$\mathbb{F}_{2^m}$ 上の楕円曲線ペアリングの Java 実装

張　一凡 †　　金山　直樹 †　　岡本　栄司 †

† 筑波大学システム情報工学研究科リスク工学専攻
305-5873 つくば市天王台 1-1-1

tyouiifan@cipher.risk.tsukuba.ac.jp

あらまし　本研究では標数 2 の有限体 $\mathbb{F}_{2^m}$ 上のペアリングライブラリを Java 言語にて実装し，その性能評価を行う．標数 3 の有限体でのペアリング実装例は存在するが標数 2 の場合では今のところ実装例が知られていない．提案するライブラリではペアリングを容易かつ高速に実装できるように改良し，ペアリングプロトコルの実装利用や実験での利用で活用できることを目指している．また，ライブラリの構築に当たり理論的評価のみでなく実働時間において汎用計算機で計算されるペアリ ングアルゴリズムの有効性を評価した．

Java implementation of pairing on elliptic curves over $\mathbb{F}_{2^m}$

Yifan Zhang†　　Naoki Kanayama†　　Eiji Okamoto†

†Department of Risk Engineering,　University of Tsukuba
1-1,　Tenoudai,　Tsukuba,　Ibaraki,　305-5873 Japan

tyouiifan@cipher.risk.tsukuba.ac.jp

Abstract　In this paper, the authors implemented libraries of some pairing functions on elliptic curves over binary fields by Java language and carried out a performance test of them.

1　まえがき

近年情報セキュリティ分野ではペアリング暗号が次世代暗号として注目を浴びるようになってきている．ショート署名方式 [4] や ID ベース暗号 [3] などが，ペアリング暗号の代表的なプロトコルとしてあげられる．これらはペアリング写像の持つ双線形性という特性を用いて処理を行っており，電子署名のデータサイズの減少，公開鍵の管理コストの削減などの面で既存の暗号方式では構成が困難だった問題を解決している．

今日，Java 言語は普及度の高いプログラミング言語となって，その仕様と解読の容易さからプログラムを組む時に Java 言語を選ぶ人が増えてきている．しかし残念ながらペアリングは，例えば RSA 暗号などのような既存方式に比べ構造が簡単でないため実装が容易でなく，Java 言語による公開ライブラリは現在著者らの知る限り，いまのところ提供されていない．

Java 言語でのペアリングライブラリが存在すれば，Java 上でのペアリングを使用したライブラリの開発，Java 組込みマシンでのペアリング計算が可能となるため，ペアリング暗号の利便性の向上が期待される．さらに，Java でライブラリを組むことによって，その解読性から再構築にも利用にも容易なライブラリを構築できる．

このような状況を踏まえ，本研究では Java 言語を用いて簡単なペアリングライブラリを構築し，その性能評価を行う．

1.1 目的

Java 言語でペアリングのライブラリを構築するにあたり，まずはペアリング計算の基礎となる有限体演算と楕円曲線上の加算 (後述) を扱えるクラスを作成する必要がある．C や C++などの言語と違い，Java 言語にはこれらの演算を扱えるライブラリが知られていないため，一番低いレベルである有限体演算から全て自作する．

本研究では η_T ペアリングを実装し，理論面および実働時間面から性能評価を行う．

1.2 既存研究

既存の類似研究では川原らの Java における標数 3 の有限体上の楕円曲線のペアリング実装 [6] があげられる．これに対して，本研究では汎用 CPU 向きといわれる標数 2 での初の Java によるペアリング実装を行う．

