社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2006－CSEC－34（2）
2006／7／20

Type-II All One Polynomial Field 上での平方根導出アルゴリズムの高速実装

加藤 英洋† 王 鳳 野上 保之† 森川 良孝†

† 岡山大学自然科学研究科 〒700-8530 岡山市津島中 3-1-1
E-mail: †{katou,wangfeng,nogami,morikawa}@trans.cne.okayama-u.ac.jp

あらまし 近年，Weil ペアリングや Tate ペアリング等のペアリング技術を用いたグループ署名の研究が行われており，このペアリングには有限体上で定義される楕円曲線が用いられている．この楕円曲線上の有理点を求めるために定義体上の平方根導出が必要となるが，一般に平方根導出は他の計算に比べて時間がかかることが知られている．著者らは，F_{p^m} における高速な平方根導出アルゴリズムを提案しており，高速な四則演算が行える拡大体 AOPF (all one polynomial field) を提案している．上述の平方根導出アルゴリズムでは，計算にフロベニアス写像を用いている．また，AOPF はフロベニアス写像に計算を一切必要としないので上述の平方根導出アルゴリズムの実装に適している．本稿では，具体的には F_{p^6} 等の拡大次数において実装高速な平方根導出アルゴリズムを実装し，計算機シミュレーションを行った結果について報告する．

キーワード 楕円曲線，平方根導出

Square Root Calculation Algorithm over Type-II All One Polynomial Field

Hidehiro KATOU†, Wang FENG, Yasuyuki NOGAMI†, and Yoshitaka MORIKAWA†

† Graduate School of Natural Science and Technology, Okayama University Tsushimanaka 1-1-1, okayama-shi, Okayama, 700-8530 Japan
E-mail: †{katou,wangfeng,nogami,morikawa}@trans.cne.okayama-u.ac.jp

Abstract The authors proposes a square root (SQRT) algorithm in F_{p^m} ($m = r_0 r_1 \cdots r_{n-1} 2^d, r_i$: odd prime, $d > 0$: integer). We compute the inverse SQRT in $F_{p^{2^d}}$ using MW-ST algorithm. Then the Frobenius mappings with an addition chain are adopted for this SQRT algorithm, in which a lot of computations in a given extension field F_{p^m} are also reduce to those in a proper subfield by the norm computations. Those reductions of the field degree increases efficiency in the SQRT implementation. More specifically the Smart algorithm and proposed algorithm in F_{p^6}, for example, were implemented on a Pentium4 (3.8FHz) computer using the C++ programming language and NTL Library. The computer simulations showed that, on average, the proposed algoithm accelerates the SQRT computation by 3 times in F_{p^6}, compared to the Smart algorithm.

Key words Elliptic curve, Square root calculation.

1. まえがき

近年，Weil ペアリングや Tate ペアリング等のペアリング技術を用いたグループ署名の研究が盛んに行われており，このペアリングの実装には拡大体を定義体とした楕円曲線が用いられる．この楕円曲線では，曲線上の点である有理点を求めるために定義体における平方根計算が必要となる．しかし，一般に平方根計算は有限体の四則演算と比べて時間がかかることが知られている．著者らはこれまでに，高速な平方根計算法として moving window-sign testing(MW-ST) アルゴリズムを提案した．また，平方剰余判定および MW-ST を改良し，平方剰余判定においてノルム計算を用い，部分体の平方剰余判定に帰着させる．そして，そこで計算した結果を用いることによって高速化を行った平方根計算法を提案した．また，この手法を Bailey らによって提案された拡大体 optimal extension field(OEF) [1]，著者らが提案した拡大体 all one polynomial field(AOPF) において実装を行った．

本稿では，ノルム計算を用いた平方根計算法を，著者らが提案した拡大体 Type-II の all one polynomial field(Type-II AOPF) において実装を行う．高速化を行った平方根計算法お

