2006－CSEC－35 (4)
2006／12／8

BREW 携帯電話でのペアリング暗号の高速実装

吉富 基† 高木 剛† 清本 晋作‡ 田中 俊昭‡

†公立はこだて未来大学情報アーキテクチャ学科
〒 041-8655 函館市亀田中野町 116-2

‡株式会社 KDDI 研究所
〒 356-8502 埼玉県ふじみ野市大原 2 丁目 1 番 15 号

概要 ペアリング暗号は ID ベース暗号など，従来の暗号方式にはない応用アプリケーションが実現できる．公開鍵暗号と比較してペアリング暗号は処理速度が遅いことが問題であったが，Duursma-Lee アルゴリズムやその改良版である η_T ペアリングにより比較的高速に実現できるようになった．本論文では，携帯電話などの比較的処理能力の低いユビキタスデバイスへのペアリング暗号の適用可能性を評価するため，携帯電話の BREW アプリケーションとしてソフトウェア実装を行った．$\mathbb{F}_{3^{97}}$ 上の超特異曲線を利用したペアリングを BREW 携帯電話 W41T,W41H において実装評価した結果，100 msec を切る処理速度を実現し，携帯電話用の暗号アプリケーションに十分適用可能であるとの結論を得た．

キーワード： ペアリング暗号，携帯電話，BREW，高速実装

Efficient Implementation of Pairing Computation on Mobile Phones using BREW

Motoi Yoshitomi† Tsuyoshi Takagi† Shinsaku Kiyomoto‡ Toshiaki Tanaka‡

†Future University - Hakodate, School of System Information Science
116-2, Kamedanakano-cho, Hakodate, 041-0806, Japan

‡KDDI R&D Laboratories Inc
2-1-15, Ohara, Fujimino, Saitama, 356-8502, Japan

Abstract Pairing based cryptosystems can accomplish novel security applications such as ID-based encryption etc. which have not been constructed without pairing. The processing speed of pairing based cryptosystems is relatively slow compared with the other conventional public key cryptosystems. However, several efficient algorithms for computing the pairing functions have been proposed, namely Duursma-Lee algorithm and its variant η_T pairing. In this paper, we present an efficient implementation of the pairing function over some mobile phones, and examine the feasibility of the pairing based cryptosystems on ubiquitous devices. Indeed the processing speed of our implementation in BREW on au W41T and W41H achieves under 100 milliseconds using the supersingular curve over $\mathbb{F}_{3^{97}}$. It has become fast enough for implementing security applications using the pairing function on mobile phones.

Keywords: Pairing Based Cryptosystems, Mobile Phone, BREW, Efficient Implementation

1 はじめに

Tate ペアリングを利用することにより short signature [5] や ID ベース暗号システム [4] など，従来の公開鍵暗号にはない，新たな暗号アプリケーションを実現できる．Tate ペアリングによる署名長は楕円曲線暗号の約半分となるため，short signature はメモリ領域が制限されたアプリケーションに適したデジタル署名技術である．また，従来の暗号では公開鍵は意味のない文字列であったが，ID ベース暗号システムでは，利用者が簡単に記憶できる E-mail アドレスや IP アドレスに置き換えることができる．

しかし，ペアリング暗号の演算速度が，他の暗号に比べ遅いことが問題点として挙げられている．文献 [2] では，ペアリング暗号は RSA 暗号や楕円曲線暗号の 5 倍以上遅いということが報告されている．一方，標数 3 の有限体上の超特異曲線を用いたペアリングの高速化手法として，Duursma-Lee アルゴリズム [12] や η_T ペアリング [1] が提案された．

また，ユビキタス社会の発展により，携帯電話などの比較的処理能力の低いユビキタスデバイス上でのペアリング暗号の利用を考える必要がある．特に，ユビキタスデバイスとして最も身近な携帯電話にお

いて，実装評価を行うことは重要である．携帯電話としては，Java プラットフォーム対応携帯電話と BREW プラットフォーム対応携帯電話の 2 種類があるが，Java プラットフォーム対応携帯電話によるペアリングの実装結果は既に報告されている [10]．本稿では，BREW プラットフォーム対応携帯電話 au W41T, W41H によるペアリングの実装について報告を行う．

ペアリング計算の高速化の検討は以下の手順で実施した．まず，高速化の検討を効率的に実施するため，各有限体に分けてプログラムを記述しており，有限体の演算ごとにプログラム関数を作成した．作成した関数を基にして，関数のタイミングによるプロファイルを取得することで，各有限体において処理時間を多く必要としている関数を調査した．その結果，有限体 $\mathbb{F}_{3^{97}}$ の乗算関数がプログラム全体の約 80% の処理時間を必要としていることが判明した．そこで乗算に重点をおいた高速化を行った．乗算アルゴリズムは，乗算を高速に行える Comb 法 [9] を用い，拡大次数 97 に特化した配列の工夫による改良を行った．また，プログラム内のループを展開し，パイプラインハザードの回数を減らすことによる高速化も行った．Duursma-Lee アルゴリズムに関しては，拡大体 $\mathbb{F}_{3^{6\cdot 97}}$ において，0 や 2 といった定数を含む乗算処理が存在することを利用した高速化を行った．