楕円曲線の選択の一つの目安となる埋め込み次数 (§2.3 参照) についてであるが，我々が用いる楕円曲線の埋め込み次数は 4 である．これは川原らの実装で用いている楕円曲線の次数 6 より小さい．ショート署名等で使う署名長は，埋め込み次数 k の楕円曲線を使うと，従来方式と同じ程度の安全性を保つために必要な署名長が従来方式のおおよそ $1/k$ になると言われている．これは例えば 1024 ビットの出力を得るのに標数 3 の楕円曲線を用いる場合は 194 ビット程度でよいのに対し，標数 2 の楕円曲線の場合だと 256 ビット程度のサイズになることを意味する．このため，ペアリングの基礎体のサイズの点では標数 2 のケースは標数 3 のケースに比べ効率的ではないが，標数 2 の場合は計算機により適していると考えられるので，標数 2 の時のペアリング計算の性能評価をする意義は十分ある．そこで本研究では標数 2 の場合のペアリングを対象とする．

2 数学基礎

2.1 有限体 $\mathbb{F}_{2^m}$

標数 2 の有限体を，以後は簡単に $\mathbb{F}_{2^m}$ と表記する．本研究ではこの有限体上でのペアリングを対象とし，Java 言語での $\mathbb{F}_{2^m}$ の演算処理から実装する．まず，そのための準備をする．

$\mathbb{F}_{2^m}$ の元 α は，予め選んでおいた $\mathbb{F}_2$ 上の既約 m 次多項式 $\gamma(x)$ を法とした次数 $m-1$ の多項式を用いて $\alpha = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \ldots + a_1x + a_0$ と記述される．ここでは $a_i \in \mathbb{F}_2$ である．このような多項式表現された元 $a_{m-1}x^{m-1} + \ldots + a_1x + a_0$ にも有限体に適応される加算乗算が定義される．例えば $(a_{m-1}x^{m-1} + \ldots + a_1x + a_0) \times x = a_{m-1}x^m + \ldots + a_1x^2 + a_0x \pmod{\gamma(x)}$ とする．同じように x の次数を上げていけば $\mathbb{F}_{2^m}$ の元の間での乗算が定義できる．このような有限体演算を用いてペアリング実装を行う．

2.2 楕円曲線

体 $\mathbb{K}$ 上の楕円曲線とは以下の式で定義される曲線を指す．

$$\begin{aligned} F(x,y) &= y^2 + (a_1x + a_3)y \\ &\quad - (x^3 + a_2x^2 + a_4x + a_6) = 0. \end{aligned} \tag{1}$$

ただし F は等式 $\dfrac{\partial F}{\partial x} = \dfrac{\partial F}{\partial y} = F(x,y) = 0$ が解を持たないものとする．ここで $a_i \in \mathbb{K}$ であって， 以下 $\mathbb{K} = \mathbb{F}_q$ とする．このような曲線上の点に対しては常に一つだけ接線が存在する．

今回用いる曲線は $\mathbb{F}_2$ 上の楕円曲線

$$F(x,y) = y^2 + y - (x^3 + x + b) = 0 \tag{2}$$

($b \in \{0,1\}$) を扱う．この曲線は Supersingular と呼ばれる性質をもっている (例えば [9]，[10] などを参照).

式 (1) で定義された楕円曲線の点 P, Q に対し，点の加法を以下のように定義する：まず，P, Q を通る直線 l ($P = Q$ の場合は P における接線) を引き，l と楕円曲線 E のもう一つの交点 $R_0 = (x_0, y_0)$ を求め，R_0 に対する “対称

点” $R = (x_0, -y_0 - a_1x - a_3)$ を $P + Q$ と定める．この演算によって E の点集合は無限遠点 ∞ を単位元とするアーベル群をなす (詳細は例えば [9]，[10] などを参照).

有限体 $\mathbb{F}_{p^m}$ 上の楕円曲線 E の点でその座標が $\mathbb{F}_{p^m}$ の元である有理点の集合に無限遠点を加えた集合を $E(\mathbb{F}_{p^m})$ とあらわす:

$$E(\mathbb{F}_{p^m}) := \{E \text{ の点 } (x, y); x, y \in \mathbb{F}_{p^m}\} \cup \{\infty\}.$$

集合 $E(\mathbb{F}_{p^m})$ は上の加法について，E の点 (無限遠点ふくむ) 全体の群の，有限な部分群をなす.

