BREW 携帯電話における R-ate ペアリングの実装

井山 政志 † 清本 晋作 ‡ 福島 和英 ‡ 田中 俊昭 ‡ 高木 剛 †

† 公立はこだて未来大学
041-8655 北海道函館市亀田中野町 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

あらまし 携帯電話，センサノードや Smartcard におけるペアリングの実装は数多く報告されている．しかし，これまで高機能な OS を搭載しない一般的な携帯電話において，128 ビット AES のセキュリティレベルにおけるペアリングの実装は報告されていない．標数の大きなペアリングにおいて最も高速なペアリングとして R-ate ペアリングが提案されている．本稿では，128 ビット AES のセキュリティレベルにおける R-ate ペアリングを BREW 携帯電話においてソフトウェア実装を行った．高速化にあたり最終冪に Addition Chain を適用した結果 $\mathbb{F}_p$ の乗算回数を約 8%削減できた．以上のアルゴリズムを実装した結果，BREW 携帯電話（ARM9 225MHz）における R-ate ペアリングの演算時間は 1.60 秒となった．また同じセキュリティレベルにおける BREW 携帯電話上で RSA と ECC との時間比較を行った結果，R-ate ペアリングは RSA と ECC 同等の演算時間であった．

Implementation of the R-ate Pairing using BREW Mobilephones

Tadashi Iyama† Shinsaku Kiyomoto ‡ Kazuhide Fukushima ‡
Toshiaki Tanaka ‡ Tsuyoshi Takagi†

†Future University Hakodate,
116-2, Kamedanakano-cho, Hakodate,
Hokkaido, 041-8655, Japan

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

Abstract Many implementations of pairings on embedded devices such as mobile phones, sensor nodes and smartcards have been proposed. However, pairing at the security level equivalent to 128-bit AES key has not been implemented in mobile phones without high-level OS such as Windows. The R-ate pairing is one of the fastest pairings over large prime fields. In this paper, we implement the R-ate pairing at the security level equivalent to 128-bit AES key on BREW mobile phones. Especially, we speed up the final exponentiation of pairings using addition chain, and the number of multiplications over a prime field can be reduced by about 8%. As a result, the timing of our implementation of the R-ate pairing on ARM 9 225MHz becomes 1.60 seconds, which is comparable to those of RSA and ECC on the same platform.

1 はじめに

ペアリング暗号を利用することにより ID ベース暗号等の従来の公開鍵暗号では実現困難であった暗号プロトコルが多く提案されている．ペアリングの実装アルゴリズムとして，小さな標数を利用した η_T ペアリング [1] や大きな標数を利用した Ate ペアリング [6]，及びその高速化である R-ate ペアリング [10] が知られている．

PC だけでなく携帯電話，センサノードや Smartcard におけるペアリングの実装が数多く報告されている．携帯電話上の実装として川原等 [8] による η_T ペアリングや吉富等 [17, 18] による η_T ペアリングと Ate ペアリングの実装が報告されている．ATmega128L 上の実装として Oliveira 等 [13] や石黒等 [7] による η_T ペアリングや Szczechowiak 等 [16] による Ate ペアリングの実装が報告されている．また Smartcard 上の実装として Scott 等 [15] による Ate ペアリングの実装が報告されている．上記の実装は 80 ビット AES（RSA 1024 ビット，ECC 160 ビット）のセキュリティレベル [9] である．一方，128 ビット AES（RSA 3072 ビット，ECC 256 ビット）のセキュリティレベルにおけるペアリングの実装としては宮崎等 [11] による ATmega128L 上の η_T ペアリングの実装が報告されている．また Smartcard 上の実装として Devegili 等 [4] による Ate ペアリングの実装が報告されている．しかし，これまで GMP 等の多倍長ライブラリを利用できる Windows のような高機能な OS を搭載しない一般的な携帯電話において，128 ビット AES のセキュリティレベルにおけ

るペアリングの実装は報告されていない．

本稿では 128 ビット AES のセキュリティレベルにおけるペアリングを BREW 携帯電話においてソフトウェア実装する．Ate ペアリングにおいて Miller ループと最終冪のそれぞれが計算量の多い演算となっている．R-ate ペアリングは Ate ペアリングの Miller ループを半分にすることができる．最終冪において Devegili 等 [4] は最終冪に現れる冪を分解し高速に計算する手法を提案し，Scott 等 [14] は Addition Chain を用いた高速化を提案した．最終冪の演算時間は BN 曲線のパラメータを決める整数値 z の Hamming Weight に依存する．そのため本稿では Devegili 等 [4] の $z = 2^{62} + 2^{61} + 7981$ における Ate ペアリングと R-ate ペアリングにおいて，上記の最終冪に別の Addition Chain を用いて更なる高速化を提案する．特に z，$6z$，$6z + 5$ の冪乗算において提案 Addition Chain を用いることにより上記の従来方式より $\mathbb{F}_p$ の乗算回数を約 8%削減できた．