上記のような高速化に際する検討を行い，Duursma-Lee アルゴリズム，η_T ペアリングにおいてペアリング計算の比較評価を行った．標数 3 の有限体 $\mathbb{F}_{3^{97}}$ 上の超特異曲線を用いた η_T ペアリングを携帯電話 W41T，W41H で実装した結果は，100 msec を切る処理速度を実現できた．

以下本稿の構成として，2 章では有限体の演算について述べ，3 章でペアリングについてを紹介する．4 章では BREW によるペアリングの実装とその結果について述べ，5 章においてまとめとする．

2 有限体の演算

本章では標数 3 の有限体 $\mathbb{F}_3$ を m 次拡大した有限体 $\mathbb{F}_{3^m}$ と，その 3 次拡大体である $\mathbb{F}_{3^{3m}}$，6 次拡大体である $\mathbb{F}_{3^{6m}}$ の演算及び有限体 $\mathbb{F}_{3^m}$ の乗算について高速化したアルゴリズムについて説明する．

安全性を考慮すると，暗号に利用するには現在のコンピュータスペックで，有限体 $\mathbb{F}_{3^{6m}}$ の離散対数問題を解読困難にするため，$3^{6m} \approx 1024$ ビット程度以上が望まれており，他の文献 [1, 2, 3, 7, 10, 11] で採用されている $m = 97$ を選んだ．

2.1 元の表現

係数が基礎体 $\mathbb{F}_3 = \{0, 1, 2\}$ の元である多項式を $\mathbb{F}_3[x]$ と表すとき，有限体 $\mathbb{F}_{3^m}$ は，既約多項式 $f(x)$ を $f(x) = x^{97} + x^{12} + 2$ として，$\mathbb{F}_{3^m} = \mathbb{F}_3[x]/f(x)$ と表される多項式全体の集合である．$\mathbb{F}_{3^m}$ の元 $A(x)$ は係数である $\mathbb{F}_3$ の元を m 個並べ，

$$A(x) = (a_{m-1}, a_{m-2}, \cdots, a_1, a_0),\ a_i \in \mathbb{F}_3$$

と表すことにする．

$\mathbb{F}_3 = \{0, 1, 2\}$ は状態が 3 つであり，1 ビットで表せる状態が $\{0, 1\}$ の 2 つであるため，2 ビットを用いて $\mathbb{F}_3$ の元を表現する．この 2 ビットを hi ビット，lo ビットと呼ぶことにし，$\mathbb{F}_{3^m}$ の元 $A(x)$ は (hi, lo)-ビットからなる 2 つの配列で表す [10]．対象とする CPU のワード長を W とすると，配列の要素数は $\lceil m/W \rceil$ となり，今回の実行環境で使用した CPU ワード長は 32 ビットであるため，配列の要素数は 4 である．$A(x)$ の 2 つの配列を合わせて $A[j]$ と表記し，
$A[3] = (0, \cdots, 0, a_{96}), A[2] = (a_{95}, a_{94}, \cdots, a_{64})$,
$A[1] = (a_{63}, a_{62}, \cdots, a_{32}), A[0] = (a_{31}, a_{30} \cdots, a_0)$
を保持する．また，$A[j]$ の k 番目の (hi, lo)-ビットを $A[j]_k$ のように表現し，例として $A[1]_0$ は a_{32} を表す．

2.2 $\mathbb{F}_{3^m}$ の乗算

$A(x), B(x) \in \mathbb{F}_{3^m}$ に対する乗算として，Shift-and-Add 法 [2] と呼ばれるアルゴリズムがある．Shift-and-Add 法は $A(x)$ を (hi, lo)-ビット毎に 1 つ左へシフトし $B(x)$ の各桁の (hi, lo)-ビット情報に基づいて加算を行う方法である．また，Shift-and-Add を高速化した Comb 法 [9] と呼ばれるアルゴリズムがある．Comb 法 は $A(x)$ の配列の要素数分 4 つを並列に処理するため，$A(x)$ のシフトの回数が Shift-and-Add 法と比較して 1/4 となる．今回，Comb 法を用いることで，Shift-and-Add 法よりシフト処理を削減した．

