴリズム等の改良アルゴリズムではこの性質が失われている。一方、実際の応用においては、既約因子の最高次数が既知であり且つ被分解多項式の次数より大幅に小さい場合もある。例えば、種数 2 の超楕円曲線の等分多項式の既約因子の最高次数は等分多項式の次数の 3/4 乗程度になることが知られている [IM10]。この性質を利用し等分多項式の因数分解を高速化できれば種数 2 の超楕円曲線の位数計算が高速化され、安全な超楕円曲線暗号を効率的に構成可能である。このような応用例において、改良アルゴリズムは Cantor-Zassenhaus アルゴリズムが本来有していた好ましい性質を利用できないこととなる。

そこで、本論文では Cantor-Zassenhaus アルゴリズムの改良アルゴリズムを既約因子の最高次数が与えられた場合に拡張し、既約因子の最高次数が小さいときに一層効率化するアルゴリズムを構成する。また、その計算量を評価するとともに実装実験を行い、拡張アルゴリズムの有効性を検証する。

* 情報セキュリティ大学院大学, 神奈川県横浜市神奈川区鶴屋町 2-14-1, Institute of Information Security, 2-14-1, Tsuruya-cho, Kanagawa-ku, Yokohama-city, Kanagawa 221-0835, JAPAN
† 神奈川大学理学部情報科学科, 神奈川県平塚市土屋 2946, Faculty of Science, Kanagawa Univ., 2946, Tsuchiya, Hiratsuka-shi, Kanagawa 259-1293, Japan.

Algorithm 1 Cantor-Zassenhaus の DDF

Input: $f \in \mathbb{F}_q[X]$, monic, squarefree
Output: $\{g_k \mid 1 \le k \le t\}$ s.t.
$f = \prod_{1\le k\le t} g_k$, $g_k = \prod_{1\le j\le r_k} g_{k,j}$,
$g_{k,j} \in \mathbb{F}_q[X]$: irreducible/$\mathbb{F}_q$, $\deg g_{k,j} = k$, $t \le \deg f$

```
h ← X mod f, k ← 0
while f ≠ 1 do
    h ← h^q mod f, k ← k + 1
    g ← gcd(f, h − X)
    if g ≠ 1 then
        L ← L ∪ {(g, k)}
        f ← f/g, h ← h mod f
return L
```

本論文では、まず第 2 節で多項式の因数分解を定義し、Cantor-Zassenhaus アルゴリズムとその改良アルゴリズムを紹介する。次に第 3 節で既約因子の最高次数が与えられた因子次数制限付き因数分解を定義し、この問題に Cantor-Zassenhaus アルゴリズムとその改良アルゴリズムを適用した場合の計算量について議論する。そして、第 4 節で Cantor-Zassenhaus アルゴリズムの改良アルゴリズムを因子次数制限付き因数分解に拡張し、その計算量を評価する。さらに、第 5 節で拡張アルゴリズムの実装実験による評価を行い、第 6 節でまとめる。

本論文では $M(n)$ で n 次多項式の乗算の $\mathbb{F}_q$ 上の計算量、n^ω で $n \times n$ 行列の行列乗算の $\mathbb{F}_q$ 上の計算量を表す。詳細評価では、多項式乗算に FFT 乗算を仮定し、$M(n) = n^{1+o(1)}$ とする。また、行列乗算は、Coppersmith と Winograd[CW90] が示した $\omega = 2.376$、Coppersmith と Winograd より現実的な Strassen [Str69] の行列乗算アルゴリズムを利用した場合の $\omega = 2.808$、標準的な行列乗算を利用した場合の $\omega = 3$ に対して個別に評価する。

2 多項式の因数分解と Cantor-Zassenhaus アルゴリズム

本節では、多項式の因数分解と因数分解アルゴリズムについて、以降で必要となる知識をまとめる。

2.1 多項式の因数分解

q を奇素数の冪とする。位数 q の有限体 $\mathbb{F}_q$ 上のモニック多項式 $f \in \mathbb{F}_q[X]$ の因数分解は、

$$f = \prod_{1\le i\le r} f_i^{e_i}$$

を満足する組 $(f_i, e_i) \in \mathbb{F}_q[X] \times \mathbb{N}$ の集合 $\{(f_i, e_i) \mid 0 \le i \le r\}$ を求める計算である。ここで、f_i は $\mathbb{F}_q$ 上既約なモニック多項式である。

以下では多項式の因数分解の確率的多項式時間アルゴリズムである Cantor-Zassenhaus アルゴリズムと既存の改良アルゴリズムについてまとめる。

2.2 Cantor-Zassenhaus アルゴリズム

Cantor-Zassenhaus アルゴリズムは

- Squarefree decomposition (SFD)
- Distinct-degree factorization (DDF)
- Equal-degree factorization (EDF)

の 3 アルゴリズムから構成される。以下で各々の概略と既知の改良を紹介する。

2.2.1 Squarefree decomposition

Cantor-Zassenhaus アルゴリズムでは最初に被分解多項式に対して squarefree decomposition (SFD) を適用する。SFD は多項式を squarefree 多項式の積に分解する処理である。ここで、squarefree 多項式とは 1 次以上の任意の多項式の平方で割り切れない多項式のことである。即ち、SFD はモニック多項式 $f \in \mathbb{F}_q[X]$ に対して、$f = \prod_{1\le i\le u} g_i^{s_i}$ を満足する $\{(g_i, s_i) \in \mathbb{F}_q[x] \times \mathbb{N} \mid 1 \le i \le u\}$ を求める処理である。ここで g_i はモニックな squarefree 多項式であり、$i \ne j$ のとき $\gcd(g_i, g_j) = 1$ である。SFD の計算量は、$n = \deg f$ としたとき、$O(M(n)\log n) = O(n^{1+o(1)})$ である [Yun76]。