2.3 楕円曲線上のペアリング

$\mathbb{G}_1$，$\mathbb{G}_2$ を加法群，$\mathbb{G}_T$ を乗法群とし，いずれも位数 r の有限群とする．ペアリングとは，$\mathbb{G}_1 \times \mathbb{G}_2$ から $\mathbb{G}_T$ への双線型写像である． すなわち，

$$\begin{aligned} &e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T,\ (P, Q) \mapsto e(P, Q), \\ &e(P_1 + P_2, Q) = e(P_1, Q)e(P_2, Q), \\ &e(P, Q_1 + Q_2) = e(P, Q_1)e(P, Q_2). \end{aligned}$$

を満たす e である．有限体 $\mathbb{F}_q$ 上の楕円曲線を用いたペアリング (以後これを単に楕円ペアリングと呼ぶ) では，$\mathbb{G}_1$ として，$E(\mathbb{F}_q)$ の位数 r の部分群，$\mathbb{G}_2$ として，$E(\mathbb{F}_{q^k})$ の位数 r の部分群を用いる． $\mathbb{G}_T$ は 1 の r 乗根の成す群 μ_r で，これは $\mathbb{F}_{q^k}$ の乗法群に含まれている．ここで整数 k は，$E(\mathbb{F}_{q^k})$ が E の位数 r の点を全て含むような最小の整数で，埋め込み次数と呼ばれる．埋め込み次数は $r|(q^k - 1)$ を満たす最小の k であることが知られている．例えば，式 (2) で与えられる楕円曲線の場合は $k = 4$ であることが知られている.

ペアリング暗号の安全性は $\mathbb{F}_{q^k}$ の乗法群上での離散対数問題の困難さに基づいているので，ペアリングの値のサイズは現状では 1024 ∼ 2048 ビットを想定している． 例えば 1024 ビット以上のペアリングの計算を考えると，式 (2) の楕円曲線を用いるのであれば $k = 4$ であるから，$\mathbb{F}_q = \mathbb{F}_{2^m}$ は $m = 256$ 以上であることが必要である．$\mathbb{F}_{2^{mk}}$ を構成する時，$\mathbb{F}_{2^{mk}}$ の元は $\mathbb{F}_{2^m}$ の元を k 個組で持つ．高速な拡大体 $\mathbb{F}_{2^{mk}}$ の構成には $\mathbb{F}_{2^m}$ の k の因数次拡大を繰り返し $\mathbb{F}_{2^{mk}}$ を構成するタワーフィールドと呼ばれる方法が用いられる.

3 ペアリングの計算法

暗号でよく用いられるペアリングとして Tate ペアリングが上げられるが, その改良版である η_T ペアリングの実装を今回の対象とする.

3.1 η_T ペアリング

η_T ペアリングは Barreto ら [1] によって提案されたペアリングである．オリジナルの Tate ペアリングのミラーアルゴリズム [7] による計算ではループの回数が $\log_2 r$ であるのに対し，η_T ペアリングはそれより少ないループ回数で計算できるという利点を持つ．例えば式 (2) の曲線を用いた場合は Tate ペアリングのループ回数はほぼ m なのに対し，η_T ペアリングの場合はそれが約半分になる.

