SCIS 2007 The 2007 Symposium on Cryptography and Information Security
Sasebo, Japan, Jan. 23-26, 2007
The Institute of Electronics, Information and Communication Engineers

Compressed Jacobian Coordinates

小林 鉄太郎* 星野 文学 * 青木 和麻呂 *
Tetsutaro Kobayashi Fumitaka Hoshino Kazumaro Aoki

あらまし VietCrypt2006 で発表された OEF 上の楕円曲線の座標、Compressed Jacobian coordinates の一般化を行う。Compressed Jacobian coordinates は、拡大体上で定義された楕円曲線上の点の加算・二倍算を高速に行える座標であり、特に OEF(拡大次数 5) においては従来の Jacobian coordinates よりも 28% 程度高速となり、ベースポイントの位数が 244bit の楕円曲線のスカラー倍が 41.9 μsec per operation on a 2.82-GHz Athlon 64 FX PC という速度が示されている。本稿では、OEF 以外の楕円曲線暗号への Compressed Jacobian coordinates の適用を考察する。

Keywords: elliptic curve cryptosystem, coordinate system, OEF, fast software implementation

1 はじめに

EC-DSA や EC-ElGamal など楕円曲線暗号は、RSA 暗号に比べて鍵サイズが少なく、演算速度が速いというメリットがある。特に、OEF [1] 上の楕円曲線はソフトウェア実装を行った際に高速になる。

楕円曲線上の点には複数の表現方法があり、よく使われるものとしてはアフィン座標とヤコビ座標がある。楕円曲線上の演算 (ECADD、ECDBL) の計算コストはこれら座標系の選択によって決定されるため、効率の良い座標の研究が行われてきた。逆元と乗算の速度比によって効率のよい座標系が決定される。

Compressed Jacobian coordinates [3] では、「擬似逆元」という逆元と似た性質をもつ関数を用いる。OEF では擬似逆元は逆元とくらべて高速に計算できるため、これを利用して楕円加算・楕円 2 倍演算を高速化することができた。

本稿では、OEF 以外の楕円曲線暗号への Compressed Jacobian coordinates の適用を考察する。

拡大体上の楕円曲線暗号が用いられるケースとして、ペアリングを用いた方式への適用があげられる。楕円曲線のペアリングを用いることで、通常の離散対数ベースの暗号では実現が困難だった IBE などの新しいアプリケーションが提案されている。ペアリングの関数は通常、$\mathbb{F}_p$ 上の楕円曲線上の点と $\mathbb{F}_{p^k}$ 上の楕円曲線の点から有限体 $\mathbb{F}_{p^k}$ へのマップを行うので、IBE などペアリングを利用した方式では、$\mathbb{F}_{p^k}$ 上の楕円演算を行う必要がある。

楕円曲線暗号において OEF を用いる場合は拡大次数 k は素数を用いることが多いが、MNT 曲線などで生成された ペアリング用の曲線では $k = 2, 4, 6$ などの合成数を用いるのが一般的である。この場合、 Compressed Jacobian coordinates は複数の表現方法が考えられる。本稿では、合成数拡大次数の場合の最適解について議論を行う。

Jacobian coordinates が楕円曲線上の点 $P = (X, Y, Z)$ を $X, Y, Z \in \mathbb{F}_{p^k}$ を用いてあらわしているが、Compressed Jacobian は $P = (X, Y, z), X, Y \in \mathbb{F}_{p^k}, z \in \mathbb{F}_p$ であらわす。これにより、楕円加算・楕円二倍の演算コストを削減することができる。Jacobian coordinates ($a = -3$) では

$$\texttt{ECADD} = 8\texttt{M} + 5\texttt{S},\ \texttt{ECDBL} = 4\texttt{M} + 4\texttt{S}$$

となるのに対し、Compressed Jacobian coordinates では

$$\texttt{ECADD} = 2\texttt{M} + 1\texttt{S} + 1\texttt{P} + 7\texttt{v},\ \texttt{ECDBL} = 2\texttt{M} + 2\texttt{S} + 1\texttt{P} + 3\texttt{v}$$