2.2.2 Distinct-degree factorization

Distinct-degree factorization (DDF) は SFD によって得られたモニック squarefree 多項式 $f \in \mathbb{F}_q$ に対し、f の k 次モニック既約因子の積 g_k の集合

$$\{g_k \in \mathbb{F}_q[X] \mid 1 \le k \le t \le \deg f\}$$

を求める処理である。即ち、$\{g_{k,j} \in \mathbb{F}_q[X] \mid 1 \le j \le r_k\}$ を f の k 次モニック既約因子の集合としたとき、

$$f = \prod_{1\le k\le t} g_k, \quad g_k = \prod_{1\le j\le r_k} g_{k,j}$$

を満足する g_k を $1 \le k \le t \le \deg f$ に対してを求める処理である。

以下では Cantor-Zassenhaus アルゴリズム及び各改良アルゴリズムの DDF を紹介する。

Cantor-Zassenhaus アルゴリズム Cantor-Zassenhaus アルゴリズム [CZ81] では

$$X^{q^k} - X \tag{1}$$

が k 次以下の $\mathbb{F}_q$ 上既約なモニック多項式全ての積であることを利用して、DDF を効率的に実現している。即ち $\gcd(f, X^{q^k} - X)$ が f の k 次以下のモニック既約因子の積であることから、f の k 次モニック既約因子の積 g_k は

$$g_k = \gcd(f, X^{q^k} - X) / \gcd(f, X^{q^{k-1}} - X)$$

で与えられる。この式にしたがって、$k = 1, 2, \ldots$ に対して g_k を逐次的に求めることで効率的なアルゴリズムが得られる。Cantor-Zassenhaus の DDF アルゴリズムを、文献 [GG13, ,Algorithm 14.3] にしたがい Algorithm 1 に示す。

Algorithm 1 において、$k = 1, \ldots, t = O(\deg f)$ に対して行う f を法とする式 (1) の計算がアルゴリズムを律速し、その計算量は、$n = \deg f$ としたとき、$O(nM(n)\log q) = O(n^{2+o(1)}\log q)$ である。

Algorithm 1 に関していくつかの改良アルゴリズムが知られているが、どれも式 (1) に現れる、$k = 1, \ldots, t \le n$ に対する $X^{q^k} \mod f$ の逐次計算をブロック化することで高速化を図っている。以下では改良アルゴリズムの概略を紹介する。

Gathen-Shoup アルゴリズム Gathen-Shoup アルゴリズム [GS92] では、Moenck と Borodin [MB72] の multipoint evaluation アルゴリズムを利用して $X^{q^k} \mod f$ の計算を効率化している。今、$\gamma_i(X) \equiv X^{q^{2^i}} \mod f$ とすると、$\gamma_0 \equiv X^q \bmod f$ は繰り返し 2 乗法によって $O(M(n)\log q)$ で計算可能である。また、$X^q, \ldots, X^{q^{2^i}}$ から式 $X^{q^{2^i+j}} \equiv \gamma_i(X^{q^j}) \bmod f,\ j = 1, \ldots, 2^i$ によって、$X^q, \ldots, X^{q^{2^{i+1}}}$ を計算可能である。この計算は multipoint evaluation アルゴリズムにより $O(M(n)^2 \log 2^i)$ で実現可能である。したがって、$k = 1, \ldots, t \le n$ に対する式 (1) の計算が $O(M(n)^2 \log n \log t)$ で可能である。Gathen-Shoup アルゴリズムではさらに改良が加えられ、この計算を

$$O((n/t)M(nt)\log t) \tag{2}$$

で実現している。さらに、通常は $t = n$ として計算を行うため、計算量は $O(M(n^2)\log n) = O(n^{2+o(1)})$ となる。Multipoint evaluation アルゴリズムの詳細については [GG13, Section 10.1] 等を参照されたい。

Kaltofen-Shoup アルゴリズム Kaltofen-Shoup アルゴリズム [KS97, Sho95] では式 (1) の代わりに

$$X^{q^{\ell j}} - X^{q^i} \tag{3}$$

を利用する。ここで、$\ell j > i$ のとき $X^{q^{\ell j}} - X^{q^i} = (X^{q^{\ell j - i}} - X)^{q^i}$ より、式 (3) の多項式は $\ell j - i$ 次以下の $\mathbb{F}_q$ 上既約なモニック多項式の全てを因子とする。そこで、$\ell \le n$, $m = \lceil n/\ell \rceil$ と置き、$i = 0, \ldots, \ell - 1$, $j = 1, \ldots, m$ に対して $\gcd(X^{q^{\ell j}} - X^{q^i}, f)$ を計算すれば、$k = 1, \ldots, n$ に対して、k 次以下のモニック既約多項式である f の因子の積によって構成された多項式が得られ、DDF アルゴリズムを構成可能である。Kaltofen-Shoup アルゴリズムでは、baby-step giant-step 戦略により、式 (3) の計算において $i = 0, \ldots, \ell - 1$ に対する $X^{q^i} \mod f$ の計算と $j = 1, \ldots, m$ に対する $X^{q^{\ell j}} \mod f$ の計算を個別に行う。$i = 0, \ldots, \ell - 1$ に対する $X^{q^i} \mod f$ の計算には繰り返し 2 乗法を利用し、その計算量は、

$$O(\ell M(n)\log q) \tag{4}$$

である。一方、$j = 1, \ldots, m$ に対する $X^{q^{\ell j}} \mod f$ の計算には Brent と Kung [BK78] の modular composition アルゴリズム（Brent-Kung アルゴリズム）を利用する。

Algorithm 2 Kaltofen-Shoup の DDF [KS97]