よび平方剰余判定法では，ノルム計算を用いるが，この計算にはフロベニアス写像を用いる．Type-II AOPF において，フロベニアス写像は元の要素の並び替えによって実現可能であり，前計算を行うことによって，写像時に一切の計算を必要としないため，上記の平方根計算アルゴリズムの実装に適している．

平方根導出アルゴリズムの実装においては，比較対象として良く知られている Smart のアルゴリズムを用いる．計算機シミュレーションによって計算時間を測定した結果，6 次拡大体では平方剰余判定および平方根計算にかかった時間は，Smart のアルゴリズムに比べて，約 3 倍高速であった．その結果として問題を帰着させることにより高速化できていることを示す．

本稿を通して，p は有限体の標数，m は拡大体の拡大次数，F_{p^m} は標数 p 拡大次数 m を用いて構成した拡大体，$F_{p^m}^*$ は F_{p^m} の乗法に関する群を表す．

2. 数学的準備

本節では，Type-II AOPF について紹介する．

2.1 Type-II Optimal Normal Basis

まず，Type-II ONB は以下のように定義される [2].

【定義 2.1.1】 AOP $(x^{2m+1}-1)/(x-1)$ の根 ω を用いた m 個の元の組として以下を考える．

$$\{\omega+\omega^{-1},\omega^2+\omega^{-2},\ldots,\omega^m+\omega^{-m}\}. \tag{1}$$

m と p が以下の条件 1 を満たし，かつ 2a あるいは 2b のいずれかを満たすとき，式 (1) は F_{p^m} において基底をなす．この場合，式 (1) は正規基底でもあり，Type-II ONB と呼ぶ．

Type-II ONB の構成条件

1 ：$2m+1$ が素数である．

2a ：p が F_{2m+1} 上の原始元である．

2b ：$2 \mid (m-1)$ かつ F_{2m+1} 上の元 p の位数が m である．

条件 2a，2b を満たす場合，以下の性質より Type-II ONB を構成する．

- 条件 1，2a を満たす場合，$2m$ 次の AOP は F_p 上で既約となり，Type-II ONB を構成する．
- 条件 1，2b を満たす場合は法多項式は非既約となるが，互いに 2 つの m 次既約多項式に因数分解される．

2.2 Type-II AOPF の定義

拡大体 Type-II AOPF F_{p^m} は以下のように定義される．

【定義 2.2.1】 [3]:

標数 p ：擬メルセンヌ素数 $p=2^l \pm c$ を用いる．

l は $\log_2 c \leqq l/2$ とする．

法多項式 ：$2m$ 次 AOP を用いる．

$$(x^{2m+1}-1)\,/\,(x-1) = x^{2m}+x^{2m-1}+\cdots+x+1. \tag{2}$$

基底 ：前節で紹介した Type-II ONB を用いる．

$$\{\omega+\omega^{-1},\omega^2+\omega^{-2},\ldots,\omega^m+\omega^{-m}\}. \tag{3}$$

Type-II AOPF の特長として以下のようなものが挙げられる．

- Type-II ONB の構成条件より，$2m+1$ が素数なので拡大次数 m に奇数を設定できる．よって Type-II AOPF では奇数次拡大体を構成できる(注1)．

2.3 Cyclic Vector Multiplication Algorithm

Type-II AOPF の任意の元 X を次式のように表す．ここで $x_i \in F_p$ かつ $m \geqq i \geqq 1$ である．

$$X = \sum_{i=1}^{m} x_i(\omega^i+\omega^{-i}), \tag{4}$$

$$= x_1\omega+\cdots+x_m\omega^m+x_m\omega^{-m}+\cdots+x_1\omega^{-1} \tag{5}$$

さらに，Type-II AOPF の任意の元 Y と，$X \cdot Y = Z$ となる Z を考える．CVMA は一般式として以下の通りになる．

$$z_j = \sum_{k=1}^{m}(x_{\langle 2^{-1}j-k\rangle}-x_{\langle 2^{-1}j+k\rangle})(y_{\langle 2^{-1}j+k\rangle}-y_{\langle 2^{-1}j-k\rangle}) \tag{6}$$