Algorithm 1 改良 Comb 法 $m = 97$

INPUT: $A(x), B(x) \in \mathbb{F}_{3^m}, W$: ワード長
OUTPUT: $C(x) = A(x) \cdot B(x) \bmod f(x) \in \mathbb{F}_{3^m}$

```
1: C(x) ← 0
2: for j ← 0 to 3 do
3:     C(x) ← C(x) + A[j]_0 B(x) x^{jW}
4: end for
5: for i ← 1 to W − 1 do
6:     for j ← 0 to 2 do
7:         C(x) ← C(x) + A[j]_i B(x) x^{jW+i}
8:     end for
9: end for
10: for i ← 2m − 2 downto m do
11:     c_{i−85} ← c_{i−85} − c_i
12:     c_{i−97} ← c_{i−97} + c_i
13:     c_i ← 0
14: end for
15: return C(x)
```

Algorithm 1 に改良 Comb 法を示す．このアルゴリズムは，Comb 法を拡大次数 $m = 97$ に特化して高速化したアルゴリズムである．$A[3]$ では，$A[3]_0$ に 96 次の係数のみ保持しているため，$A[3]_1$ から $A[3]_{31}$ までが 0 である．この 0 の部分に注目し，$A[j]_0$ の

加算処理部分のループを展開することにより，以降 $A[3]$ の加算処理を行わず，全体のループ数を 1 回減らすことができる．Algorithm 1 のステップ 1〜9 が Comb 法であり，ステップ 10〜14 がリダクションと呼ばれ，$\mathbb{F}_3[x]$ の元を $\mathbb{F}_{3^m}$ の元とする処理である．Comb 法の主な演算処理はステップ 5〜9 であり，ステップ 2〜4 が $A[j]_0$ のループ展開部分である．ステップ 2 に比べステップ 6 ではループの回数が 4 回から 3 回と，1 回分減っている．

2.3 $\mathbb{F}_{3^m}$ のその他の演算

$A(x), B(x) \in \mathbb{F}_{3^m}$ に対する加算は，次数に対応する各係数同士の $\mathbb{F}_3$ の加算である．各係数の (hi, lo)-ビットに対し，論理演算子 AND，OR，XOR を用いて演算を行い，値を更新する．$\mathbb{F}_{3^m}$ の減算についても $\mathbb{F}_{3^m}$ の加算と同様の演算処理を行っている [7]．

$A(x) \in \mathbb{F}_{3^m}$ に対する 3 乗算は，$A(x)^3 = \sum_{i=0}^{m-1} a_i x^{3i}$ と計算できる．各係数を次数に対応する係数に引き伸ばすためのテーブルを作成し，このテーブルに従って各係数の (hi, lo)-ビットの引き伸ばし処理を行う [7]．この引き伸ばし処理は，乗算処理を必要としないため高速である．

$A(x) \in \mathbb{F}_{3^m}$ に対する乗法逆元は，標数 2 の拡張ユークリッド互除法 [9] を改良し，標数 3 に対応した拡張ユークリッド互除法 [10] を用いて実装した．

また，$\mathbb{F}_{3^m}$ の各演算のコストは，加算・減算コストを A，乗算コストを M，3 乗算コストを C，乗法逆元コストを I として表すと，次の関係となる．

$$A < C < M < I$$

我々の実装では，3 個の不等号におけるコストの差は，それぞれおよそ 10 倍程度であった (4.4 節参照)．

2.4 拡大体 $\mathbb{F}_{3^{3m}}, \mathbb{F}_{3^{6m}}$ の演算

本稿では，$\mathbb{F}_{3^m}$ を規約多項式 $g(x) = x^3 - x - 1$ により 3 次拡大した $\mathbb{F}_{3^{3m}} = \mathbb{F}_{3^m}[x]/g(x)$，および $\mathbb{F}_{3^{3m}}$ を規約多項式 $h(x) = x^2 + 1$ により更に 2 次拡大した $\mathbb{F}_{3^{6m}} = \mathbb{F}_{3^{3m}}[x]/h(x)$ のアルゴリズムを取り扱う．多項式 $g(x), h(x)$ の根をそれぞれ ρ, σ とすると，関係式 $\rho^3 = \rho + 1, \sigma^2 = -1$ を満たす．ここで，$\mathbb{F}_{3^{6m}}$ の元 A は，$\alpha_i \in \mathbb{F}_{3^{3m}}, a_j \in \mathbb{F}_{3^m}$ に対して，

$$\begin{aligned} A &= \alpha_1\sigma + \alpha_0 \\ &= a_5\sigma\rho^2 + a_4\sigma\rho + a_3\sigma + a_2\rho^2 + a_1\rho + a_0 \end{aligned}$$

と表される $(i = 0, 1, j = 0, 1, \cdots, 5)$．$\mathbb{F}_{3^{3m}}, \mathbb{F}_{3^{6m}}$ の加算，減算，乗算，3 乗算，乗法逆元いずれの演算も論文 [7] と同様の演算処理を行っている．