Input: $f \in \mathbb{F}_q[X]$, monic, squarefree
Output: $\{g_k \mid 1 \le k \le t\}$ s.t.
$f = \prod_{1 \le k \le t} g_k$, $g_k = \prod_{1 \le j \le r_k} g_{k,j}$,
$g_{k,j} \in \mathbb{F}_q[X]$: irreducible/$\mathbb{F}_q$, $\deg g_{k,j} = k$

1: $\ell \leftarrow \lceil n^\beta \rceil$, $m \leftarrow \lceil n/(2\ell) \rceil$
2: **for** $i = 0, \ldots, \ell$ **do**
3: $\quad h_i \leftarrow X^{q^i} \bmod f$
4: Compute $H_j \leftarrow X^{q^{\ell j}} \bmod f$ for $j = 1, \ldots, m$ by the modular composition algorithm
5: Compute $I_j \leftarrow \prod_{1 < i \le \ell} H_j - h_i$ for $j = 1, \ldots, m$ by using the multipoint evaluation algorithm.
6: $f^* \leftarrow f$
7: **for** $j = 1, \ldots, m$ **do**
8: $\quad F_j \leftarrow \gcd(f^*, I_j)$
9: $\quad f^* \leftarrow f^*/F_j$
10: **for** $j = 1, \ldots, m$ **do**
11: $\quad g \leftarrow F_j$
12: $\quad$ **for** $i = \ell - 1, \ldots, 0$ **do**
13: $\quad\quad g^* \leftarrow \gcd(g, H_j - h_i)$
14: $\quad\quad L \leftarrow L \cup \{(g^*, \ell j - i)\}$
15: $\quad\quad g \leftarrow g/g^*$
16: **if** $f^* \neq 1$ **then**
17: $\quad L \leftarrow L \cup \{(f^*, \deg f^*)\}$
18: **return** L

Modular composition アルゴリズムは、$\deg g, \deg h < \deg f$ を満足する多項式 $f, g, h \in \mathbb{F}_q[X]$ に対して、$g(h) \bmod f$ を計算するアルゴリズムである。$n = \deg f$ としたとき、Brent-Kung アルゴリズムの計算量は $O(n^{(\omega+1)/2})$ であり、これを $i = 1, \ldots, m$ に対する $X^{q^{\ell j}} \bmod f$ の計算に適用すれば、$O(mn^{(\omega+1)/2})$ で計算可能である。Kaltofen-Shoup アルゴリズムではこの計算をさらに改良し、$i = 1, \ldots, m$ に対する $X^{q^{\ell j}} \bmod f$ の計算を

$$O(m^{(\omega-1)/2} n^{(\omega+1)/2}) \tag{5}$$

で実現している。

Kaltofen-Shoup アルゴリズムは式 (3) を利用する他にも multipoint evaluation アルゴリズム等のブロック計算手法を用いて詳細に効率化が図られている。Algorithm 2 に、文献 [KS97] にしたがって Kaltofen-Shoup の DDF アルゴリズムを示す。Algorithm 2 に現れる β は Step 2-3 と Step 4 の計算量のバランスをとり最適化するためのパラメータであり、$0 < \beta < 1$ に値をとる。

Algorithm 2 において $n_0 = \ell m$ と置くと、Step 15 までに n_0 次以下の DDF は終了しており、残次数 $n - n_0$ が n_0 以下であれば、Step 15 を終了した時点で f^* は 1 であるか $\deg f^* > n_0$ の既約多項式である。したがって、これらの場合に関して Step 16, 17 の処理を行うことで、DDF を完了できる。以上の議論から m は $n/2 \le n_0 = \ell m$ を満足すれば十分であり、Step 1 において、$m = \lceil n/\ell \rceil$ の代わりに $m = \lceil n/(2\ell) \rceil$ と設定されている。

Algorithm 2 の各ステップの計算量は Step 2-3 が $O(\ell M(n)\log q)$, Step 4 が $O(m^{(\omega-1)/2} n^{(\omega+1)/2})$ である。

また、Step 5 以降の計算量は

$$O(M(\ell)M(n)\log\ell + M(m)M(n)\log m) \tag{6}$$

であり、Step 2-4 がアルゴリズムを律速する [KS97]。以上より、Algorithm 2 の計算量は

$$O(\ell M(n)\log q) + O(m^{(\omega-1)/2}n^{(\omega+1)/2}) = O(n^{\beta+1+o(1)}\log q + n^{(1-\beta)(\omega-1)/2+(\omega+1)/2}) \tag{7}$$

である。ここで、$\beta = 2(\omega-1)/(\omega+1)$ とすると、式 (7) の 2 つの項の n の指数が等しくなり、$\log q$ を固定した場合の計算量の最適値が得られる。したがって、$\omega = 2.376$ のとき $\beta = 0.815$ であり Algorithm 2 の計算量は $O(n^{1.815}\log q)$ となる。また、$\omega = 2.808$ のとき $\beta = 0.815$ であり $O(n^{1.950}\log q)$ となる。さらに、$\omega = 3$ とすると $\beta = 1$ となり Cantor-Zassenhaus の DDF と同一の計算量となる。

NTL アルゴリズム　文献 [KS97, Sho95] では、行列乗算を利用しない modular composition アルゴリズムと、それを利用した「実用的な」DDF アルゴリズムも提案されている。このアルゴリズムは Shoup の NTL ライブラリ [Sho90] に実装されている。そこで以下ではこのアルゴリズムを NTL アルゴリズムと呼ぶ。NTL アルゴリズムでは、modular composition を

$$O(n^2 + n^{1/2}M(n)) \tag{8}$$

で実現する。また、$j = 1,\ldots,m$ に対する $X^{q^{\ell j}} \bmod f$ の計算のみならず、$i = 1,\ldots,\ell-1$ に対する $X^{q^i} \bmod f$ の計算にも繰り返し 2 乗法の代わりに modular composition を利用している [1]。したがって、アルゴリズムは $\beta = 0.5$ のときに最適化され、計算量は

$$O(n^{1/2}(n^2 + \sqrt{n}M(n)) + M(n)\log q) = O(n^{2.5} + n^{1+o(1)}\log q) \tag{9}$$