ここで $m \geqq j \geqq 1$ である．例として $m=2$ の場合を考える．X, Y を以下のようにすると，

$$X = x_1(\omega+\omega^{-1})+x_2(\omega^2+\omega^{-2}), \tag{7a}$$

$$Y = y_1(\omega+\omega^{-1})+y_2(\omega^2+\omega^{-2}), \tag{7b}$$

となり，Z を計算すると，

$$Z = z_1(\omega+\omega^{-1})+z_2(\omega^2+\omega^{-2}), \tag{8}$$

$$z_1 = (x_1-x_2)(y_2-y_1)-x_1y_1, \tag{9a}$$

$$z_2 = (x_1-x_2)(y_2-y_1)-x_2y_2, \tag{9b}$$

となり，$(x_1-y_1)(y_2-y_1)$ のように重複する計算が現れる．この重複した計算の結果をメモリに格納しておくことによって，値を使いまわし，計算を効率化することができる．

2.4 Type-II AOPF における逆元計算法

Type-II AOPF において，逆元演算には伊東-辻井アルゴリズム (ITA) [5] を用いる．また，ITA の際に $\phi(X)=X^p$ となるフロベニアス写像を用いる．この写像は，

$$X = \sum_{i=1}^{m} x_i(\omega^i+\omega^{-i}) \tag{10}$$

とすると $\phi(X)$ は以下のようになる．

$$\phi(X) = X^p = \sum_{i=1}^{m} x_i^p(\omega^i+\omega^{-i})^p = \sum_{i=1}^{m} x_i(\omega^{ip}+\omega^{-ip}) \tag{11}$$

よって $(\omega^i+\omega^{-i})$ の要素は p 乗することによって $(\omega^j+\omega^{-j})$，$(j \equiv ip \bmod 2m+1)$ となる．このようにフロベニアス写像は元の要素の並び替えにより処理できる．よって $1 \leqq i \leqq m$ に対して $ip \bmod 2m+1$ を事前に計算しておけば，写像の際に一切の計算を必要としない．

(注1)：TypeI ONB を用いる TypeI AOPF では偶数次拡大体しか構成できない [4].

3. F_{p^m} 次拡大体における平方根計算法

本章では，著者らが提案した高速な平方剰余判定法と平方根計算法について紹介する．著者らが提案した手法では，F_{p^m} における平方根計算をノルム計算を用いて部分体 $F_{p^{2^d}}$ 上の平方根計算法に帰着させている．さらに，$F_{p^{2^d}}$ 上に帰着させた平方根計算結果を F_{p^m} 上の平方根に写像する際に，平方剰余判定で行った計算の結果の一部を使い回すことができる．この手法を用いて平方根計算を高速に行っている．以降，本アルゴリズムに用いる平方剰余判定・平方根計算について述べる．

3.1 平方剰余判定法

3.1.1 従来の平方根判定法

元 $x \in F_{p^m}$ において平方剰余判定は以下のように行われる．

$$C^m(x) = x^{(p^m-1)/2} = \begin{cases} 1, & x \text{ が平方剰余のとき} \\ -1, & x \text{ が平方非剰余のとき} \end{cases} \tag{12}$$

式 (12) において，$x^{(p^m-1)/2}$ は一般にはバイナリ法を用いて計算を行う．しかし，楕円曲線暗号では p に 160bits を超えるの素数を用いるので，このべき乗算には多くの拡大体上の乗算，2 乗算を必要とする．

3.1.2 平方剰余判定の高速実装

ここでは拡大体 F_{p^m} における平方剰余判定の高速実装法を紹介する．まず，拡大次数 m を $m = \bar{r}\bar{m}$ とする．ここで，$\bar{r}$ は奇素数，$\bar{m}$ は合成数もしくは 1 である．これを用いて式 (12) の乗数部分は以下のように変形できる．

$$\frac{(p^m-1)}{2} = \left[1 + p + p^{\bar{m}} + \cdots + (p^{\bar{m}})^{\bar{r}-1}\right] \cdot \frac{(p^{\bar{m}}-1)}{2} \tag{13}$$