更に，提案方式を用いた Ate ペアリングと R-ate ペアリングを BREW 携帯電話に実装した．本稿においてペアリングの実装に使用した多倍長ライブラリは吉富等 [18] と同様のライブラリを使用し，有限体の構成は Devegili 等 [4] と同じものを利用した．ARM9 225MHz における R-ate ペアリングの演算時間は 1.60 秒となり，Ate ペアリングは 2.44 秒となった．これは R-ate ペアリングにおける Miller ループの演算時間が全体の約 50%を占めているため，R-ate ペアリングの約 1.5 倍の時間となる．また従来の公開鍵暗号との比較も行った．ARM9 225MHz における 3072 ビット RSA の演算時間はは 7.51 秒となり，256 ビット ECC の演算時間は 0.98 秒となった．この結果から R-ate ペアリングは RSA と ECC と同等の時間で実装できることがわかった．

2 12 次拡大体と BN 曲線

本章では R-ate ペアリングの実装に必要な 12 次拡大体の構成と BN 曲線について述べる．

2.1 12 次拡大体

12 次拡大体は 2 次拡大体と 6 次拡大体から構成される．基礎体を $\mathbb{F}_p$ とするとそれぞれの拡大体の構成は Devegili 等 [4] と同様に以下のように表現する．

$$\begin{aligned}\mathbb{F}_{p^2} &= \mathbb{F}_p[u]/(u^2+2)\\ \mathbb{F}_{p^6} &= \mathbb{F}_{p^2}[v]/(v^3-\xi)\\ \mathbb{F}_{p^{12}} &= \mathbb{F}_{p^6}[w]/(w^2-v)\\ &= \mathbb{F}_{p^2}[W]/(W^6-\xi)\end{aligned}$$

ここで $\xi = -u - 1 \in \mathbb{F}_{p^2}$ である．標数 p が $p \equiv 7 \pmod 8$ の場合 -2 は $\mathbb{F}_p$ で平方非剰余となり，u^2+2 が $\mathbb{F}_p[u]$ で既約となる．また $p \equiv 1 \pmod 6$ の場合 ξ は $\mathbb{F}_{p^2}$ で平方非剰余かつ 3 乗非剰余となり，$W^6 - \xi$ が $\mathbb{F}_{p^2}[W]$ で既約となる．$\alpha \in \mathbb{F}_{p^{12}}$ を以下の 3 通りで表し，実装アルゴリズムにより使い分ける．

$$\begin{aligned}\alpha &= a_0 + a_1 w \qquad a_0, a_1 \in \mathbb{F}_{p^6}\\ &= (a_{0,0} + a_{0,1}v + a_{0,2}v^2)\\ &\quad +(a_{1,0} + a_{1,1}v + a_{1,2}v^2)w \quad a_{i,j} \in \mathbb{F}_{p^2}\\ &= a_{0,0} + a_{1,0}W + a_{0,1}W^2\\ &\quad +a_{1,1}W^3 + a_{0,2}W^4 + a_{1,2}W^5\end{aligned}$$

12 次拡大体の演算は加算，減算，乗算，2 乗算，逆元算で構成され，5 つの演算は $\mathbb{F}_p$ の演算で構成されている．乗算と 2 乗算は Karatsuba 法とそれを 2 乗算に適用した方法を用いて計算する [3]．

2.2 BN 曲線

本稿では Barreto-Naehrig（BN）曲線 [2]

$$E(\mathbb{F}_p) = \{(x, y) \in \mathbb{F}_p \times \mathbb{F}_p \mid y^2 = x^3 + b\} \cup \{\infty\}$$

を用いる．ここで $b \in \mathbb{F}_p$，∞ は無限遠点である．また BN 曲線のパラメータは

$$\begin{aligned}p &= 36z^4 + 36z^3 + 24z^2 + 6z + 1\\ r &= 36z^4 + 36z^3 + 18z^2 + 6z + 1\\ t &= 6z^2 + 1\end{aligned}$$

となる．ここで $z \in \mathbb{Z}$，p と r は素数であり，$\#E(\mathbb{F}_p) = r$ となる [2]．t は $\pi : (x, y) \mapsto (x^p, y^p)$ となる p 乗 Frobenius 写像 π のトレースである．そして BN 曲線の埋め込み次数は 12 である．埋め込み次数とは $r \mid p^k - 1$ を満たす最小の正の整数 k のことである．そして本稿では Devegili 等 [4] によって示されている $z = 2^{62} + 2^{61} + 7981$ と $b = 3$ を使用する．