となる。

NTL アルゴリズムでは、Kaltofen-Shoup アルゴリズムと異なり、Step 5 の計算に multipoint evaluation を利用せず通常の剰余乗算を利用している。したがって、Step 5 以降の計算量は

$$O(\ell m M(n)) = O(n^{2+o(1)}) \tag{10}$$

となるが、Kaltofen-Shoup アルゴリズムと同様に、これは Step 2-4 の計算量で抑えられ、NTL アルゴリズムの DDF の計算量は式 (9) で与えられる。

Kedlaya-Umans アルゴリズム　Kedlaya と Umans は、$O(n^{1+o(1)})$ の modular composition アルゴリズムを提案し、これを Kaltofen-Shoup アルゴリズムに適用した [KU11]。Kedlaya-Umans アルゴリズムでは、NTL アルゴリズムと同様に $\beta = 0.5$ として、$j = 1,\ldots,m$ に対する $X^{q^{\ell j}} \bmod f$ の計算のみならず、$i = 1,\ldots,\ell-1$ に対する $X^{q^i} \bmod f$ の計算にも modular composition アルゴリズムを利用している。また、Step 5 以降は Kaltofen-Shoup アルゴリズムと同一である。したがって、Kedlaya-Umans アルゴリズムにおいても、Step 2-4 が律速であり、DDF が

$$O(n^{1/2}n^{1+o(1)} + n^{1+o(1)}\log q) = O(n^{1.5+o(1)} + n^{1+o(1)}\log q) \tag{11}$$

で実現される。

Kedlaya-Umans の DDF アルゴリズムは、現在のところ最良計算量のアルゴリズムであるが、現在の計算機で扱える問題に対しては効率的ではないことが知られている [BZ09]。

2.2.3 Equal-degree factorization

Equal-degree factorization (EDF) は、DDF によって得られた k 次モニック既約多項式の積 $g_k \in \mathbb{F}_q[X]$ を k 次既約多項式に分解する処理である。即ち、$\deg g_{k,j} = k$ を満足する既約多項式 $g_{k,j}$ の積

$$g_k = \prod_{1\le j\le r_k} g_{k,j}$$

に対して、多項式集合 $\{g_{k,j} \mid 1 \le i \le r_k\}$ を求める処理である。

Cantor-Zassenhaus アルゴリズムでは EDF に確率的多項式時間アルゴリズムが用いられている。Cantor-Zassenhaus アルゴリズムの EDF の計算量は、$n_k = \deg g_k = kr_k$ として、$O((k\log q + \log n_k)M(n_k)\log r_k)$ である。

EDF は $k = 1\ldots,t \le \deg f$ に対して実行されるので計算量の合計を見積ると、$n = \deg f$ に対して $\sum_{0\le k\le t} n_k = n$ であることに注意して、

$$O(\sum_{0\le k\le t}((k\log q + \log n_k)M(n_k)\log r_k)) = O(nM(n)\log q) = O(n^{2+o(1)}\log q) \tag{12}$$

を得る [GG13, Theorem 14.14]。

DDF と同様に、EDF に対しても多くの改良が知られている。von zur Gathen と Shoup の改良アルゴリズムは Brent-Kung の modular composition アルゴリズムを

[1] $i = 1$ に対しては繰り返し 2 乗法を利用する。

表 1: Cantor-Zassenhaus アルゴリズムの計算量 (†: $\omega = 2.376$, ‡: $\omega = 2.808$, ††: $\omega = 3$)

Square-free decomposition	
Yun [Yun76]	$O(n^{1+o(1)})$
Distinct-degree factorization	
Cantor-Zassenhaus [CZ81]	$O(n^{2+o(1)} \log q)$
Gathen-Shoup [GS92]	$O(n^{2+o(1)})$
Kaltofen-Shoup [KS97]	$O(n^{1.815} \log q)$ †
	$O(n^{1.950} \log q)$ ‡
NTL [Sho95]	$O(n^{2.5} + n^{1+o(1)} \log q)$
Kedlaya-Umans [KU11]	$O(n^{1.5+o(1)} + n^{1+o(1)} \log q)$
Equal-degree factorization	
Cantor-Zassenhaus [CZ81]	$O(n^{2+o(1)} \log q)$
Gathen-Shoup [GS92]	$O(n^{1.189+o(1)} + n^{1+o(1)} \log q)$ †
	$O(n^{1.404+o(1)} + n^{1+o(1)} \log q)$ ‡
	$O(n^{1.5+o(1)} + n^{1+o(1)} \log q)$ ††
NTL [Sho95]	$O(n^2 + n^{1+o(1)} \log q)$
Kedlaya-Umans [KU11]	$O(n^{1+o(1)} \log q)$

利用して、$k = 1 \ldots, t \leq \deg f$ に対する合計計算量

$$O(\sum_{0 \leq k \leq t} (n_k^{(\omega+1)/2+o(1)} \log(n_k/r_k) + n_k^{1+o(1)} \log n_k/k \log q)) = O(n^{(\omega+1)/2+o(1)} + n^{1+o(1)} \log q)) \quad (13)$$

を実現している。これは、$\omega = 2.376$ のとき $O(n^{1.189+o(1)} + n^{1+o(1)} \log q)$ であり、$\omega = 2.808$ のとき $O(n^{1.404+o(1)} + n^{1+o(1)} \log q)$ となる。また、$\omega = 3$ のとき $O(n^{1.5+o(1)} + n^{1+o(1)} \log q)$ となる。さらに、modular composition に「現実的な」アルゴリズムを利用した NTL アルゴリズムでは EDF の計算量が $O(n^2 + n^{1+o(1)} \log q)$ であり、Kedlaya と Umans の改良アルゴリズムでは $O(n^{1+o(1)} \log q)$ である。