これより，$C^m(x)$ は以下のようになる．

$$C^m(x) = C^{\bar{m}}(\bar{x}),\ \bar{x} = N^m_{\bar{m}}(x) = x^{1+p^{\bar{m}}+\cdots+p^{\bar{m}(\bar{r}-1)}} \tag{14}$$

式 (14) において，$N^m_{\bar{m}}(x)$ のノルム計算を行うことにより $C^m_{\bar{m}}(\bar{x})$ つまり $\bar{m}$ 次拡大体 (部分体) における平方剰余判定に帰着される(注2)．図 1 に $m = 30,\ 60$ の場合の C^m の計算における構造を示す．ここで $N_i(\cdot)$ はノルム計算を意味する．ノルム計算 $N^m_{\bar{m}}(x)$ ではフロベニアス写像を用いる．フロベニアス写像 $\phi(x) = x^p$ は拡大体上の元 x の p 乗を求める写像であり，i 回の繰り返しを $\phi^{[i]}(x) = x^{p^i}$ と表す．本稿で実装する Type-II AOPF におけるフロベニアス写像は，元の各要素の入れ替えによって実装できるので計算を必要としない．フロベニアス写像を用いることで $N^m_{\bar{m}}(x)$ は以下のようになる．

$$N^m_{\bar{m}}(x) = \prod_{i=0}^{\bar{r}-1} \phi^{[i\bar{m}]}(x) \tag{15}$$

さらに，フロベニアス写像を用いた $N^m_{\bar{m}}(x)$ の計算において加法連鎖を用いる．加法連鎖は計算を行った値を効果的に再利用する手法である．ここでは $N^m_{\bar{m}}(x)$ を計算するために，まず次式に示す $\Phi^m_{\bar{m}}, \bar{\Phi}^m_{\bar{m}}$ の計算を行う．

$$\Phi^m_{\bar{m}} = \prod_{i=1}^{(\bar{r}-1)/2} \phi^{(2i-1)\bar{m}}(x) \tag{16}$$

$$\bar{\Phi}^m_{\bar{m}} = \prod_{i=0}^{(\bar{r}-1)/2} \phi^{(2i)\bar{m}}(x) \tag{17}$$

加法連鎖には色々な手法があるが，上記式 (16),(17) のように偶数べき乗と奇数べき乗の式に分割して用いたのは，平方根計算のときに $\Phi^m_{\bar{m}}$(奇数べき乗の方の途中計算の結果) を用いるためである．加法連鎖の例として図 3 に $\bar{r} = 11$ における $N^m_{\bar{m}}(x)$ の計算法を示す．

(注2)：x は拡大体 F_{p^m} の元であり，$\bar{x} = N^m_{\bar{m}}(x)$ は部分体 $F_{p^{\bar{m}}}$ の元である．

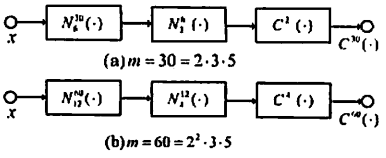

図 1 $C^m(x)$ の構造の例

Fig. 1 Examples of the structure of $C^m(x)$

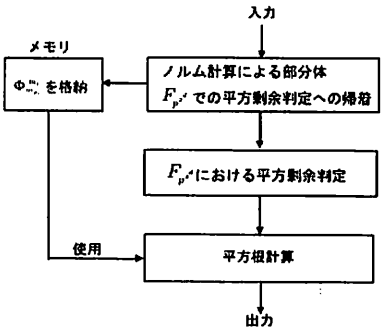

図 2 平方剰余判定・平方根計算のフローチャート

Fig. 2 Flow of SQRT calculation and QR test.