式 (2) によって定義されるような楕円曲線に関しては η_T ペアリングを下記のように記述できる:

$$\begin{aligned} \eta_T(P, Q) &= f_{T,P}(\phi(Q))^{(2^{2m}-1)(2^m \mp 2^{\frac{m+1}{2}}+1)}, \\ f_{T,P}(\phi(Q)) &= l(\phi(Q)) \prod_{j=0}^{\frac{m-1}{2}-j} g_{[2^j]P}(\phi(Q))^{2j} \end{aligned}$$

ここで，g_V は $V = (x_V, y_V)$ に対して定義される 2 変数関数 $g_V(x, y) = (x_V^2 + 1)(x_V + x) + y_V + y$ で，$[2^j]P$ は P の 2^j 倍点である．そして入力点は $P = (x_P, y_P)$, $Q = (x_Q, y_Q) \in E(\mathbb{F}_{2^m})$ である．ϕ は $E(\mathbb{F}_{2^m})$ の点から $E(\mathbb{F}_{2^{4m}})$ の点に移す写像 (distortion map と呼ばれている) である．つまり $\phi(Q)$ は $E(\mathbb{F}_{2^{4m}})$ の点であることに注意しておく.

η_T ペアリングの計算アルゴリズムを以下に記す ([**Algorithm** 1]). アルゴリズム中の λ，v_1，v_2 は m によって決まる定数である (詳細は [1] を参照). このアルゴリズムでは distortion map は内部で展開され最適化を行っているのでその作業を実装段階では意識する必要はない.

Algorithm 1 楕円曲線上の η_T ペアリング.

Input: $P, Q \in E(\mathbb{F}_{2^m})$

Output: $e(P,Q) \in \mathbb{F}_{2^{mk}}$

$P = (x_P, y_P)$, $Q = (x_Q, y_Q)$

$u \leftarrow \lambda$

$f \leftarrow y_Q + u(x_Q + x_P + 1) + y_P + \epsilon + s(x_Q + u) + t$

for $i = 0$ to $(m-1)/2$ **do**

$u \leftarrow x_P + v_1$

$x_P \leftarrow \sqrt{x_P}$, $y_P \leftarrow \sqrt{y_P}$

$g \leftarrow u(x_P + x_Q + v_1) + y_P + y_Q + (1 - v_1)x_P + v_2 + s(u + x_Q) + t$

$f \leftarrow f \times g$

$x_Q \leftarrow x_Q^2$, $y_Q \leftarrow y_Q^2$

end for

return $f^{(2^{2m}-1)(2^m \mp 2^{(m+1)/2}+1)}$

3.2 最終べき計算について

上で示したアルゴリズムは大きく二つのパートに分類することができる．ひとつは $f_P(Q)$ を計算する「メインループ」と呼ばれる部分で，もう一つが最後のべき乗計算でこれを「最終べき」と呼ぶことにする．

ペアリングの最終べきは多量の乗算とべき乗演算を行う必要があり，工夫無しではメインループより演算量が多いこともある．そのため，用いている有限体や楕円曲線の特性を可能な限り利用して高速化を行う必要がある．

本研究の楕円ペアリングでは最終べきは [2] にある最終べきの最適化手法を用いる．これにより，メインループ以上の計算コストを要していた[1]最終べきの処理を，メインループに比べ無視できるレベルにまで引き下げられる．

4 ソフトウェア実装

4.1 有限体表現

$\mathbb{F}_{2^m}$ の元を表す多項式の係数を int のビットで表す．一般的に m は素数を用い，m 次の既約多項式 $\gamma(x)$ を一つ選び，ビットあふれしたものを短縮させる．通常は，計算時間を考慮し，既約多項式として $\gamma(x) = x^m + x^d + 1$ (ただし $1 \leq d \leq m-1$) の型のものを用いる．

$\mathbb{F}_{2^m}$ の元の表現には m ビット必要になり，安全なペアリング暗号の設計のためには $m > 160$ が必要となる．このため，元の表現には多倍長のビットが扱える int 配列が必要となる．高速なペアリングライブラリを実装するためにまず，有限体の元の格納クラスを実装する必要がある．本実装では int の配列 $a[j], a[j-1], \ldots, a[0]$ に対し，下の $a[0]$ からビットを並べることで m ビットのビット列を格納させる方法をとっている．具体的には例えば $a[0]$ が整数の 1 に対応するとき有限体の元は 1 であり，$a[0]$ が 2 のときは多項式表現された有限体の元 x を表す．

int を用いて m ビットの多項式のすべての元の係数を表すとき，int は $mInt = m/32 + 1$ 個必要である．その int の配列を coef[$mInt$], coef[$mInt-1$], ..., coef[1], coef[0] と表し，これらの最上位 int を coef[$mInt$] とし，mBit $= m(\mathrm{mod}32)$ ビット以上のビットに 1 が存在するとき，合同計算より多項式の次数を下げるようにしている．