2.2.4 Cantor-Zassenhaus アルゴリズムの計算量

以上で示した Cantor-Zassenhaus アルゴリズムとその改良アルゴリズムの計算量を表 1 にまとめる。

表 1 から明らかなように、Cantor-Zassenhaus アルゴリズムにおいて DDF が律速であり、最良の Kedlaya-Umans アルゴリズムで $O(n^{1.5+o(1)}+n^{1+o(1)} \log q)$、現実的な選択である Strassen の行列乗算を利用した Kaltofen-Shoup アルゴリズムで $O(n^{1.950} \log q)$、通常の利用で最も実用的であると考えられる NTL アルゴリズムで $O(n^{2.5} + n^{1+o(1)} \log q)$ である。

3 因子次数制限付き因数分解

本節では、既約因子の最高次数が既知の場合の因数分解問題を与え、前節で紹介した Cantor-Zassenhaus アルゴリズムとその改良アルゴリズムをこの問題に適用した場合の計算量について議論する。

3.1 因子次数制限付き因数分解

因子次数制限付き因数分解は、次数 n のモニック多項式 $f \in \mathbb{F}_q[X]$ と $0 < \alpha \leq 1$ が与えられ、f の既約因子の次数が n^α 以下であるとき、即ち f が n^α 次以下の既約多項式 f_i, $i = 1, \ldots, r$ の積

$$f = \prod_{1 \leq i \leq r} f_i^{e_i}, \ \deg f_i \leq n^\alpha \quad (14)$$

であるときに、$(f_i, e_i) \in \mathbb{F}_q[X] \times \mathbb{N}$ の集合

$$\{(f_i, e_i) \mid 0 \leq i \leq r\} \quad (15)$$

を求める処理である。

SFD は因子次数制限付き因数分解に対して計算量が変化しないので、以降では議論の単純化のため、f が square-free であるとし、SFD を介さずに DDF アルゴリズムに入力されると仮定する。

3.2 Cantor-Zassenhaus アルゴリズムの適用

ここでは、Cantor-Zassenhaus とその改良アルゴリズムを因子次数制限付き因数分解に適用した場合の計算量を DDF と EDF 各々について個別に評価する。

3.2.1 DDF

Algorithm 1 に示した Cantor-Zassenhaus の DDF アルゴリズムでは、既約因子の最高次数が n^α のとき Step 2–7 の while 文の繰り返し回数が n^α 以下になるので、計算量が $O(n^{\alpha+1+o(1)} \log q)$ に削減される。一方、改良アルゴリズムでは、計算をブロック化して効率化するために入力多項式の次数 n に依存したパラメータ設定が行われる。したがって、既約因子の最高次数が n より小さい場合にも計算量は削減されない。例えば、Algorithm 2 に示した Kaltofen-Shoup アルゴリズムでは、n から導かれた ℓ, m が以降の繰り返し回数を定めるため、既約因子の最高次数によって計算量は変化しない。

3.2.2 EDF

Cantor-Zassenhaus アルゴリズムにおける EDF では、$k = 1, \ldots, t \leq n$ に対してアルゴリズムが適用されるが、式 (12) に示したように合計の計算量は既約因子の最高次数に影響されない。同様に、Gathen-Shoup の改良アルゴリズムにおいても、式 (13) に示したように合計の計算量は既約因子の最高次数に影響されない。

4 因子次数制限付き因数分解への DDF の拡張

前節で述べたように、既存アルゴリズムを因子次数制限付き因数分解問題に適用した場合、Cantor-Zassenhaus の DDF アルゴリズム以外は通常の因数分解と同一の計算量が必要である。また、Cantor-Zassenhaus アルゴリズムにおいては DDF と EDF の計算量が同一のため、DDF が高速化された場合 EDF が計算量の主項となり、実際

には大幅な高速化が期待されるものの漸近計算量は削減されない。

SFD はその性質から既約因子の次数よって計算量が変化しないことを前節で述べ、EDF も既約因子の次数に制限がある場合も通常と同一の計算量が必要であることを前節で示した。一方、Cantor-Zassenhaus アルゴリズムのみならずその改良アルゴリズムにおいても、DDF は既約因子の次数に制限がある場合に対して効率的な拡張が可能であると考えられる。また、Cantor-Zassenhaus アルゴリズムと異なり、改良アルゴリズムは DDF が律速であるため、DDF の計算量削減がアルゴリズム全体の漸近的計算量削減につながると期待される。そこで以下では、Gathen-Shoup アルゴリズム、Kaltofen-Shoup アルゴリズム、NTL アルゴリズム、Kedlaya-Umans アルゴリズムの DDF を因子次数制限付き因数分解に拡張する。また、拡張した DDF の計算量を評価する。

4.1 DDF の拡張

4.1.1 Gathen-Shoup アルゴリズムの拡張

Gathen-Shoup の DDF では、$t = n^\alpha$ とし、$k = 1, \ldots, t \le n$ に対して式 (1) を計算することで、因子次数制限付き因数分解に対応可能である。しかし、式 (2) より、このときの計算量は

$$\begin{aligned} O((n/t)M(nt)\log t) &= O(n^{1-\alpha}M(n^{1+\alpha})\log n^\alpha) \\ &= O(n^{2+o(1)}) \end{aligned}$$

となる。したがって、拡張した Gathen-Shoup の DDF を因子次数制限付き因数分解に適用した場合、現実的な高速化は期待されるものの漸近計算量は削減されない。

4.1.2 Kaltofen-Shoup アルゴリズムの拡張

2.2.2 節で議論したように、Kaltofen-Shoup の DDF では、Algorithm 2 の Step 1 で決定した ℓ, m に対して $n_0 = \ell m$ と置いたときに、n_0 が $n_0 \le n/2$ を満足すればアルゴリズムが動作するが、既約因子の次数が n^α 以下の場合には、$n_0 \le n^\alpha$ であっても十分である。したがって、Algorithm 2 の Step 1 を