3.2 平方根計算・平方剰余判定

図 2 に平方剰余判定から平方根計算までの流れを示す．以降，これに沿って順に説明をする．

3.2.1 ノルム計算による部分体での平方剰余判定への帰着

拡大次数 $m = r_0 r_1 \cdots r_{n-1} 2^d$ とし，以下のように定義する．

$$m_j = r_j \cdots r_{n-1} 2^d \tag{18}$$

また，$m_0 = m,\ m_n = 2^d$ となる．$C^m(x)$ は以下に示すフロベニアス写像を用いたノルム計算 $\bar{x}_{j+1} := N^{m_j}_{m_{j+1}}(\bar{x}_j)$ をくり返し行うことによって $C^{2^d}(\bar{x}_n)$ に帰着する．

$$\bar{x}_{j+1} = N^{m_j}_{m_{j+1}}(\bar{x}_j) = \prod_{i=0}^{(r_j-1)} \phi^{[im_{j+1}]}(\bar{x}_j) \tag{19}$$

ここで $0 \leqq j \leqq n-1$，$\bar{x}_0 = x$ である．また，ノルム計算

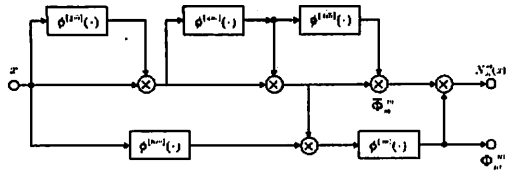

図 3　加法連鎖を用いた $r=11$ における $N_m^m(x)$ の計算例
Fig. 3　Example of the addition chain for $N_{m(x)}^m$, $r=11$.

$N_{m_{j+1}}^{m_j}(\bar{x}_j)$ には加法連鎖を用いて以下のように $\Phi_{m_{j+1}}^{m_j}, \bar{\Phi}_{m_{j+1}}^{m_j}$ を求める．

$$\Phi_{m_{j+1}}^{m_j} = \prod_{i=1}^{(r_j-1)/2} \phi^{(2i-1)m_{j+1}}(x) \tag{20}$$

$$\bar{\Phi}_{m_{j+1}}^{m_j} = \prod_{i=0}^{(r_j-1)/2} \phi^{(2i)m_{j+1}}(x) \tag{21}$$

また，すべての $\Phi_{m_{j+1}}^{m_j}$ $(0 \leqq j \leqq n-1)$ の値は平方根計算の際に用いるのでメモリに保存しておく．

式 (19) より，拡大体 F_{p^m} における平方剰余判定が部分体 $F_{p^{m_j}}$ $(0 \leqq j \leqq n-1)$における平方剰余判定に次から次へと帰着され，最終的には $F_{p^{2d}}$ における平方剰余判定となる．

3.2.2　$F_{p^{2d}}$ における平方剰余判定

まず，式 (22) に示す式が成り立つような整数 $T \geqq 0$ と奇数 s を求める．

$$p^{2d} = 2^T s + 1 \tag{22}$$

式 (22) より，乗法群 $F_{p^{2d}}^*$ は位数 $2^T s$ の巡回群となる．すなわち，$F_{p^{2d}}^*$ には 2^T を位数とする乗法群が存在する．この群を S_T とし，位数が 2^i の乗法群を S_i とすると，S_T の部分群は順に以下のようになる．

$$S_T \supset S_{T-1} \supset \cdots \supset S_2 \supset S_1 = \{\pm 1\} \supset S_0 = \{1\} \tag{23}$$

F_{p^m} における任意の平方非剰余元 c に対して，そのノルムの計算結果である $\bar{c} = N_{m_n}^m(c)$ は $F_{p^{2d}}$ において平方非剰余となる．また，位数 2^T の群 S_T に対して $c_0 := (\bar{c})^s$ は生成元となる．同様に位数 2^{T-1} の群 S_{T-1} に対して $c_1 := (c_0)^2$ は生成元となる．一般に位数 2^{T-k} の群 S_{T-k} に対して $c_k := (c_{k-1})^2$ は生成元となる $(k = 1, \cdots, T)$．また，$1 \leqq k \leqq d-1$ において c_k が存在する部分体は以下のようになる．

$$c_k \in S_{T-k} - S_{T-k-1} \subset F_{p^{2d}} \tag{24a}$$

$d \leqq k \leqq T-2$ においては次式のようになる．

$$c_k \in S_{T-k} - S_{T-k-1} \subset F_p, \text{when } 4 \mid (p-1) \tag{24b}$$

$$c_k \in S_{T-k} - S_{T-k-1} \subset F_{p^2}, \text{when } 4 \nmid (p-1) \tag{24c}$$