4.2 有限体演算

$\mathbb{F}_2$ における加算は元どうしの排他的論理和の演算に相当する．$\mathbb{F}_{2^m}$ の元 $\alpha(x) = a_{m-1}x^{m-1} + \ldots + a_1 x + a_0$ と $\beta(x) = b_{m-1}x^{m-1} + \ldots + b_1 x + b_0$ では $\gamma(x)$ を法とした多項式の加算を行う．$\alpha(x) + \beta(x) = (a_{m-1} + b_{m-1})x^{m-1} + \ldots + (a_1 + b_1)x + a_0 + b_0$ と加算は $\mathbb{F}_2$ の条件より $a_i + b_i$ は a_i と b_i が異なるとき 1 となり，それ以外は 0 となるためである．よって，加算はすべての int 間の排他的論理和演算を用いる．

$\mathbb{F}_{2^m}$ の乗算では，一般的な Shift-add アルゴリズムはコストが大きいため採用せず，[5] の Right-to-Left comba method を実装し，平方計算は [5] の方法を実装している．これらの $\mathbb{F}_{2^m}$ 演算アルゴリズムを実装することにより，有限体計算が高速化され，ペアリング計算全体の高速化につながる．特に乗算，平方計算はペアリングのアルゴリズムの中で多く用いられるので，その計算コストを減少させることは重要な課題

[1] BigInteger でタワーフィールド上の除算を用いた著者らによる実装では η_T ペアリングの最終べきにメインループの 1．3 倍の実行時間がかかっている

となる．しかし，これらの演算では演算結果が入力の 2 倍の長さを持ってしまうため，CPU ワード長による合同演算実装している．

逆元演算では [5] の拡張ユークリッド互除法を実装している．その重さは 5.1.1 にあるとおり乗算 10 回以上に匹敵し，ペアリングではこの逆元演算を減らすことは速度向上に直結する．

$\mathbb{F}_{2^m}$ 上の平方根演算としては [8] の平方根演算アルゴリズムを実装している．

5 実装結果と評価・考察

本研究では η_T ペアリングを Java で実装したが，外部ライブラリには一切頼らず有限体演算から実装している．

実装環境は [表 1] のようになっている．この

言語	Java1.6.07
CPU	Core2DuoT7250
メモリ	2GB
OS	WindowsVista

表 1: 実装環境

状況下で Java の serverVM で速度評価を行った結果は以下の通りである．

5.1 実装結果と性能評価

5.1.1 有限体演算について

ペアリング演算の基礎となる $\mathbb{F}_{2^m}$ での演算速度は [表 2] のようになっている．この表からわかるように，演算の重さは，逆元演算，乗算，平方根演算，平方計算，合同計算，加算の順に大きくなっている．加算は上述したとおりビットの格納に用いた int 間の排他的論理和を int の個数回だけ行ったものに等しいため，それと対比して他の処理の演算量の多さが見て取れる．ペアリング計算は，この有限体演算に基づいているので，当然，演算量の多い演算の回数をできるだけ減らすことが重要になってくる．この表から互いの処理時間関係がわかるので，ペアリングのアルゴリズムを構成する時の参考になる．

	$m = 79$	$m = 241$
加算	54	85
乗算	3178	10587
平方計算	205	307
平方根計算	7102	24171
逆元演算	33653	144080
合同計算	57	100

表 2: $\mathbb{F}_{2^m}$ 上での有限体演算コスト [nsec].