となる。ただし、$\texttt{M}, \texttt{S}, \texttt{P}, \texttt{v}$ を、有限体 $\mathbb{F}_{q^k}$ 上の乗算、自乗、擬似逆元、ベクトル乗算のコストとする。

一方、合成数拡大 $k = 6$ の場合

- $\mathbb{F}_p$-Compressed Jacobian, $z \in \mathbb{F}_p$
- $\mathbb{F}_{p^2}$-Compressed Jacobian, $z \in \mathbb{F}_{p^2}$
- $\mathbb{F}_{p^3}$-Compressed Jacobian, $z \in \mathbb{F}_{p^3}$

の 3 通りが考えられる。この 3 つの Compressed Jacobian coordinate の選択によって P と v の演算コストが変化するため楕円加算・楕円二倍の演算コストもそれによって決定される。

2 章で記号の定義を行い、3 章でこのコストについての議論を行い、最適解を示す。

* NTT Information Sharing Platform Laboratories, NTT Corporation, 1-1 Hikarinooka, Yokosuka-shi, Kanagawa-ken, 239-0847 Japan.

2 準備

2.1 OEF

OEF [1] は有限体 $\mathbb{F}_{q^m}$ のうち、以下の性質を満たすものである。

- q はプロセッサの最大ワードサイズに近いが それを超えない値
- $q = 2^n \pm c$ ただし $\log_2 c \leq n/2$
- 既約二項式 $f(x) = x^m - \omega$ が存在すること

以下のような多項式表現で OEF 上の元をあらわす。$A \in \mathbb{F}_{q^m}$,

$$A = a_{m-1}\alpha^{m-1} + \cdots + a_1\alpha + a_0$$

ただし $a_i \in \mathbb{F}_q$ であり、 $\alpha \in \mathbb{F}_{q^m}$ は $f(x)$ の原始元とする。q はプロセッサのワードサイズ以下の数なので、A を m 個のレジスタを用いてあらわすことが出来る。

2.2 計算コスト

本稿では、ある素数または素数のべきを q として、q の m 次拡大体 $\mathbb{F}_{q^m}$ 上の楕円加算 ECADD および楕円二倍 ECDBL のコストを以下の記号を用いて論ずる。

M は $\mathbb{F}_{q^m}$ 乗算の計算量

S は $\mathbb{F}_{q^m}$ 自乗の計算量

I は $\mathbb{F}_{q^m}$ 逆元の計算量

P は $\mathbb{F}_{q^m}$ 擬似逆元の計算量

A は $\mathbb{F}_{q^m}$ の加算の計算量

v は乗算 $\mathbb{F}_q \times \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$ の計算量

m は $\mathbb{F}_q$ 乗算の計算量

s は $\mathbb{F}_q$ 自乗の計算量

i は $\mathbb{F}_q$ 逆元の計算量

a は $\mathbb{F}_q$ 加算の計算量

f は $\mathbb{F}_{q^m}/\mathbb{F}_q$ フロベニウス写像の計算量

2.3 有限体の表現

この章では、有限体の表現と演算コストについて論ずる。

$\mathbb{F}_{q^m}$ 上の四則演算によるアルゴリズムを考えると多くの場合、逆元演算は乗算に比べてコストが高い。したがって、しばしば有理数表現 $\frac{N}{D}$ $(N, D \in \mathbb{F}_{q^m}^{\times})$ を用いることで、逆元を計算しない方法が用いられる。

$$\left(\frac{N}{D}\right)^{-1} = \frac{D}{N}$$

一方、有理数表現を用いると乗算と加算の回数が増す。

$$\begin{aligned} \frac{N_1}{D_1}\frac{N_2}{D_2} &= \frac{N_1N_2}{D_1D_2} \\ \frac{N_1}{D_1} + \frac{N_2}{D_2} &= \frac{N_1D_2 + N_2D_1}{D_1D_2} \end{aligned}$$