$k = T-1$ においては次式のようになる．

$$c_{T-1} \in S_1 - S_0 = \{-1\} \subset F_p \tag{24d}$$

上記のように，$d \leqq k \leqq T-2$ においては特殊であり，$4 \mid (p-1)$ の場合は，c_k は F_p の元であり，$4 \nmid (p-1)$ の場合は c_k は F_{p^2} の元となる．以上の関係を用いて平方剰余判定を行う．$\bar{x} = N_{m_n}^m(x)$ において，$x_0 := (\bar{x})^s$ が，式 (25) を満たす場合は平方非剰余，満たさない場合平方剰余となる．

$$x_0 \in S_T - S_{T-1} \tag{25}$$

実装においては以下の 4 つの場合分けを行い，計算をする．

a)　$1 < d$

これは，式 (24a) の場合であり，式 (26) を満たす場合，x_0 は $F_{p^{2d}}$ において平方非剰余となり，満たさない場合，x_0 は $F_{p^{2d}}$ において平方剰余となる．

$$x_0 \in F_{p^{2d}} - F_{p^{2d-1}} \tag{26}$$

b)　$d = 1$，$4 \nmid (p-1)$ のとき

これは式 (24c) の場合であり，$m = 2$ として式 (12) の判定を行う．

c)　$d = 1$，$4 \mid (p-1)$ のとき

これは式 (24b) の場合となるが，この場合は a) の場合と同様に平方剰余判定を行う．

d)　$d = 0$ のとき

これは式 (24d) の場合となり，楽体上の平方剰余判定となるので，$m = 1$ として式 (12) の判定を行う．

3.2.3　平方根計算

式 (14)，(19) よりノルム計算部は以下のようになる．

$$\bar{x}_{j+1} = N_{m_{j+1}}^{m_j}(\bar{x}_j) = \bar{x}_j^{(p^{m_{j+1}})^{r_j-1}+\cdots+p^{m_{j+1}}+1} \tag{27}$$

式 (27) の両辺に $(\bar{x}_j \bar{x}_{j+1})^{-1}$ を掛け，1/2 乗をすると次式となる．

$$(\bar{x}_j)^{-\frac{1}{2}} = (\Phi_{m_{j+1}}^{m_j})^{\frac{1+p^{m_{j+1}}}{2}} \cdot (\bar{x}_{j+1})^{-\frac{1}{2}}, 0 \leqq j \leqq n-1 \tag{28}$$

ここで $\Phi_{m_{j+1}}^{m_j}$ は式 (20) で与えられる．また，式 (28) において，$0 \leqq j \leqq n-1$ の範囲で式をまとめると次式になる．

$$(\bar{x}_0)^{-\frac{1}{2}} = (\bar{x}_n)^{-\frac{1}{2}} \cdot \prod_{j=0}^{n-1} (\Phi_{m_{j+1}}^{m_j})^{\frac{1+p^{m_{j+1}}}{2}} \tag{29}$$

$\bar{x}_0 = x, \bar{x}_n = \bar{x}$ より，両辺に x を掛けると式 (29) は次式のようになる．

$$x^{\frac{1}{2}} = \bar{x}^{-\frac{1}{2}} \cdot x \cdot \prod_{j=0}^{n-1} (\Phi_{m_{j+1}}^{m_j})^{\frac{1+p^{m_{j+1}}}{2}} \tag{30}$$

$\frac{1+p^{m_{j+1}}}{2}$ のべき乗算においては，以下のように式変形を行う．

$$\frac{p^{m_{j+1}}+1}{2} = [1 + p + \cdots + p^{(m_{j+1})-1}] \cdot \frac{p-1}{2} + 1 \tag{31}$$