$\mathbb{F}_{3^{3m}}, \mathbb{F}_{3^{6m}}$ の演算に必要とされる $\mathbb{F}_{3^m}$ の演算コストを表 1 に示す．表 1 より，$\mathbb{F}_{3^{6m}}$ の加算・減算と 3 乗算のコストは $\mathbb{F}_{3^{3m}}$ の丁度 2 倍であり，乗算と乗法逆元のコストは 2 倍以上である．

演算	$\mathbb{F}_{3^{3m}}$	$\mathbb{F}_{3^{6m}}$
加算・減算	3A	6A
乗算	12A + 6M	51A + 18M
3 乗算	3A + 3C	6A + 6C
乗法逆元	6A + 15M + 1I	57A + 38M + 1I

表 1: $\mathbb{F}_{3^{3m}}, \mathbb{F}_{3^{6m}}$ の演算に必要な $\mathbb{F}_{3^m}$ の演算コスト

3 ペアリングのアルゴリズム

本章ではペアリングのアルゴリズムを示し，ペアリング計算の高速化の検討について考察する．

3.1 Tate ペアリング

Tate ペアリングでは有限体上の楕円曲線の演算を必要とする．利用する楕円曲線は標数 3 の超特異曲線であり，有限体 $\mathbb{F}_{3^m}$ 上の楕円曲線を $E(\mathbb{F}_{3^m})$ として以下の曲線で表される．

$$E(\mathbb{F}_{3^m}) = \{(x, y) \in (\mathbb{F}_{3^m})^2 \mid y^2 = x^3 - x + 1\} \cup \{\mathcal{O}\}$$

ただし $\mathcal{O}$ は無限遠点とする．この楕円曲線 $E(\mathbb{F}_{3^m})$ の位数 $\sharp E(\mathbb{F}_{3^m})$ は $\sharp E(\mathbb{F}_{3^m}) = 3^m + 3^{(m+1)/2} + 1$ となる．

r は $r|\sharp E(\mathbb{F}_{3^m})$ となるような素数として，$r|(3^{6m} - 1)$ を満たす必要がある．$E(\mathbb{F}_{3^m})$ の位数 r の部分群を $E(\mathbb{F}_{3^m})[r]$ とすると，Tate ペアリングは以下のように定義される．

$$e(\cdot, \cdot) : E(\mathbb{F}_{3^m})[r] \times E(\mathbb{F}_{3^{6m}})[r] \to \mathbb{F}^*_{3^{6m}}/(\mathbb{F}^*_{3^{6m}})^r$$

$E(\mathbb{F}_{3^{6m}})$ の元は，点 $Q = (x, y) \in E(\mathbb{F}_{3^m})$ を $E(\mathbb{F}_{3^{6m}})$ へリフティングする distortion 写像 $\phi(x, y) = (-x + \rho, y\sigma)$ を利用して生成できる．

Tate ペアリングは $P,\ Q \in E(\mathbb{F}_{3^m})[r],\ a \in \mathbb{Z}$ に対し，双線形性 $e(aP,\ Q) = e(P,\ aQ) = e(P,\ Q)^a$ を満たす．

3.2 アルゴリズム

Tate ペアリングの演算アルゴリズムは Miller によって最初に提案され [13]，その後 Duursma と Lee により，標数 3 の有限体上の超特異曲線において Closed Form を利用した効率的なアルゴリズムが提案された [6]．現在では 3 乗根の演算を削除した改良アルゴリズムである Duursma-Lee アルゴリズム [12, Table 4] が知られている．Duursma-Lee アルゴリズムを Algorithm 2 に示す．

Duursma-Lee アルゴリズムでは最終冪と呼ばれるステップが存在し，$\mathbb{F}_{3^{6m}}$ の元を T として，$T^{(3^{3m}-1)}$ を計算する必要がある．この計算は $T = \tau_1\sigma + \tau_0$ に対して，$T^{(3^{3m}-1)} = (-\tau_1\sigma + t_0)(\tau_1\sigma + \tau_0)^{-1}$ と式変形を行うことで高速に計算できる [11]．実際に Algorithm 3 では，最終冪 $T^{(3^{3m}-1)}$ が $\mathbb{F}_{3^{6m}}$ の乗算

Algorithm 2 Duursma-Lee アルゴリズム [12]