1: $B \leftarrow \min(n^\alpha, n/2)$, $\ell \leftarrow \lceil B^\beta \rceil$, $m \leftarrow \lceil B/\ell \rceil$

と書き換えることが可能であり、この変更によって Kaltofen-Shoup アルゴリズムの DDF が因子次数制限付き因数分解に拡張される。以下では Algorithm 2 にこの拡張を施した拡張 DDF の計算量を評価する。以下の “Step i” は、Algorithm 2 の Step i を表す。まず、式 (7) より、

$$\begin{aligned} &O(\ell M(n)\log q) + O(m^{(\omega-1)/2}n^{(\omega+1)/2}) \\ &= O(n^{\alpha\beta+1+o(1)}\log q + n^{\alpha(1-\beta)(\omega-1)/2+(\omega+1)/2}) \end{aligned}$$

を得る。そこで、この式の 2 つの項の n の指数が等しくなるように

$$\beta = \frac{(\alpha+1)(\omega-1)}{\alpha(\omega+1)} \tag{16}$$

とすると、$\log q$ を固定したときの計算量の最適値が得られる。このとき、Step 2-4 の計算量は

$$O(n^{((\omega-1)\alpha+2\omega)/(\omega+1)+o(1)}\log q)$$

である。ただし、$\beta < 1$ より、この計算量でアルゴリズムが動作するためには $\alpha > (\omega-1)/2$ が必要である。したがって、$\omega = 2.376$ のとき $0.688 < \alpha \le 1$ の範囲で計算量

$$O(n^{0.408\alpha+1.408}\log q), \tag{17}$$

が得られ、$\omega = 2.808$ のとき $0.904 < \alpha \le 1$ の範囲で計算量

$$O(n^{0.475\alpha+1.475}\log q) \tag{18}$$

が得られる。また、$\alpha \le (w-1)/2$ の場合には、Cantor-Zassenhaus の DDF と同一の計算量

$$O(n^{\alpha+1+o(1)}\log q) \tag{19}$$

となる。

一方、Step 5 以降の計算量の合計は式 (6) より、

$$\begin{aligned} &O(M(\ell)M(n)\log\ell + M(m)M(n)\log m) \\ &\qquad = O(n^{\alpha\beta+1+o(1)} + n^{\alpha(1-\beta)+1+o(1)}) \end{aligned}$$

である。ここで行列乗算の性質より $2 \le \omega$ であることを考慮すると、式 (16) より

$$\beta \ge \frac{\alpha+1}{3\alpha} \ge \frac{2}{3}$$

を得る。したがって、Step 5 以降の計算量の合計は

$$O(n^{\alpha\beta+1+o(1)} + n^{\alpha(1-\beta)+1+o(1)}) = O(n^{\alpha\beta+1+o(1)}) \tag{20}$$

となり、Step 2-4 の計算量で抑えられる。

以上より、因子次数制限付き因数分解に対する Kaltofen-Shoup アルゴリズムの拡張 DDF の計算量は式 (17), (18) 及び (19) で与えられる。

4.1.3 NTL アルゴリズムの拡張

NTL アルゴリズムの因子次数制限付き因数分解への拡張は、Algorithm 2 の Step 1 に対して 4.1.2 節で示した Kaltofen-Shoup の DDF に対する拡張と同一の変更を加えることで達成される。そこで以下ではこの拡張を施した NTL アルゴリズムの DDF の計算量を評価する。

まず、Algorithm 2 の Step 2-4 の計算量は、式 (8), (9) より

$$\begin{aligned} &O(n^{\alpha/2}(n^2 + n^{1.5+o(1)}) + n^{1+o(1)}\log q) \\ &\qquad = O(n^{\alpha/2+2} + n^{1+o(1)}\log q) \end{aligned} \tag{21}$$

で与えられる。また、Step 5 以降の計算量は式 (10) より

$$O(\ell m M(n)\log n) = O(n^{\alpha+1+o(1)})$$

で与えられるが、これは式 (21) で抑えられる。したがって、式 (21) が因子制限付き因数分解に対する NTL アルゴリズムの DDF の計算量を与える。

表 2: 因子次数制限付き因数分解における DDF の計算量 (†: $\omega = 2.376$, ‡: $\omega = 2.808$, ††: $\omega = 3$)

Cantor-Zassenhaus	$O(n^{\alpha+1+o(1)} \log q)$
Gathen-Shoup	$O(n^{2+o(1)})$
Kaltofen-Shoup	$O(n^{0.408\alpha+1.408} \log q)$; $\alpha > 0.688$ †
	$O(n^{\alpha+1+o(1)} \log q)$; $\alpha \leq 0.688$ †
	$O(n^{0.475\alpha+1.475} \log q)$; $\alpha > 0.904$ ‡
	$O(n^{\alpha+1+o(1)} \log q)$; $\alpha \leq 0.904$ ‡
NTL	$O(n^{\alpha/2+2+o(1)} + n^{1+o(1)} \log q)$
Kedlaya-Umans	$O(n^{\alpha/2+1+o(1)} + n^{1+o(1)} \log q)$

4.1.4 Kedlaya-Umans アルゴリズムの拡張

Kedlaya-Umans アルゴリズムの因子制限付き因数分解への拡張は、NTL アルゴリズムの場合と同様に、Algorithm 2 の Step 1 に対して 4.1.2 節で示した Kaltofen-Shoup の DDF に対する拡張と同一の変更を加えることで達成される。この拡張を施した Kedlaya-Umans の DDF は、式 (11) より Algorithm 2 の Step 2-4 の計算量が

$$O(n^{\alpha/2} n^{1+o(1)} + n^{1+o(1)} \log q) = O(n^{\alpha/2+1+o(1)} + n^{1+o(1)} \log q) \quad (22)$$