$[\cdot]$ の部分を加法連鎖とフロベニアス写像を用いて計算し，$(p-1)/2$ にはバイナリ法を用いる．また，式 (30) の $\bar{x}^{-1/2} \in F_{p^{2d}}$ においては，MW-ST アルゴリズム [6] を用いて計算を行う．このように F_{p^m} における平方根計算を平方剰余判定で計算した値を用いることで $F_{p^{2d}}$ における平方根計算に帰着させ，高速に計算を行うことが可能となる．

4. コンピュータシミュレーション

本章では，平方剰余判定，平方根計算法を Type-II AOPF 上に実装し，計算時間を測定した結果を示す．今回使用した拡大次数，標数を以下に示す．

$p = 2^{160} + 7 \quad m = 2$
$p = 2^{160} + 7 \quad m = 3$
$p = 2^{160} + 7 \quad m = 5$
$p = 2^{160} + 471 \quad m = 6 = 3 \cdot 2$
$p = 2^{160} + 291 \quad m = 8 = 2^3$
$p = 2^{160} + 643 \quad m = 9$
$p = 2^{160} + 7 \quad m = 11$
$p = 2^{160} + 357 \quad m = 14 = 7 \cdot 2$

C++言語で NTL ライブラリを用い，Pentium4(3.8GHz)CPU において実装を行った．比較するアルゴリズムとしては，従来の平方剰余判定法，Smart の平方根計算アルゴリズム [2] を用いる．表 1 にランダムな平方剰余元を用いた平方根計算と平方剰余判定の平均時間を示す．拡大次数と各計算時間を見ると，従来の平方剰余判定法と Smart アルゴリズムは拡大次数が増加するにつれて計算時間も増加している．一方，本稿で述べた平方剰余判定と平方根計算法では，拡大次数の増加に対して計算時間のばらつきがある．これは，平方剰余判定の際，式 (14) において拡大次数の因数のうち奇素数のものを，計算を必要としないフロベニアス写像を用いたノルム計算を行い，部分体の計算へ帰着させることができるので，平方剰余判定は拡大次数の因数に 2 が少ないほど高速に行うことができるからである．また，本稿の平方根計算では，平方剰余判定におけるノルム計算に用いた値をくり返し用いるので，ノルム計算が少ない場合，つまり拡大次数の因数に奇素数が少ないほど平方根計算時に用いる計算が少なくなるので平方根計算が速くなっている．また，平方根計算の前には元の平方剰余性を判定する必要がある．平方剰余判定・平方根計算を合わせた計算時間を表 2 に示す．表 2 を見ると，従来の平方剰余判定と Smart アルゴリズムに比べて本稿で用いた平方剰余判定・平方根計算法ではそれぞれ拡大次数に含まれる奇数因数倍程度高速に計算を行うことができた．

表 1　平方根・平方剰余計算時間
Table 1　Running Time for SQRT and QR Test

F_{p^m}	計算方法	計算時間 (msec)
$m = 2$ $p = 2^{160} + 7$	Conventional QR test	3.44
	Fast QR test	5.94
	Smart algorithm	3.44
	Fast SQRT	0.15
$m = 3$ $p = 2^{160} + 7$	Conventional QR test	8.28
	Fast QR test	2.19
	Smart algorithm	7.97
	Fast SQRT	2.66
$m = 5$ $p = 2^{160} + 7$	Conventional QR test	30.0
	Fast QR test	4.38
	Smart algorithm	31.41
	Fast SQRT	5.78
$m = 6$ $p = 2^{160} + 471$	Conventional QR test	50.63
	Fast QR test	25.93
	Smart algorithm	54.06
	Fast SQRT	8.91
$m = 8$ $p = 2^{160} + 291$	Conventional QR test	115.62
	Fast QR test	111.72
	Smart algorithm	113.44
	Fast SQRT	0.93
$m = 9$ $p = 2^{160} + 643$	Conventional QR test	164.37
	Fast QR test	11.4
	Smart algorithm	164.85
	Fast SQRT	31.41
$m = 11$ $p = 2^{160} + 7$	Conventional QR test	267.97
	Fast QR test	16.25
	Smart algorithm	264.69
	Fast SQRT	22.97
$m = 14$ $p = 2^{160} + 357$	Conventional QR test	591.57
	Fast QR test	113.13
	Smart algorithm	599.84
	Fast SQRT	38.75