INPUT: $P=(x_p,y_p), Q=(x_q,y_q) \in E(\mathbb{F}_{3^m})[r]$
OUTPUT: $e(P,Q) \in \mathbb{F}_{3^{6m}}$
1: initialization:
  $T \leftarrow 1$ (in $\mathbb{F}_{3^{6m}}$)
  $a \leftarrow x_p, b \leftarrow y_p, x \leftarrow {x_q}^3, y \leftarrow {y_q}^3$ (in $\mathbb{F}_{3^m}$)
  $d \leftarrow 1$ (in $\mathbb{F}_3$)
2: **for** $i \leftarrow 0$ to $m-1$ **do**
3: $a \leftarrow a^9$, $b \leftarrow b^9$ (in $\mathbb{F}_{3^m}$)
4: $c \leftarrow a+x+d$ (in $\mathbb{F}_{3^m}$)
5: $R \leftarrow -by\sigma - \rho^2 - c\rho - c^2$ (in $\mathbb{F}_{3^{6m}}$)
6: $T \leftarrow T^3$ (in $\mathbb{F}_{3^{6m}}$)
7: $T \leftarrow TR$ (in $\mathbb{F}_{3^{6m}}$)
8: $y \leftarrow -y$ (in $\mathbb{F}_{3^m}$)
9: $d \leftarrow d-1$ (in $\mathbb{F}_3$)
10: **end for**
11: final exponentiation:
  $T \leftarrow$ Algorithm_3 (T)
12: **return** T

Algorithm 3 最終冪 (Duursma-Lee アルゴリズム)

INPUT: $T=\tau_1\sigma+\tau_0 \in \mathbb{F}_{3^{6m}}$
OUTPUT: $T^{(3^{3m}-1)} \in \mathbb{F}_{3^{6m}}$
  $U \leftarrow T^{-1}$ (in $\mathbb{F}_{3^{6m}}$)
  $\tau_1 \leftarrow -\tau_1$ (in $\mathbb{F}_{3^{3m}}$)
  $T \leftarrow UT$ (in $\mathbb{F}_{3^{6m}}$)
  return T

1 回と乗法逆元 1 回で計算可能であり，3 乗算を $3m$ 回と乗法逆元 1 回で演算するより高速となる．

また，$E(\mathbb{F}_{3^m})$ の Frobenius 写像を利用することにより，メインループの回数が Duursma-Lee アルゴリズムの約半分となる η_T ペアリング [1] が知られている．η_T ペアリングに対しても，3 乗根の演算を削除した改良アルゴリズムが提案されている [3, Alg.2]．η_T ペアリングのアルゴリズムを Algorithm 4 に示す．

η_T ペアリングにおいても最終冪と呼ばれるステップが存在し，

$$S=(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)$$

に対して T^S を計算する．$\eta_T(P,Q)$ は双線形性を満たさないが，$\eta_T(P,Q)^S$ は双線形性を満たす．T^S の計算は Algorithm 5 に従って計算する．$T^{(3^{3m}-1)}$ の冪乗部分は Duursma-Lee アルゴリズムの最終冪の計算と同様である．一方，$T^{(3^m+1)(3^m-3^{(m+1)/2}+1)}$ の冪乗部分は文献 [3] と同様な方法で乗算・3 乗算を繰り返す必要があるため，最終冪の演算コストは η_T ペアリングの方が大きい．また，最終冪の計算において，冪乗部分を展開して乗法逆元を 1 回用いて計算するよりも，乗法逆元を 2 回用いる方が高速に計算できる．

DL (Alg.2)	4635A + 972C + 1511M + 1I
(最終冪 (Alg.3))	(108A + 56M + 1I)
η_T (Alg.4)	4359A + 1654C + 1129M + 2I
(最終冪 (Alg.5))	(1000A + 1164C + 148M + 2I)

表 2: ペアリングに必要とされる $\mathbb{F}_{3^m}$ の演算コスト

Algorithm 4 η_T ペアリング [3]

INPUT: $P=(x_p,y_p), Q=(x_q,y_q) \in E(\mathbb{F}_{3^m})[r]$,
  $S=(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)$
OUTPUT: $(\eta_T(P,Q)^S)^{3^{(m+1)/2}} \in \mathbb{F}_{3^{6m}}$
1: initialization:
  $a \leftarrow x_p, b \leftarrow -y_p, x \leftarrow x_q, y \leftarrow y_q$ (in $\mathbb{F}_{3^m}$)
  $d \leftarrow 1$ (in $\mathbb{F}_3$)
  $c \leftarrow a+x+d$ (in $\mathbb{F}_{3^m}$)
  $T \leftarrow y\sigma + b\rho - bc$ (in $\mathbb{F}_{3^{6m}}$)
2: **for** $i \leftarrow 0$ to $(m-1)/2$ **do**
3: $c \leftarrow a+x+d$ (in $\mathbb{F}_{3^m}$)
4: $R \leftarrow by\sigma - \rho^2 - c\rho - c^2$ (in $\mathbb{F}_{3^{6m}}$)
5: $T \leftarrow TR$ (in $\mathbb{F}_{3^{6m}}$)
6: $T \leftarrow T^3$ (in $\mathbb{F}_{3^{6m}}$)
7: $b \leftarrow -b$ (in $\mathbb{F}_{3^m}$)
8: $x \leftarrow x^9$, $y \leftarrow y^9$ (in $\mathbb{F}_{3^m}$)
9: $d \leftarrow d-1$ (in $\mathbb{F}_3$)
10: **end for**
11: final exponentiation:
  $T \leftarrow$ Algorithm_5 (T)
12: **return** T