この $E(\mathbb{F}_p)$ は以下の 6 次のツイストを持つ．

$$E'(\mathbb{F}_{p^2}) = \{(x, y) \in \mathbb{F}_{p^2} \times \mathbb{F}_{p^2} \mid y^2 = x^3 + b/\xi\} \cup \{\infty\}$$

そして $E'(\mathbb{F}_{p^2})$ の位数は $\#E'(\mathbb{F}_{p^2}) = r(2p - r)$ であり，位数が r となる $E'(\mathbb{F}_{p^2})$ の部分群 $E'(\mathbb{F}_{p^2})[r]$ が存在する．$W^6 - \xi$ の根を W' としたとき $(x, y) \mapsto (xW'^2, yW'^3)$ で定義される準同型写像が存在する．

Devegili 等のパラメータを利用することで p が 256 ビット，r が 256 ビットとなり，$\mathbb{F}_{p^{12}}$ のビット長は 3067 ビットとなる．これは 128 ビット AES のセキュリティレベルをほぼ満たす．

3 BN 曲線上のペアリング

本章では BN 曲線上の Ate ペアリングと R-ate ペアリングについて説明する．

3.1 Ate ペアリング

Ate ペアリングは 2006 年に Hess 等によって提案された [6]．$P \in E'(\mathbb{F}_{p^2})$，$Q \in E(\mathbb{F}_p)$ に対して Ate ペアリング $e_A(P, Q) \in \mathbb{F}_{p^{12}}$ を次により定義する．

$$e_A(P, Q) = f_{6z^2, P}(Q)^{(p^{12}-1)/r}$$

ここで $f_{6z^2,P}$ は Miller 関数とする [5].

Algorithm 1 Ate ペアリング [5]

INPUT: $P \in E'(\mathbb{F}_{p^2})[r], Q \in E(\mathbb{F}_p), 6z^2 \in \mathbb{Z}$
OUTPUT: $e_A(P,Q) \in \mathbb{F}_{p^{12}}$
1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i \leftarrow \lfloor \log_2(6z^2) \rfloor - 2$ to 0 **do**
4: $T \leftarrow 2T$
5: $f \leftarrow f^2 \cdot l_{T,T}(Q)$
6: **if** $(6z^2)_i = 1$ **then**
7: $T \leftarrow T + P$
8: $f \leftarrow f \cdot l_{T,P}(Q)$
9: **end if**
10: **end for**
11: $f \leftarrow f^{(p^{12}-1)/r}$
12: **return** f

Algorithm 1 のステップ 3〜10 のループが Miller ループであり，その回数は $\lfloor \log_2(6z^2) \rfloor - 1$ である．また整数 a の i 番目のビットを $(a)_i$ とする．ここで $6z^2 \approx \sqrt{p}$ である．そしてステップ 11 の冪乗算を最終冪と呼ぶ．

3.2 R-ate ペアリング

R-ate ペアリングは 2008 年に Lee 等によって Ate ペアリングを一般化したペアリングとして提案された [10]．$P \in E'(\mathbb{F}_{p^2})$，$Q \in E(\mathbb{F}_p)$ に対して R-ate ペアリング $e_R(P,Q) \in \mathbb{F}_{p^{12}}$ を次により定義する．

$$\begin{aligned} & e_R(P,Q) \\ = \; & (f_{6z+2,P}(Q) \cdot (f_{6z+2,P}(Q) \cdot l_{(6z+2)P,P}(Q))^p \\ & \cdot l_{\pi((6z+3)P),(6z+2)P}(Q))^{(p^{12}-1)/r} \end{aligned}$$

ここで $l_{A,B}$ は A と B を通る直線である．

Algorithm 2 R-ate ペアリング [5]

INPUT: $P \in E'(\mathbb{F}_{p^2})[r], Q \in E(\mathbb{F}_p), 6z+2 \in \mathbb{Z}$
OUTPUT: $e_R(P,Q) \in \mathbb{F}_{p^{12}}$
1: $T \leftarrow P$
2: $f \leftarrow 1$
3: **for** $i \leftarrow \lfloor \log_2(6z+2) \rfloor - 2$ to 0 **do**
4: $T \leftarrow 2T$
5: $f \leftarrow f^2 \cdot l_{T,T}(Q)$
6: **if** $(6z+2)_i = 1$ **then**
7: $T \leftarrow T + P$
8: $f \leftarrow f \cdot l_{T,P}(Q)$
9: **end if**
10: **end for**
11: $f \leftarrow f \cdot (f \cdot l_{T,P}(Q))^p \cdot l_{\pi(T+P),T}(Q)$
12: $f \leftarrow f^{(p^{12}-1)/r}$
13: **return** f

Algorithm 2 のステップ 3〜10 の Miller ループの回数は $\lfloor \log_2(6z+2) \rfloor - 1$ である．$6z+2 \approx \sqrt{6z^2} \approx \sqrt[4]{p}$ なので，Ate ペアリングのおよそ半分のループ回数となる．ステップ 11 では Miller ループによって求めた f と $(6z+2)P$ を用いて計算している．そしてステップ 12 の最終冪に現れる冪が Ate ペアリングと同じ冪であることに注意する．