C++言語，NTL ライブラリ，Pentium4(3.8GHz)PC 使用

5. むすび

本稿では，著者らが提案したノルム計算を用いた平方根計算法を，同じく著者らが提案した拡大体 Type-II AOPF において実装を行った．計算機シミュレーションにおいては，比較相手として Smart のアルゴリズムを用いた．計算時間を測定した結果，平方剰余判定および平方根計算にかかった時間は，Smart のアルゴリズムに比べて，本稿の平方根計算法では，6 次拡大体では約 3 倍というように，測定した拡大次数の奇数因数倍高速であった．これは，平方剰余判定において，ノルム計算によって 2 のべき乗次の部分体に帰着させているからである．このように従来法に比べて高速に平方根計算を行うことができた．

文　献

[1] D.Bailey and C. Paar, "Optimal extension fields for fast arithmetic in public-key algorithms," CRYPTO 2002, LNCS 2442,pp.354-368,2002.

[2] I.Blake, G.Seroussi and N.Smart, "Elliptic Curves in Cryptography," LNS 265, Camprige Univ. Press, 1999

[3] Y.Nogami, S.Shinonaga and Y.Morilawa, "Fast Implementation of Extension Fields with Type-II ONB and Cyclic Vector Multiplication Algorithm," IEICE TRANS. FUNDAMENTALS,Vol.E88-A, No.5 May 2005

[4] Y.Nogami, A.Saito and Y.Morikawa, "Finite Extension Field with Modulus of All-Ome Polynomial and Representation of Its Elements for Fast Arithmetic Operations," IEICE TRANS. FUNDAMENTALS,Vol.E86-A, No.9 September 2003

[5] T.Itoh and S.Tsujii, "A fast algorithm for computing multiplicative inverses in $GF(2^m)$ using normal basis," Inf. Comput., vol.78,pp.171-177,1988.

[6] F. Wang, Y. Nogami and Y. Morikawa "An Efficient Square Root Computation In Finite Fields $GF(p^{2^d})$," IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, Vol. E88-A, No. 10, pp2792-2799, 2005.

表 2　平方根・平方剰余計算合計時間

Table 2　Total Running Time for SQRT and QR Test

F_{p^m}	計算方法	計算時間 (msec)
$m = 2$	C.QR test ans Smart algorithm	6.56
$p = 2^{160} + 7$	Fast QR test ans SQRT	6.09
$m = 3$	C.QR test ans Smart algorithm	16.25
$p = 2^{160} + 7$	Fast QR test ans SQRT	4.85
$m = 5$	C.QR test ans Smart algorithm	61.41
$p = 2^{160} + 7$	Fast QR test ans SQRT	10.16
$m = 6$	C.QR test ans Smart algorithm	104.69
$p = 2^{160} + 471$	Fast QR test ans SQRT	34.84
$m = 8$	C.QR test ans Smart algorithm	229.06
$p = 2^{160} + 291$	Fast QR test ans SQRT	112.65
$m = 9$	C.QR test ans Smart algorithm	329.22
$p = 2^{160} + 643$	Fast QR test ans SQRT	42.81
$m = 11$	C.QR test ans Smart algorithm	532.66
$p = 2^{160} + 7$	Fast QR test ans SQRT	39.22
$m = 13$	C.QR test ans Smart algorithm	1191.41
$p = 2^{160} + 357$	Fast QR test ans SQRT	151.88

C++言語，NTL ライブラリ，Pentium4(3.8GHz)PC 使用