すなわち、有理数表現は逆元演算を乗算と加算数回で代用する方法と言える。逆元が非常に遅い場合には有効である。

有限体が $\mathbb{F}_{q^m}$ の場合、擬似逆元が存在する。真の逆元にくらべて高速に演算できる場合がある。

擬似逆元の定義を以下に示す。

Definition 2.1 (Pseudo-Inversion Algorithm)

Input: $X \in \mathbb{F}_{q^m}$

Output: $(\iota(X), \mathbf{N}(X)) \in \mathbb{F}_{q^m} \times \mathbb{F}_q^{\times}$ $s.t.$ $\iota(X)X = \mathbf{N}(X)$

$\iota(X)$ を X の擬似逆元、 $\mathbf{N}(X)$ を X の co-擬似逆元と呼ぶ。擬似逆元を高速に演算できる場合には、$\frac{N}{d}$ $(N \in \mathbb{F}_{q^m}, d \in \mathbb{F}_q^{\times})$ という有理数表現が有効である。従来の有理数表現より高速に計算可能である。

$$\begin{aligned} \left(\frac{N}{d}\right)^{-1} &= \frac{\iota(N)d}{\mathbf{N}(N)} \\ \frac{N_1}{d_1}\frac{N_2}{d_2} &= \frac{N_1N_2}{d_1d_2} \\ \frac{N_1}{d_1} + \frac{N_2}{d_2} &= \frac{N_1d_2 + N_2d_1}{d_1d_2} \end{aligned}$$

$m > 1$ の場合には $\mathbb{F}_{q^m}$ と $\mathbb{F}_q$ の乗算は $\mathbb{F}_{q^m}$ どうしの乗算よりも高速に計算可能である。擬似逆元アルゴリズムを用いることで、連続する演算を高速化できる。この論文は、このアイデアを楕円曲線上のスカラー倍に適用する。

2つの擬似逆元演算アルゴリズムの例を 2.4 章と 2.5 章に示す。

2.4 Norm による擬似逆元計算

擬似逆元の演算を行なうために、q^i によるべき乗を行なう方法。

$X \in \mathbb{F}_{q^m}$ に対し、

$$z = \prod_{i=0}^{m-1} X^{q^i}$$

をノルムと呼ぶ。ノルム z は $z \in \mathbb{F}_q$ という性質がある。定義より

$$z = X\left(\prod_{i=1}^{m-1} X^{q^i}\right)$$

また

$$Y = \prod_{i=1}^{m-1} X^{q^i}$$

とおく。出力値 Y が擬似逆元となり、z が co-擬似逆元である。上記演算を伊東辻井 [2] アルゴリズムを用いて高速に行なうことができる。

たとえば $m = 5$ の場合は以下のように計算できる。

$$\begin{aligned} Y &\leftarrow X \cdot X^q \\ Y &\leftarrow Y \cdot Y^{q^2} \\ Y &\leftarrow Y^q \\ z &\leftarrow X \cdot Y \end{aligned}$$

計算量は 2 項式を用いた多項式表現を前提とすると以下のとおり。

$$\mathtt{P} = 2\mathtt{M} + 3\mathtt{f} + 1\mathtt{v} \approx 2\mathtt{M} + 3.4\mathtt{v}$$

2.5 ユークリッドの互除法をもちいた擬似逆元

`deg(X)` を `X` の次数とし、`lead(X)` を `X` の先等の係数とする。`swap(X,Y)` は `X` と `Y` の値を入れ替える。以下のアルゴリズムで擬似逆元を演算することが出来る。

```
A = f(x) ; // irreducible polynomial
B = g(x) ; // input polynomial
(where deg(A) > deg(B)) ≠ 0

C =  0  ;
D =  1  ;

while(deg(B) > 0){
  b = lead(B) ;
  while(deg(A) >= deg(B)){
    a = lead(A) ;
    n = deg(A) - deg(B) ;

    A *= b ; C *= b ;
    A -= a*B*x^n ;
    C -= a*D*x^n ;
  } ;
  swap(A,B) ;
  swap(C,D) ;
} ;

// output D as pseudo-inversion
// output B as co-pseudo-inversion
```