3.3 ヤコビアン座標における演算

Algorithm 1，2 においてアフィン座標の点 (x,y) に対応するヤコビアン座標の点 (X,Y,Z) を用いる [5]．ここで $x = X/Z^2, y = Y/Z^3$ である．ステップ 1 において $P = (x,y)$ を $T = (x,y,1)$ に初期化する．ステップ 4 においてヤコビアン座標における 2 倍算 $2T$ を行う．$T = (X,Y,Z), 2T = (X_3,Y_3,Z_3)$ とすると，

$$\begin{aligned} X_3 &= 9X^4 - 8XY^2 \\ Y_3 &= 3X^2(4XY^2 - X_3) - 8Y^4 \\ Z_3 &= 2YZ \end{aligned}$$

となる．そしてステップ 5 において T の接線 $l_{T,T}(Q)$ を求める．$Q = (x,y)$ とし 2.1 節のように $\mathbb{F}_{p^{12}} = \mathbb{F}_{p^2}[W]/(W^6 - \xi)$ とすると，$l_{T,T}(Q) = Z_3Z^2y - 2Y^2W^3 - 3X^2W(Z^2x - XW^2) \in \mathbb{F}_{p^{12}}$ となる．

ステップ 7 においてヤコビアン座標とアフィン座標の点の加算 $T+P$ を行い，ステップ 8 において T と P を通る直線 $l_{T,P}(Q)$ を求める．$T = (X_1,Y_1,Z_1), P = (X_2,Y_2), T+P = (X_3,Y_3,Z_3)$ とすると，

$$\begin{aligned} X_3 &= (Y_2Z_1^3 - Y_2)^2 - (X_2Z_1^2 - X_1)^2(X_1 + X_2Z_1^2) \\ Y_3 &= (Y_2Z_1^3 - Y_1)[X_1(X_2Z_1^2 - X_1)^2 - X_3] \\ &\quad -Y_1(X_2Z_1^2 - X_1)^3 \\ Z_3 &= (X_2Z_1^2 - X_1)Z_1 \end{aligned}$$

そして $l_{T,P}(Q) = (y - Y_2W^3)Z_3 - (Y_2Z_1^3 - Y_1)W(x - X_2W^2) \in \mathbb{F}_{p^{12}}$ となる．

Algorithm 2 のステップ 11 においてループによって求めた $f, (6z+2)P$ を用いる．ここでは $(6z+2)P + P$ を計算し，$l_{(6z+2)P,P}(Q)$ を求める．そして，$(6z+2)P$ をアフィン座標の点に変換し，$l_{\pi((6z+3)P),(6z+2)P}(Q)$ を求める．

3.4 最終冪

Devegili 等は最終冪における冪 $(p^{12}-1)/r$ を (p^6-1)，(p^2+1) と $(p^4-p^2+1)/r$ の 3 つに分解して高速に計算する方法を提案している [4]．(p^6-1) 乗は

$$f^{p^6-1} = \frac{f_0 - f_1w}{f_0 + f_1w} \quad f_0, f_1 \in \mathbb{F}_{p^6}$$

となり，$\mathbb{F}_{p^{12}}$ の逆元算 1 回と乗算 1 回で計算できる．そして $f_{i,j} \in \mathbb{F}_{p^2}$ $(i,j = \{0,1,2\})$ とすると，p 乗は

$$\begin{aligned} f^p &= (f_{0,0}^p + (\gamma_2 \cdot f_{0,1}^p)v + (\gamma_4 \cdot f_{0,2}^p)v^2) \\ &\quad +(\gamma_1 \cdot f_{1,0}^p + (\gamma_3 \cdot f_{1,1}^p)v + (\gamma_5 \cdot f_{1,2}^p)v^2)w \end{aligned}$$

となることを利用して (p^2+1) 乗を求める．ここで $\gamma_i = \xi^{i(p-1)/6}$ $(i = \{1,2,3,4,5\})$ であり事前計算が可能である．また $f_{i,j}^p$ は $\mathbb{F}_{p^2}$ の共役をとることで計算できる．そのため p 乗は $\mathbb{F}_{p^2}$ の乗算 5 回で計算できる．

$(p^4-p^2+1)/r$ の冪乗算（Hard Exp.）の方法は Devegili 等が提案した方法 [4] と Scott 等の提案した方法 [14] がある．Devegili 等は以下のアルゴリズムで計算している．

Algorithm 3 Devegili 等の Hard Exp. [4]