5.1.2 ペアリング計算について

ペアリングの計算時間は [表 3] のようになっている．数値は 1000 回計算したものの平均値である．この実行速度はペアリングのライブラリとしては機械語や CPU 特化の多倍長演算を用いたライブラリよりは低速であるが，Java 実装としては高速に実行できるものを得られたと考える．なお，オリジナルの Tate ペアリングを実装したところ，同じ環境の下で 46[msec] 程度であったので, η_T ペアリングの方が高速であることが確認できた．

メインループ	9.28
最終べき	0.56
ペアリング全体	9.96

表 3: ペアリング計算時間 [msec].

5.2 BigInteger クラスを用いた場合

Java の標準クラスには BigInteger クラスという多倍長演算を行うクラスが存在する．このクラスは m ビットのビット列を格納する多倍長演算クラスとして有限体 $\mathbb{F}_{2^m}$ の元の表現に使用できる．そのクラスを用いて本研究と同じ評価環境でペアリングの実装評価を行ったところ，メインループ部では η_T ペアリングが 87$[msec]$ となっており，本研究の結果と比較すると約 10 倍ほど時間を要していることがわかる．その原因は BigInteger の排他的論理和などの処理の遅

さ[2]と int による CPU ワード長の処理がなかったためである．このため実装を行える有限体演算アルゴリズムに制限がかかり，有限体の演算速度も上がらなかった．このためライブラリ開発ではペアリングの高速計算を目指し，BigInteger クラスを使用しなかった．

6 むすび

本研究の結果として，一般的に普及している程度の性能の CPU において約 10 ミリ秒でペアリングの計算をできる Java のライブラリを得た．これは Java 依存であるため CPU の多倍長演算命令を使用していない分，アセンブリを使用した C 言語等による実装には速度的に劣るが，使用する分にはストレスを感じない程度だと考えられる．先行研究 [6] では標数 3 の有限体上でのペアリング計算を行っているが，実行時間としてはほぼ同等のものが標数 2 の有限体上でも実装できた．

7 謝辞

Jean-Luc Beuchat 氏 (筑波大) に深謝します．

参考文献

[1] P.S.L.M. Barreto, S. Galbraithm, C. OhEigeartaigh and M. Scott, “Efficient Pairing Computation on Supersingular Abelian Varieties,” Cryptology ePrint Archive, Report 2004/375.

[2] J.-L. Beuchat, N. Brisebarre, J. Detrey, E. Okamoto and F. Rodriguez-Henriquez, ”A Comparison Between Hardware Accelerators for the Modified Tate Pairing over $\mathbb{F}_{2^m}$ and $\mathbb{F}_{3^m}$,” Cryptology ePrint Archive: Report 2008/115.

[3] D. Boneh and M. Franklin, “Identity-based encryption from the Weil pairing,” SIAM Journal of Computing, 32(3), p.p.586–615, 2003.

[4] D. Boneh, B. Lynn and H. Shacham, “Short signatures from the Weil pairing,” LNCS 2248, p.p. 514–532. Springer, 2002.

[5] D. Hankerson, J.L. Hernandez and A. Menezes, “Software Implementation of Elliptic Curve Cryptography over Binary Fields,” Cryptographic Hardware and Embedded Systems, Springer-Verlag, p.p.1–24, 2000.

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

[7] V. S. Miller, “Short Programs for functions on Curves,” avaivable at http://crypto.stanford.edu/miller/

[8] F. Rodriguez-henriquez, N. A. Saqib, A. Diaz-Perez and C.K. Koc, “Cryptographic Algorithms on Reconfigurable Hardware,” Springer-Verlag, 2006.

[9] J. H. Silverman, The arithmetic of elliptic curves, Springer, 1986.

[10] L.C. Washington, Elliptic curves, Number theory and cryptography, Chapman, 2003.

[2] 一番速い処理で本研究の多倍長実装の 4 倍の時間を要する