Algorithm 5 最終冪 (η_T ペアリング)

INPUT: $T=\tau_1\sigma+\tau_0 \in \mathbb{F}_{3^{6m}}$,
  $S=(3^{3m}-1)(3^m+1)(3^m-3^{(m+1)/2}+1)$
OUTPUT: $T^S \in \mathbb{F}_{3^{6m}}$
1: $U \leftarrow T^{-1}$ (in $\mathbb{F}_{3^{6m}}$)
2: $\tau_1 \leftarrow -\tau_1$ (in $\mathbb{F}_{3^{3m}}$)
3: $T \leftarrow UT$ (in $\mathbb{F}_{3^{6m}}$)
4: $A \leftarrow T$ (in $\mathbb{F}_{3^{6m}}$)
5: **for** 0 to 96 **do**
6: $T \leftarrow T^3$ (in $\mathbb{F}_{3^{6m}}$)
7: **end for**
8: $T \leftarrow AT$ (in $\mathbb{F}_{3^{6m}}$)
9: $A \leftarrow T$ (in $\mathbb{F}_{3^{6m}}$)
10: **for** 0 to 48 **do**
11: $T \leftarrow T^3$ (in $\mathbb{F}_{3^{6m}}$)
12: **end for**
13: $U \leftarrow T^{-1}$ (in $\mathbb{F}_{3^{6m}}$)
14: **for** 0 to 47 **do**
15: $T \leftarrow T^3$ (in $\mathbb{F}_{3^{6m}}$)
16: **end for**
17: $T \leftarrow AT$ (in $\mathbb{F}_{3^{6m}}$)
18: $T \leftarrow UT$ (in $\mathbb{F}_{3^{6m}}$)
19: **return** T

η_T ペアリング (η_T) と Duursma-Lee アルゴリズム (DL) アルゴリズムの演算コストを表 2 に示す．表 2 から判るように，実際に η_T ペアリングの最終冪のコストが Duursma-Lee アルゴリズムの最終冪のコストに比べ大きい．

Duursma-Lee アルゴリズムと η_T ペアリングの出力値の関係は次のようになっている．

$$\eta_T(P,Q)^{3(3^{(m+1)/2}+1)^2} = e(P,Q)^{-3^{(m+3)/2}}$$

また，$\eta_T(P,Q)^S$ から $e(P,Q)$ を求めるには $U=\eta_T(P,Q)$ とおくと，

$$e(P,Q)=\left(U^2 \cdot U^{3^{(m+1)/2}} \cdot \sqrt[3^m]{U^{3^{(m-1)/2}}}\right)^{-1}$$

となる [3].

4 BREWによるペアリングの実装

この節ではBREW[1] プラットフォーム対応携帯電話に対するペアリングの実装と，ペアリングの計算の高速化について考察する．

4.1 プログラムの構造と基本関数

ペアリング計算プログラムのソースコードは，各有限体，楕円曲線及びペアリングアルゴリズムに分けて記述しており，それぞれの演算ごとにプログラム関数を作成した．この関数には，$\mathbb{F}_{3^m}$ の加算はFF_Add，$\mathbb{F}_{3^m}$ の乗算はFF_Multi，$\mathbb{F}_{3^m}$ の3乗算はFF_Cubeと関数名を付けている．

携帯電話用のコンパイルには，BREWアプリケーション標準のコンパイラであるARMコンパイラを使用した．ソースコードの作成とARMコンパイラによるコンパイルはPCで行った．コンパイルによって作成されたファイルはBREWアプリケーションの実行ファイル(*.mod)となる．この実行ファイルを携帯電話上のBREW環境で動作させ速度を計測する実験を行う．

4.2 プログラムの解析

ペアリングの実装では，各有限体において処理時間を多く必要としている関数を，関数のプロファイルを取得することにより調査し考察した．

Duursma-Leeアルゴリズムと η_T ペアリングのプロファイルの結果の一部を表3に示す．ただし，この結果は各アルゴリズムのみのデータではなく，楕円曲線上の点の算出などの主要計算部分以外の演算も数%含んでいる．

・Duursma-Leeアルゴリズム

関数時間	%	ヒットカウント	関数
534.520	80.0	151400	_FF_Multi
51.035	7.6	97776	_FF_Cube
37.052	5.5	343906	_FF_Add

・η_T ペアリング

関数時間	%	ヒットカウント	関数
338.506	78.6	113200	_FF_Multi
53.508	12.4	165976	_FF_Cube
10.281	2.4	351406	_FF_Add