INPUT: $f \in \mathbb{F}_{p^{12}}$, p, r, $z \in \mathbb{Z}$
OUTPUT: $f^{(p^4-p^2+1)/r} \in \mathbb{F}_{p^{12}}$
1: $a \leftarrow f^{-(6z+5)}, b \leftarrow a^p, b \leftarrow a \cdot b$
2: $f_1 \leftarrow f^p$, $f_2 \leftarrow f^{p^2}$, $f_3 \leftarrow f^{p^3}$
3: $f \leftarrow f_3 \cdot [b \cdot (f_1)^2 \cdot f_2]^{6z^2+1} \cdot b \cdot (f_1 \cdot f)^9 \cdot a \cdot f^4$
4: **return** f

Scott 等の提案方法では事前に f^{-z}，f^{z^2}，f^{-z^3}，f^p，f^{p^2}，f^{p^3}，$(f^{-z})^p$，$(f^{z^2})^p$，$(f^{-z^3})^p$ と $(f^{z^2})^{p^2}$ を計算し，以下の y_0〜y_6 を求める．

$$
\begin{aligned}
y_0 &= f^p \cdot f^{p^2} \cdot f^{p^3},\ y_1 = 1/f,\ y_2 = (f^{z^2})^{p^2},\\
y_3 &= (f^{-z})^p,\ y_4 = f^{-z}/(f^{z^2})^p,\ y_5 = 1/f^{z^2},\\
y_6 &= f^{-z^3}/(f^{-z^3})^p
\end{aligned}
$$

ここで逆元算は $\mathbb{F}_{p^{12}}$ の共役をとることで計算できる．そして上記の y_0〜y_6 を用いると $(p^4-p^2+1)/r$ 乗した値は $y_0 \cdot y_1^2 \cdot y_2^6 \cdot y_3^{12} \cdot y_4^{18} \cdot y_5^{30} \cdot y_6^{36}$ と同じである．そしてこの演算を以下のアルゴリズムで求める．

Algorithm 4 Scott 等の Hard Exp. [14]

INPUT: $y_0, y_1, y_2, y_3, y_4, y_5, y_6 \in \mathbb{F}_{p^{12}}$
OUTPUT: $y_0 \cdot y_1^2 \cdot y_2^6 \cdot y_3^{12} \cdot y_4^{18} \cdot y_5^{30} \cdot y_6^{36} \in \mathbb{F}_{p^{12}}$
1: $T_0 \leftarrow y_6^2 \cdot y_4 \cdot y_5$, $T_1 \leftarrow y_3 \cdot y_5 \cdot T_0$
2: $T_0 \leftarrow T_0 \cdot y_2$, $T_1 \leftarrow (T_1^2 \cdot T_0)^2$
3: $T_0 \leftarrow T_1 \cdot y_1$, $T_1 \leftarrow T_1 \cdot y_0$
4: $T_0 \leftarrow T_0^2 \cdot T_1$
5: **return** T_0

4 提案高速化法

本章では Devegili 等と Scott 等の Hard Exp. の高速化手法について述べる．また z は Devegili 等 [4] の $z = 2^{62} + 2^{61} + 7981$ である．

4.1 冪乗算の高速化

Devegili 等の Hard Exp. では $6z+5$ と $6z^2+1$ の冪乗算，Scott 等の Hard Exp. では z の冪乗算を行っている．それぞれの冪乗算を Addition Chain を用いて高速化を行う．

$6z+5$ を 2 進展開した場合 1001 と 1011 のビットの並びが現れるので f^9 と f^{11} を事前計算し，2 乗算と乗算を組み合わせて計算する．

Algorithm 5 提案 Addition Chain (f^{6z+5})

INPUT: $f \in \mathbb{F}_{p^{12}}$
OUTPUT: $f^{6z+5} \in \mathbb{F}_{p^{12}}$
1: $f_1 \leftarrow f$, $f_2 \leftarrow f^9$, $f_3 \leftarrow f^{11}$
2: $f \leftarrow \left([(f_2^{2^{50}} \cdot f_3)^{2^4} \cdot f_3]^{2^7} \cdot f_2\right)^2 \cdot f_1$
3: **return** f

square-and-multiply 法で f^{6z+5} を計算する場合 $\mathbb{F}_{p^{12}}$ の乗算回数は 10 回であるが，Algorithm 5 を適用した場合 $\mathbb{F}_{p^{12}}$ の乗算回数を 6 回にすることができる．

$6z^2+1$ は z と $6z$ を分けて冪乗算を行う [5]．z では 101，111 と 1100 のビットの並びが現れるので f^5，f^7 と f^{12} を事前計算する．$6z$ では 1001，1011 と 1110 のビットの並びが現れるので f^9，f^{11} と f^{14} を事前計算する．

Algorithm 6 提案 Addition Chain (f^z)