で与えられる。一方、Step 5 以降は、Kaltofen-Shoup アルゴリズムと同一であるので、計算量が式 (20) で与えられるが、これは式 (22) で抑えられる。したがって Kedlaya-Umans アルゴリズムの DDF の計算量は式 (22) で与えられる。

4.2 因子次数制限付き因数分解の計算量

ここでは、4.1 節をまとめ、各アルゴリズムを因子次数制限付き因数分解に適用した場合の計算量を EDF を含めて考察する。そのために、まず表 2 に 4.1 節の結果をまとめる。

Cantor-Zassenhaus アルゴリズムでは、既約因子の次数に制限がない場合（即ち $\alpha = 1$）において、既に DDF と EDF の計算量が等しかったため、既約因子次数に制限がある場合においても（漸近的には）計算量が削減されなかった。一方、改良アルゴリズムの拡張は $\alpha = 1$ のとき DDF が律速だが、$\alpha < 1$ では EDF が律速する場合も考えられる。そこで、表 2 の結果と表 1 に示された EDF の計算量を比較し各拡張アルゴリズムにおいて DDF が計算量を律する α の範囲を評価する。まず、Kaltofen-Shoup アルゴリズムでは $\omega = 2.376$ のとき $\alpha \geq 0.189$、$\omega = 2.808$ のとき $\alpha \geq 0.404$ で DDF が律速であることが分かる。また、NTL アルゴリズムと Kedlaya-Umans アルゴリズムでは全ての α において DDF が計算量を律する。したがって、α の広い範囲で（α によって効率の変化する）拡張 DDF が律速であり、因子次数制限付き因数分解への DDF の拡張は、実際の速度向上のみならず、漸近計算量の観点からも有効である。

5 実装実験

前節では因子次数制限付き因数分解に拡張した DDF アルゴリズムの計算量評価を行ったが、本節では実装実験によって拡張 DDF アルゴリズムの効果を確認する。

実装は C++言語を用いて行い、NTL ライブラリ [Sho90] 利用した。また、コンパイラには GCC 4.4.7 を利用した。実装したアルゴリズムは、Kaltofen-Shoup アルゴリズムの因子次数制限付き因数分解への拡張 DDF と NTL アルゴリズムの因子次数制限付き因数分解への拡張 DDF、比較のための Cantor-Zassenhaus のオリジナル DDF (Algorithm 1) の 3 種類である。

NTL の拡張 DDF アルゴリズムは NTL ライブラリのソースコードを利用して作成した。Kaltofen-Shoup の拡張 DDF は、律速である Algorithm 2 の Step 4 までを実装し、Step 5 以降は NTL ライブラリのソースコードを利用した。また、行列乗算には D'Alberto と Nicolau [DN09] の Adaptive Winograd 乗算を利用した [2]。さらに、現実的な速度を得るために、パラメータ β に式 (16) で得た理論上の最適値を設定する代わりに、事前実験により実際の計算時間が最小となるパラメータ m, ℓ を推定し設定した。

実験は Intel Xeon E5-2643 3.30GHz 上の Linux 2.6 で行った。実験では $p = 2^{128} - 159$ と固定し、実装した DDF を $\mathbb{F}_p$ 上の squarefree 多項式に対して適用した。被分解多項式の次数が 10000, 20000, 40000, 80000 に対して、それぞれ α を 1, 0.8, 0.6, 0.4, 0.2 と変化させて各アルゴリズムの計算時間を計測した。Cantor-Zassenhaus アルゴリズムについては次数 10000, 20000 に対してのみ時間を計測した。表 3 に 10000 次、表 4 に 20000 次、表 5 に 40000 次と 80000 次の多項式に対する結果をそれぞれ示す。表中で "CZ" は Cantor-Zassenhaus アルゴリズム、"NTL" は NTL アルゴリズム, "KS" は Kaltofen-Shoup アルゴリズムを表す。また、表 6 に Kaltofen-Shoup アルゴリズムの拡張 DDF を 80000 次の多項式対して適用した場合の計算時間の詳細を示す。表 6 において各 Step は Algorithm 2 の Step に対応する。

各表のそれぞれの値は条件を満足するランダム多項式 8 個に対する計算時間の平均値であるが、Cantor-Zassenhaus アルゴリズムについては、値のばらつきが大きいので、平均値を "Ave." に示すとともに最大値を "Max." に示す。

Kaltofen-Shoup と NTL のオリジナルアルゴリズムは、$\alpha < 1$ に対しても $\alpha = 1$ の場合と同一の計算時間が必要であることに注意し、$\alpha = 1$ の場合と $\alpha < 1$ の場合とを比較すると、Kaltofen-Shoup と NTL の両アルゴリズムに対して DDF の拡張が有効であることが確認できる。また、α が小さいときには拡張 DDF よりも Cantor-Zassenhaus アルゴリズムの DDF の方が漸近計算量が小

[2] Adaptive Winograd 乗算は $\omega = 2.808$ のアルゴリズムである。

表 3: 拡張 DDF の実装結果（秒）: $\deg f = 10000$

α	CZ		NTL	KS
	Ave.	Max.		
1	49889	84987	836	916
0.8	6198	7173	441	510
0.6	966	1007	175	219
0.4	158	163	73	86
0.2	30	30	36	40

表 4: 拡張 DDF の実装結果（秒）: $\deg f = 20000$

α	CZ		NTL	KS
	Ave.	Max.		
1	137976	184402	4247	3548
0.8	24082	28323	2122	1803
0.6	3200	3611	781	726
0.4	447	461	292	278
0.2	69	69	118	94