表3: 関数のタイミングによるプロファイル結果

表3に示されているFF_Multiは $\mathbb{F}_{3^m}$ での乗算関数である．Duursma-Leeアルゴリズム，η_T ペアリングは，ともに $\mathbb{F}_{3^m}$ の乗算の処理時間が全体の約80%を占めおり，各アルゴリズムの高速化を行うにあたり，$\mathbb{F}_{3^m}$ の乗算関数に重点をおいて高速化を行った．

4.3 高速化の検討

Duursma-Leeアルゴリズムに関しては，以下の高速化を実施した．Algorithm 2のステップ7において，$\mathbb{F}_{3^{6m}}$ の乗算処理を行っており，$\mathbb{F}_{3^{6m}}$ の元 $R = r_5\sigma\rho^2 + r_4\sigma\rho + r_3\sigma + r_2\rho^2 + r_1\rho + r_0$ において，$r_4 = r_5 = 0$，$r_2 = 2$ を満たす．この定数部分を考慮し，$\mathbb{F}_{3^{3m}}$ の乗算を減らすための $\mathbb{F}_{3^{6m}}$ の特別な乗算関数EF6_MultiSpecialを作成し，また，関数EF6_MultiSpecial内で定数部分を使用する $\mathbb{F}_{3^{3m}}$ の特別な乗算関数EF3_MultiSpecialも作成した．この2つの特別な乗算関数を使用することにより，Duursma-Leeアルゴリズムで使用する乗算関数FF_Multiの回数を減らすことができる．

次に，$\mathbb{F}_{3^m}$ の各演算アルゴリズムについて，$\mathbb{F}_{3^m}$ の元は m 個の (hi, lo)-ビットであることに対し，実行環境のCPUワード長に依存した32ビットずつの処理を行うことになる．$\mathbb{F}_{3^m}$ の加算を例にとると，m 個の (hi, lo)-ビット全ての加算処理は4回のループで構成される．このループを展開することにより，プロセッサのパイプラインハザードの回数が減ることが期待され高速化が望める．実際にループを展開したことにより約30%の処理速度の向上が図れた．

なお，Algorithm 1において窓幅を設けたWindow法による実装も行った．PC上での実装では，窓幅2の実装は窓幅1の実装に比べて速度の向上が図れた．しかし，携帯電話での実装では逆に速度が低下したため採用していない．理由として予備計算が携帯電話のCPUキャッシュに格納できなかったことが考えられる．

4.4 実装結果

$\mathbb{F}_{3^m}$ の各演算速度とDuursma-Leeアルゴリズム，η_T ペアリングでの演算速度を表4に示す．ペアリング計算プログラムの実装はPC，携帯電話au W41T，W41Hで行った．使用したPCのスペックは，CPU: AMD Opteron™ Processor 246 (2.0GHz)，RAM: 1GByteである．携帯電話でのコンパイラは4.1節で示したARMコンパイラを用い，コンパイルオプションは実行速度において最適化を行うため，デフォルトの-Ospaceを-Otimeに変更した．PCでのコンパイラはGCC 3.4.2を用い，コンパイルオプションは-O2 -fomit-frame-pointerとした．

各演算の1回分の測定は，携帯電話では加算・減算200,000回，乗算1,500回，3乗算3,000回及び乗法逆元250回の平均で求め，Duursma-Leeアルゴリズム及び η_T ペアリングは100回の平均で求めた．PCでは $\mathbb{F}_{3^m}$ の加算・減算2,000,000回，乗算

[1] BREW™ は，Qualcomm 社の登録商標であり cdmaOne および cdma2000 携帯電話用に開発したアプリケーションプラットフォームである．日本では au (KDDI) の EZ アプリに使用されている [15].

15,000 回，3 乗算 30,000 回及び乗法逆元 2,500 回の平均で求め，Duursma-Lee アルゴリズム及び η_T ペアリングは 2000 回の平均で求めた．PC 及び携帯電話の測定は，ともに 12 回測定を行い最大値と最小値を除いた 10 回の平均で求めた．表 4 における単位はミリ秒 (msec) である．

演算	PC	W41T	W41H
加算 (A)	0.0000118	0.00056	0.00049
減算 (A)	0.0000118	0.00057	0.00049
3 乗算 (C)	0.0001625	0.00646	0.00529
乗算 (M)	0.0015839	0.05802	0.03795
乗法逆元 (I)	0.0165765	0.66668	0.43965
Duursma-Lee	2.63	114.81	75.52
η_T ペアリング	1.96	74.61	50.31

表 4: $\mathbb{F}_{3^{97}}$ 上でのペアリングの平均演算速度 (msec)