INPUT: $f \in \mathbb{F}_{p^{12}}$
OUTPUT: $f^z \in \mathbb{F}_{p^{12}}$
1: $f_1 \leftarrow f^5$, $f_2 \leftarrow f^7$, $f_3 \leftarrow f^{12}$
2: $f \leftarrow \left([(f_3^{2^{49}} \cdot f_2)^{2^4} \cdot f_3]^{2^3} \cdot f_1\right)^{2^3} \cdot f_1$
3: **return** f

Algorithm 7 提案 Addition Chain (f^{6z})

INPUT: $f \in \mathbb{F}_{p^{12}}$
OUTPUT: $f^{6z} \in \mathbb{F}_{p^{12}}$
1: $f_1 \leftarrow f^9$, $f_2 \leftarrow f^{11}$, $f_3 \leftarrow f^{14}$
2: $f \leftarrow [(f_1^{2^{50}} \cdot f_2)^{2^4} \cdot f_2]^{2^8} \cdot f_3$
3: **return** f

square-and-multiply 法で f^z と f^{6z} を計算する場合 $\mathbb{F}_{p^{12}}$ の乗算回数はそれぞれ 10 回であるが，Algorithm 6 と 7 を適用した場合 $\mathbb{F}_{p^{12}}$ の乗算回数を 7 回にすることができる．

4.2 従来方式との比較

Hankerson 等 [5] は Devegili 等の提案した最終冪における $\mathbb{F}_p$ の乗算回数は 7246 回であると述べている．そして 4.1 節の提案手法を適用した場合 $\mathbb{F}_p$ の乗算回数は 6653 回となり，約 8%削減している結果となった．

一方 Scott 等の提案した最終冪における $\mathbb{F}_p$ の乗算回数は 7046 である．そして 4.1 節の提案手法を適用した結果 $\mathbb{F}_p$ の乗算回数は 6560 回となり，約 7%削減している結果となった．

4.3 z の選択に関する考察

Miller ループと最終冪の演算時間は z の Hamming Weight（HW(z)）に依存している．本稿で用いる z における Ate ペアリングと R-ate ペアリングの Miller ループの回数を決める $6z^2$ と $6z+2$ のビット長とその Hamming Weight を表 1 に示す．

表 1: $6z^2, 6z+2$ のビット長とその Hamming Weight

	Ate($6z^2$)	R-ate($6z+2$)
ビット長	128	66
Hamming Weight	28	9

(ビット長 -1) がループの回数であり，点の倍算を行う回数である．また (Hamming Weight-1) が点の加算を行う回数である．本節ではペアリングを高速にするため，p と r が素数となり HW(z) が小さい z を選択することについて考える．しかし $z = \pm(2^{63} + \cdots)$ の場合 p が 258 ビットと大きくなるので $z = \pm(2^{62} + \cdots)$ について述べる．

p と r が素数となり HW(z) $= 1, 2$ となる z は存在しないが，HW(z) $= 3$ の場合，$z = -(2^{62}+2^{55}+1), -(2^{62}+2^{35}+2^{24}), -(2^{62}+2^{46}+2^{29}), 2^{62}+2^{41}+2^{23}$ の 4 個存在する．これらの z を用いた場合，-2 が $\mathbb{F}_p$ で平方剰余になるため拡大体の構成を変更しなくてはならない．また p と r が 254 ビット，$\mathbb{F}_{p^{12}}$ のビット長が 3039 ビットに低下してしまう．

p と r が 256 ビットの素数となる z の場合，HW(z) $= 4$ が最小であり $z = -(2^{62}+2^{61}+2^{41}+2^8), -(2^{62}+2^{61}+2^{18}+2^{13})$ の 2 個存在する．これらの z も -2 が $\mathbb{F}_p$ で平方剰余になるため拡大体の構成を変更しなくてはならない．

p と r が 256 ビットの素数で -2 が $\mathbb{F}_p$ で平方非剰余となる z の場合，HW(z) $= 5$ が最小であり $z = -(2^{62}+2^{61}+2^{54}+2^{53}+2), 2^{62}+2^{61}+2^{53}+2^{45}+2, 2^{62}+2^{61}+2^{57}+2^{55}+2$ の 3 個存在する．これらの z を用いた場合，Devegili 等の $z = 2^{62} + 2^{61} + 7981$ に本章の提案 Addition Chain を用いた最終冪より 8%程度高速になると予想される．

5 実装結果

本章では Ate ペアリングと R-ate ペアリングを BREW 携帯電話に実装した結果を示す．

5.1 開発環境

本稿では BREW 携帯電話で用いられる ARM9 プロセッサにおいて実装を行う．BREW とは Binary Runtime Environment for Wireless の略で，CDMA 対応携帯電話向けのプラットフォームである．BREW の特徴は省電力であることやコンパイラ型なので処理速度が高速なことである [12]．今回使用するプロセッサは ARM9 150MHz プロセッサ及び，ARM9 225MHz プロセッサである．