表 5: 拡張 DDF の実装結果（秒）: $\deg f = 40000, 80000$

α	$\deg f = 40000$		$\deg f = 80000$	
	NTL	KS	NTL	KS
1	22302	13982	121251	56825
0.8	10460	6484	52561	24603
0.6	3603	2374	16751	7917
0.4	1214	855	5369	2631
0.2	404	268	1718	688

表 6: Kaltofen-Shoup の拡張 DDF の実装結果（秒）: $\deg f = 80000$

α	Step 2-3	Step 4	Step 5-	Total
1	21790	25065	9969	56825
0.8	10422	12211	1970	24603
0.6	3572	3967	377	7917
0.4	1365	1182	83	2631
0.2	340	327	21	688

さいが、今回の実験では広い範囲で拡張 DDF の方が高速であった。したがって、現実的な用途では因子次数制限付き素因数分解に対しては拡張 DDF の適用が有効であると考えられる。さらに、表 6 から、今回の実験の範囲では Kaltofen-Shoup の拡張 DDF において Step 2-4 が律速であることが分かる。したがって、本実装でとった、Step 5 以降に NTL ライブラリを利用する戦略も現実的な選択であることが確認できた。

6 まとめ

本論文では既約因子の次数に制限がある多項式の因数分解に対する Cantor-Zassenhaus アルゴリズム及びその改良アルゴリズムの摘要を議論し、まず、Cantor-Zassenhaus アルゴリズムの計算量が既約因子の次数に制限がある多項式に対して削減されるが改良アルゴリズムの計算量は削減されないことを述べた。そして、改良アルゴリズムについて既約因子の次数に制限がある多項式に対する拡張を示した。拡張されたアルゴリズムにおいて、Gathen-Shoup アルゴリズムでは計算量が削減されなかったものの、Kaltofen-Shoup アルゴリズム、Kaltofen-Shoup の「実用的な」アルゴリズム、Kedlaya-Umans アルゴリズムは既約因子の次数に制限がある多項式に対し計算量が削減されることを示した。さらに、Kaltofen-Shoup アルゴリズム、Kaltofen-Shoup の「実用的な」アルゴリズムの拡張を実装し、本論文で示した拡張が実際に効果的であることを確認した。

謝辞 実験で利用したプログラムの一部は石黒司氏が情報セキュリティ大学院大学在学中に原型を作成したものです。同氏に感謝致します。本研究は JSPS 科研費 25400055 の助成を受けたものです。

参考文献

[Ber67] E. R. Berlekamp, *Factoring polynomials over finite fields*, Bell System Technical Journal **46** (1967), 1853–1859.

[Ber70] ———, *Factoring polynomials over large finite fields*, Math. Comp **24** (1970), 713–735.

[BK78] R. P. Brent and H. T. Kung, *Fast algorithms for manipulating formal power series*, J. ACM **25** (1978), 581–595.

[BZ09] R. P. Brent and P. Zimmermann, *Factoring and testing irreducibility of sparse polynomials over small finite fields*, Talk at Queen Mary, U. London, `http://wwwmaths.anu.edu.au/~brent/pd/QMt4.pdf`, 2009.

[CW90] D. Coppersmith and S. Winograd, *Matrix multiplication via arithmetic progressions*, J.Symbolic Comp. **9** (1990), no. 3, 23–52.

[CZ81] D. G. Cantor and H. Zassenhaus, *A new algorithm for factoring polynomials over finite fields*, Math. Comp. **36** (1981), 587–592.

[DN09] P. D'Alberto and A. Nicolau, *Adaptive winograd's matrix multiplications*, ACM Transactions on Mathematical Software **36** (2009), no. 1.

[GG13] J. von zur Gathen and J. Gerhard, *Modern computer algebra*, 3rd ed., Cambridge U. P., 2013.

[GS92] J. von zur Gathen and V. Shoup, *Computing Frobenius maps and factoring polynomials*, STOC '92: Proceedings of the twenty-fourth annual ACM symposium on Theory of computing, ACM Press, 1992, pp. 97–105.

[IM10] T. Ishiguro and K. Matsuo, *Fields of definition of torsion points on the Jacobians of genus 2 hyperelliptic curves over finite fields*, Proc. of SCIS2010, 2D4-6, 2010.

[KS97] E. Kaltofen and V. Shoup, *Fast polynomial factorization over high algebraic extensions of finite fields*, ISSAC '97: Proceedings of the 1997 international symposium on Symbolic and algebraic computation, ACM Press, 1997, pp. 184–188.

[KU11] K. S. Kedlaya and C. Umans, *Fast polynomial factorization and modular composition*, SIAM J. Comput. **40** (2011), no. 6, 1767–1802.

[LN97] R. Lidl and H. Niederreiter, *Finite fields*, 2nd ed., Encyclopedia of Mathematics and its Applications, no. 20, Cambridge U. P., 1997.

[MB72] R. T. Moenck and A. Borodin, *Fast modular transforms via division*, 13th Annual Symposium on Switching and Automata Theory, IEEE Computer Society, 1972, pp. 90–96.

[Sho90] V. Shoup, *NTL: A library for doing number theory*, `http://www.shoup.net/ntl/`, 1990.

[Sho95] ———, *A new polynomial factorization algorithm and its implementation*, J. Symbolic Computation **20** (1995), 363–397.

[Sho09] ———, *A computational introduction to number theory and algebra*, 2nd ed., Cambridge U.P., 2009.

[Str69] V. Strassen, *Gaussian elimination is not optimal*, Numerishe Mathematik **13** (1969), 354–356.

[Yun76] D. Y. Y. Yun, *On square-free decomposition algorithm*, ISSAC: Proceedings of the ACM SIGSAM International Symposium on Symbolic and Algebraic Computation, ACM press, 1976, pp. 26–35.

[Zas69] H. Zassenhaus, *On Hensel factorization*, Journal of Number Theory **1** (1969), 587–592.