このアルゴリズムのおおよその演算コストは $4m\mathtt{v}$ となる。

3 Compressed Jacobian Coordinates

Jacobian coordinate では楕円曲線上の点は $\mathbb{F}_{q^m}^3$ の要素として表現されていた。この表現の冗長度を擬似逆元を用いて減らすことができる。Compressed Jacobian coordinate では楕円曲線上の点は $\mathbb{F}_{q^m}^2 \times \mathbb{F}_q$ の要素となる。加算公式はいかに示すとおり。

Compressed Jacobian coordinates

$$(X, Y, z) \in \mathbb{F}_{q^m}^2 \times \mathbb{F}_q, \quad \text{where } Y^2 = X^3 + az^4X + bz^6$$

Compressed Jacobian coordinates を用いた楕円加算公式

Input: $(X_1, Y_1, z_1), (X_2, Y_2, z_2)$, Output: (X_3, Y_3, z_3).

$$\begin{aligned} z_3' &\leftarrow z_1 z_2 \\ (X_1', Y_1') &\leftarrow (X_1 z_2^2, Y_1 z_2^3) \\ (X_2', Y_2') &\leftarrow (X_2 z_1^2, Y_2 z_1^3) \\ \Lambda_n &\leftarrow \iota(X_2' - X_1') \cdot (Y_2' - Y_1') \\ \lambda_d &\leftarrow \mathbf{N}(X_2' - X_1') \\ z_3 &\leftarrow \lambda_d z_3' \\ (X_1'', Y_1'') &\leftarrow (X_1' \lambda_d^2, Y_1' \lambda_d^3) \\ (X_2'', Y_2'') &\leftarrow (X_2' \lambda_d^2, Y_2' \lambda_d^3) \\ X_3 &\leftarrow \Lambda_n^2 - X_1'' - X_2'' \\ Y_3 &\leftarrow \Lambda_n(X_1'' - X_3) - Y_1'' \end{aligned}$$

実際には Y_2'' は X_3, Y_3, z_3 を求めるのに必要ないので、Y_2'' を求めるこのコストは無いものとして考える。

Compressed Jacobian coordinates を用いた楕円二倍公式

Input: (X_1, Y_1, z_1), Output: (X_3, Y_3, z_3).

$$\begin{aligned} \Lambda_n &\leftarrow \iota(2Y_1) \cdot (3X_1^2 + az_1^4) \\ \lambda_d &\leftarrow \mathbf{N}(2Y_1) \\ z_3 &\leftarrow \lambda_d z_1 \\ (X_1'', Y_1'') &\leftarrow (X_1 \lambda_d^2, Y_1 \lambda_d^3) \\ X_3 &\leftarrow \Lambda_n^2 - 2X_1'' \\ Y_3 &\leftarrow \Lambda_n(X_1'' - X_3) - Y_1'' \end{aligned}$$

3.1 Compressed Jacobian coordinate のバリエーション

$\mathbb{F}_{p^k}$ 上の楕円曲線演算を行なう場合、k が合成数の場合は複数の Compressed Jacobian coordinate 表現が可能である。

この章では、この場合の最適解について分析する。

$k = 6$ の場合 以下の 3 通りがある。

- $\mathbb{F}_p$-Compressed Jacobian coordinates $q = p, m = 6$ として Compressed Jacobian coordinates 表現を行なう。

$$\mathtt{P} = 3\mathtt{M} + 3\mathtt{f} + 1\mathtt{v} = 3\mathtt{M} + 3.5\mathtt{v}$$

$$\mathtt{ECADD} = 5\mathtt{M} + 1\mathtt{S} + 10.5\mathtt{v}, \ \mathtt{ECDBL} = 5\mathtt{M} + 2\mathtt{S} + 6.5\mathtt{v}$$