開発は BREW SDK 3.1.2，Microsoft Visual C++ 6.0 を用いて実装を行う．コンパイラは 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（RVCT 1.2）を使用する．そして，コンパイルによって作成された実行ファイル（*.mod）を転送ケーブルで BREW 携帯電話に転送し，演算時間の評価を行う．

5.2 ペアリングの実装結果

Ate ペアリングと R-ate ペアリングでは有限体 $\mathbb{F}_p$ 及び，その拡大体 $\mathbb{F}_{p^2}$，$\mathbb{F}_{p^6}$，$\mathbb{F}_{p^{12}}$ の演算を用いる．各ペアリングはすべて有限体の演算により計算することができる．有限体 $\mathbb{F}_p$ の各演算時間を表 2 に示す．

表 2: $\mathbb{F}_p$ の各演算時間（msec）

	ARM9 150MHz	ARM9 225MHz
加算	0.0114	0.0094
減算	0.0062	0.0039
乗算	0.1205	0.0799
逆元算	0.9910	0.6549

$\mathbb{F}_p$ の加算は減算のおよそ 2 倍である．これは減算は加算に比べ p で剰余をとる頻度が少ないからである．また乗算は加算のおよそ 10 倍である．

$z = 2^{62} + 2^{61} + 7981$ に対して Scott 等の最終冪に提案 Addition Chain を適用した Ate ペアリングと R-ate ペアリング 1 回の計算に必要な $\mathbb{F}_p$ の演算回数を表 3 と表 4 に示す．

表 3: Ate ペアリングに必要な $\mathbb{F}_p$ の各演算回数

	全体	Miller 関数	最終冪
加算	56623	36049	20574
減算	41855	27517	14338
乗算	22282	15722	6560
逆元算	1	0	1

表 4: R-ate ペアリングに必要な $\mathbb{F}_p$ の各演算回数

	全体	Miller 関数	最終冪
加算	38510	17936	20574
減算	27997	13659	14338
乗算	14395	7835	6560
逆元算	2	1	1

上記の結果から ARM9 225MHz における Ate ペアリングと R-ate ペアリングの見積もり時間と実測値を表 5 に示す．

表 5: ARM9 225MHz における Ate/R-ate ペアリングの演算時間（sec）

	見積もり時間	実測値
Ate ペアリング	2.48	2.44
R-ate ペアリング	1.62	1.60

R-ate ペアリングの見積もり時間である 1.62 秒のうち，$\mathbb{F}_p$ の乗算の時間はおよそ 1.15 秒となり全体の約 70%を占めている．また加算はおよそ 22%，減算はおよそ 7%，逆元算は 1%未満である．

5.3 RSA，ECC との実装比較

本稿では，128 ビット AES のセキュリティレベルにおける Ate ペアリング，R-ate ペアリング，RSA と ECC の時間比較を表 6 に示す．RSA は 3072 ビット整数の冪乗剰余演算，ECC は 256 ビット整数のスカラー倍算の演算時間である．なお ECC における楕円曲線の有限体は，p が 256 ビットの $\mathbb{F}_p$ 及び $\mathbb{F}_{2^{283}}$ である．

表 6: 128 ビット AES のセキュリティレベルにおける公開鍵暗号の各演算時間 (sec)

	ARM9 150MHz	ARM9 225MHz
Ate ペアリング	3.68	2.44
R-ate ペアリング	2.43	1.60
RSA 暗号	10.0	7.51
ECC($\mathbb{F}_p$)	1.68	0.98
ECC($\mathbb{F}_{2^{283}}$)	5.48	3.80

表 6 より ARM9 225MHz において Ate ペアリングは 2.44 秒，RSA は 7.51 秒となり Ate ペアリングは RSA より約 3.1 倍高速であった．さらに R-ate ペアリングは 1.60 秒となり RSA より約 4.7 倍，Ate ペアリングより約 1.5 倍高速であった．ECC の実装に関して標数の大きな有限体 $\mathbb{F}_p$ を利用した実装が 0.98 秒であり，標数の小さな有限体 $\mathbb{F}_{2^{283}}$ を利用した実装が 3.80 秒となり標数の大きな有限体を利用したほうが約 3.8 倍高速となった．そして標数の大きな有限体を利用した ECC は R-ate ペアリングよりも約 1.6 倍高速となった．

80 ビット AES のセキュリティレベルにおいて，吉富等は RSA，標数の大きな ECC，標数の小さな ECC，そして Ate ペアリングの順に高速であると報告している [18]．しかし 128 ビット AES のセキュリティレベルにおいて，標数の大きな ECC，ペアリング，標数の小さな ECC，そして RSA の順に高速である結果となった．