表 4 の演算速度より，乗算を基準として加算・減算は乗算の約 0.01 倍，3 乗算は約 0.1 倍，乗法逆元は約 10 倍であることが判明した．今回の実測値を基にして見積もりを行うと，Duursma-Lee アルゴリズム，η_T ペアリングの最終冪の演算の割合は，それぞれ 4.03%, 28.26%となった．η_T ペアリングにおいて最終冪 T^S の計算量はメインループの約半分となる．また，各演算のペアリング計算時間に対する割合は，Duursma-Lee アルゴリズムでは，加算・減算 2.78%, 乗算 90.78%, 3 乗算 5.84%, 乗法逆元 0.60% であり，η_T ペアリングでは，加算・減算 3.21%, 乗算 83.14%, 3 乗算 12.18%, 乗法逆元 1.47% となった．

今回の実装による BREW アプリケーションの実行ファイル (*.mod) のサイズは，28,172Byte であった．現在の BREW アプリケーションの平均的なサイズは，300KByte の以下が中心であるため，アプリケーション全体の 1〜2 割程度の大きさとなる．

文献 [14] では，FPGA 上の実装で Duursma-Lee アルゴリズムと η_T ペアリングの速度の差がないことが述べられている．しかし，今回の PC 及び BREW プラットフォーム携帯電話では η_T ペアリングの方が高速であることが示された．

標数 3 の有限体 $\mathbb{F}_{3^{97}}$ 上の超特異曲線を利用した η_T ペアリングの実装結果は，W41T で 74.61 msec, W41H で 50.31 msec の処理速度となり，ペアリングを用いたアプリケーションを携帯電話で実現可能な処理速度を得た．

5 おわりに

本稿では，標数 3 の有限体 $\mathbb{F}_{3^{97}}$ 上の超特異曲線を用いた Duursma-Lee アルゴリズム及び η_T ペアリングを，BREW 携帯電話において実装し，演算速度の比較評価を行った．ペアリングの計算時間全体において $\mathbb{F}_{3^{97}}$ の乗算計算時間が約 80% に達すると判明したため，Comb 法を改良するなど乗算を中心とした高速化を行った．その結果，BREW 携帯電話 W41T，W41H において 100 msec を切る速度にてペアリングが計算可能となった．

これにより，short signature や ID ベース暗号などペアリングを用いた新たな暗号応用技術が，携帯電話の BREW アプリケーションとして実現可能となる．

参考文献

[1] P. Barreto, S. Galbraith, C. O. hEigeartaigh and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties", Cryptology ePrint Archive, Report 2004/375, 2004.

[2] P. Barreto, H. Kim, B. Lynn, and M. Scott, "Efficient Algorithms for Pairing-based Cryptosystems", CRYPTO2002, LNCS 2442, pp.354-368, 2002.

[3] J. Beuchat, M. Shirase, T. Takagi and E. Okamoto, "An Algorithm for the η_T Pairing Calculation in Characteristic Three and its Hardware Implementation", Cryptology ePrint Archive, Report 2006/327, 2006.

[4] D. Boneh and M. Franklin, "Identity Based Encryption from the Weil Pairing", SIAM J. Comput, vol.32, no.3, pp.514-532, 2001.

[5] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing", ASIACRYPT 2001, LNCS 2248, pp.514-532, 2001.

[6] I. Duursma and H.-S. Lee "Tate Pairing Implementation for Hyperelliptic Curves $y^2 = x^p - x + d$", ASIACRYPT 2003, LNCS 2894, pp.111-123, 2003.

[7] R. Granger, D. Page, and M. Stam, "On Small Characteristic Algebraic Tori in Pairing-Based Cryptography", Cryptology ePrint Archive, Report 2004/132, 2004.

[8] P. Granger and D. Page, "Hardware Acceleration of the Tate Pairing in Characteristic Three ", CHES 2005, LNCS 3659, pp.398-411, 2005.

[9] D. Hankerson, A. Menezes and S. Vanstone, *Guide to Elliptic Curve Cryptography*, Springer, 2004.

[10] Y. Kawahara, T. Takagi, and E. Okamoto, "Efficient Implementation of Tate Pairing on a Mobile Phone using Java", Cryptology ePrint Archive, Report 2006/299, 2006.

[11] T. Kerins, W. Marnane, E. Popovici and P. Barreto, "Efficient Hardware for the Tate Pairing Calculation in Characteristic Three", CHES 2005, LNCS 3659, pp.412-426, 2005.

[12] S. Kwon, "Efficient Tate Pairing Computation for Supersingular Elliptic Curves over Binary Fields", Cryptology ePrint Archive, Report 2004/303, 2004.

[13] V. Miller, "Short Programs for Functions on Curves", Unpublished Manuscript, 1986.

[14] C. Shu, S. Kwon and K. Gaj, "FPGA Accelerated Tate Pairing Based Cryptosystems over Binary Fields", Cryptology ePrint Archive, Report 2006/179, 2006.

[15] 茂木 健一, 関根 健三郎, 近藤 治, 白輪地 雄二, 「BREW プログラミング 実践バイブル」, 株式会社インプレス, 2004.