表 1: 教科書法を仮定した場合の楕円演算コスト

	ECADD	ECDBL
$\mathbb{F}_p$-Compressed Jacobian	7.35M	7.28M
$\mathbb{F}_{p^2}$-Compressed Jacobian	6.71M	5.98M
$\mathbb{F}_{p^3}$-Compressed Jacobian	6.85M	5.45M
Jacobian	11.00M	6.40M

- $\mathbb{F}_{p^2}$-Compressed Jacobian coordinates $q = p^2, m = 3$ として Compressed Jacobian coordinates 表現を行なう。

$$\mathtt{P} = \mathtt{M} + 2\mathtt{f} + \mathtt{v} = \mathtt{M} + 2.33\mathtt{v}$$

$$\mathtt{ECADD} = 3\mathtt{M} + 1\mathtt{S} + 9.33\mathtt{v},\ \mathtt{ECDBL} = 3\mathtt{M} + 2\mathtt{S} + 5.33\mathtt{v}$$

- $\mathbb{F}_{p^3}$-Compressed Jacobian coordinates $q = p^3, m = 2$ として Compressed Jacobian coordinates 表現を行なう。

$$\mathtt{P} = \mathtt{f} + \mathtt{v} = 1.5\mathtt{v}$$

$$\mathtt{ECADD} = 2\mathtt{M} + 1\mathtt{S} + 8.5\mathtt{v},\ \mathtt{ECDBL} = 2\mathtt{M} + 2\mathtt{S} + 4.5\mathtt{v}$$

実際には上記 v のコスト ($\mathbb{F}_q$ 上の乗算 m 回) が異なるため、$\mathbb{F}_{q^m}$ 上の乗算をどのように実現しているかによって M と v のコスト比が変わってくる。$\mathtt{S} = 0.6\mathtt{M}$ として、乗算は教科書法で行なっていると仮定すると表 1 のようになり、$\mathbb{F}_{p^3}$-Compressed Jacobian coordinates が最適となる。

一般に k 合成数に対してノルム (伊東辻井法) を用いた擬似逆元演算を用いる場合、m を小さい数とするのが最適となる。擬似逆元の演算コスト $\mathtt{P} = O(\log m)\mathtt{M}$ ベクトル乗算コスト $\mathtt{v} = \frac{1}{m}\mathtt{M}$ となるため、

$$\mathtt{ECADD} = 2\mathtt{M} + 1\mathtt{S} + 1\mathtt{P} + 7\mathtt{v},\ \mathtt{ECDBL} = 2\mathtt{M} + 2\mathtt{S} + 1\mathtt{P} + 3\mathtt{v}$$

を最小化するために、$\log m + \frac{7}{m}$, $\log m + \frac{3}{m}$ が最小となる m を選択するべきである。

4 まとめ

VietCrypt2006 で発表された OEF 上の楕円曲線の座標、Compressed Jacobian coordinates の一般化を行なった。本稿では、OEF 以外の楕円曲線暗号への Compressed Jacobian coordinates の適用を考察し、ペアリングで用いられる拡大体上の楕円曲線演算への適用を行なった。合成数次拡大の場合の Compressed Jacobian coordinates の分析を行い、最適解を示した。

参考文献

[1] Bailey, D.V., Paar, C.: Optimal extension fields for fast arithmetic in public-key algorithms. In Krawczyk, H., ed.: Advances in Cryptology — CRYPTO'98. Volume 1462 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg, New York (1998) 472–485

[2] Itoh, T., Tsujii, S.: A fast algorithm for computing multiplicative inverses in GF(2^m) using normal bases. In: Information and Computation. Volume 78. (1988) 171–177

[3] Hoshino, B., Kobayashi, T., Aoki, K.: Compressed Jacobian Coordinates for OEF In: proceedings Vietcrypt 2006, Springer-Verlag 2006, LNCS 4341, (To appear).