6 おわりに

本稿では 128 ビット AES のセキュリティレベルにおける BN 曲線上の Ate ペアリングと R-ate ペアリングを BREW 携帯電話に実装した．Devegili 等と Scott 等の提案した最終冪の計算方法に提案 Addition Chain を適用し，高速化を行った．最終冪において Devegili 等の乗算回数は 7246 回であったが，Scott 等の最終冪に提案 Addition Chain を適用し，z の冪乗算の乗算回数を削減することで $\mathbb{F}_p$ の乗算回数を 6560 回に削減することができた．ARM9 225MHz において R-ate ペアリングの演算時間は 1.60 秒となった．

また同じセキュリティレベルにおけるペアリングと RSA，ECC との時間比較を行った．比較から，RSA と標数の小さな有限体 $\mathbb{F}_{2^{283}}$ を利用した ECC より Ate ペアリングと R-ate ペアリングの方が高速であるが，標数の大きな有限体 $\mathbb{F}_p$ を利用した ECC の方がペアリングより高速であった．上記から R-ate ペアリングは従来の公開鍵暗号と同等の時間で計算可能という結果を得た．

参考文献

[1] P. Barreto, S. Galbraith, C. Ó'hÉigeartaigh and M. Scott, "Efficient Pairing Computation on Supersingular Abelian Varieties", Designs, Codes and Cryptography, vol.42, no.3, pp.239-271, 2007.

[2] P. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order", SAC 2005, LNCS 3897, pp.319-331, 2006.

[3] A. Devegili, C. Ó'hÉigeartaigh, M. Scott and R.Dahab, "Multiplication and Squaring on Pairing-Friendly Fields", Cryptography ePrint Archive, Report 2006/471, 2006.

[4] A. Devegili, M. Scott and R.Dahab, "Implementing Cryptographic Pairings over Barreto-Naehrig Curves", Pairing 2007, LNCS 4575, pp.197-207, 2007.

[5] D. Hankerson, A. Menezes and M. Scott, "Software Implementation of Pairings", Identity-Based Cryptography, pp.188-206, IOS Press, 2009.

[6] F. Hess, N. Smart and F. Vercauteren, "The Eta Pairing Revisited", IEEE Transactions on Information Theory, vol.52, no.10, pp.4595-4602, 2006.

[7] 石黒司，白勢政明，高木剛，"ATmega128L 上でのペアリング暗号の高速実装", 情報処理学会論文誌, vol.49, no.11, pp.3743-3753, 2008.

[8] 川原祐人，高木剛，岡本栄司，"Java を利用した携帯電話上での Tate ペアリングの高速実装", 情報処理学会論文誌, vol.49, no.1, pp.427-435, 2008.

[9] N. Koblitz and A. Menezes, "Pairing-Based Cryptography at High Security Levels", Cryptography and Coding 2005, LNCS 3796, pp.13-36, 2005.

[10] E. Lee, H.-S. Lee and C.-M. Park, "Efficient and Generalized Pairing Computation on Abelian Varieties", IEEE Transactions on Information Theory, vol.55, no.4, pp.1793-1803, 2009.

[11] Y. Miyazaki, M. Shirase and T.suyoshi Takagi, "Elliptic Curve/Pairing-based Cryptographies on MICAz With Some Security Levels", WISA 2009, Short Presentation Track, pp.183-195, 2009.

[12] 溝口英巳，宮北幸典，所友太，"詳細 EZ アプリ（BREW）プログラミング", 株式会社リックテレコム, 2007.

[13] L. Oliveira, M. Scott, J. Lopez and R. Dahab, "TinyPBC: Pairings for Authenticated Identity-Based Non-Interactive Key Distribution in Sensor Networks", INSS 2008, 17-19, pp.173-180, 2008.

[14] M. Scott, N. Benger and M. Charlemagne, "On the Final Exponentiation for Calculating Pairings on Ordinary Elliptic Curves", Pairing 2009, LNCS 5671, pp.78-88, 2009.

[15] M. Scott, N. Costigan and W. Abdulwahab, "Implementing Cryptographic Pairings on Smartcards", CHES 2002, LNCS 2523, pp.159-174, 2003.

[16] P. Szczechowiak, L. Oliveira, M. Scott, M. Collier and R. Dahab, "NanoECC: Testing the Limits of Elliptic Curve Cryptography in Sensor Networks", EWSN 2008, LNCS 4249, pp.134-147, 2006.

[17] M. Yoshitomi, T. Takagi, S. Kiyomoto and T. Tanaka, "Efficient Implementation of the Pairing on Mobilephones using BREW", IEICE Transaction, vol.E91-D, no.5, pp.1330-1337, 2008.

[18] 吉富基, 清本晋作, 福島和英, 田中俊昭, 高木剛, "BREW 携帯電話における通常楕円曲線上のペアリングの実装", 2009 年暗号と情報セキュリティシンポジウム, SCIS 2009, 3C2-2, 2009.