職業能力開発総合大学校紀要 第41号A (2012年3月)
Bull. Polytechnic University No41-A, March 2012

# パステーブルを用いる楕円スカラー倍算の計算量の評価

相良 佳孝† 櫻木 伸英‡ 松嶋 智子* 足原 修

# Complexity Evaluation of Elliptic Curve Scalar Multiplication Using Path Table

Yoshitaka Sagara† Nobuhide Sakuragi‡ Tomoko K. Matsushima* Osamu Ashihara

*Abstract* Efficient elliptic curve arithmetic is crucial for cryptosystems based on elliptic curves. Such cryptosystems often require computing a scalar multiple $kP$ of a base point $P$. Recently, some papers have proposed efficient algorithms to compute $\lambda P \pm \mu Q$ directly for small integers $\lambda$ and $\mu$ from given points $P$ and $Q$. The authors have developed some programs to find out the path with the minimum cost for each scalar multiplication $kP$ under the condition that several operations of $\lambda P \pm \mu Q$ can be used, and have also developed the minimum-cost path-table of scalar multiples $kP$ for $1 \leq k \leq k_{max}$. In this paper, we evaluate the costs for this algorithm and compare them with those for some conventional well-known algorithms.

**Key words:** elliptic curves, scalar multiplication, path-table, signed binary representation

## 1 まえがき

楕円曲線暗号システムの実用化においては，実装するデバイスの計算能力によらず暗号処理が高速に実行される必要があり，楕円曲線上の演算の効率化が重要な課題となっている．

楕円曲線暗号の主演算は，楕円曲線上の任意の有理点 $\mathcal{P}$ からそのスカラー倍点 $k\mathcal{P}$ を求めるスカラー倍算である．これまでに，スカラー倍算の基本演算を加算公式 $P+Q$ と 2 倍算公式 $2P$ に限定した効率的なスカラー倍算の計算方法が数多く発表されている [1]-[4]．一方，ある程度小さい整数 $\lambda$ と $\mu$ に対して，$\lambda P \pm \mu Q$ をアフィン座標系で直接求める計算公式と，それを用いた加算連鎖についての研究が最近発表された．Ciet ら [5] は $\lambda = 2,3,4$, $\mu = 1$ について，溝添ら [6] は $\lambda = 5$, $\mu = 1,2$ についてコストの低い計算公式を導いた．

著者らは，複数の基本演算公式 $\lambda P \pm \mu Q$ が利用できる場合に，それぞれの基本演算のコストを考慮した上で，最小のコストで $k\mathcal{P}$ の演算を行う加算連鎖パスを計算機プログラムで探索し，整数 $k_{max}$($10^5$ 程度) までの最適な加算連鎖のパスを記憶したパステーブルを作成した [7],[8],[9]．さらに，文献 [10] では，パステーブルに与えられている最大の整数 $k_{max}$ に対して，$k_{max} \ll K$ となるような整数 $K$ のスカラー倍点 $K\mathcal{P}$ を高階差分の考え方を用いて計算する方法を提案した．

ここでは，パステーブルを用いて，任意の整数 $k$ に対して $k\mathcal{P}$ を求めるスカラー倍算法をパステーブル法と呼ぶ．これまでに，テーブルを用いるアルゴリズムとしてはウィンドウ法が提案されているが，ウィンドウ法では楕円曲線上の有理点 $\mathcal{P}$ が与えられるたびに事前計算を行い，複数のスカラー倍点をテーブルに保持しなければならない．これに対して，パステーブル法におけるテーブルは，どのようなベース点 $\mathcal{P}$ に対しても共通であるため，あらかじめ求められたパステーブルを ROM などのメモリに記憶させておけばよいという利点がある [11],[12]．

本論文では，パステーブルで与えられる最大の整数 $k_{max}$ に対して，$k \leq k_{max}$ となるようなスカラー倍算 $k\mathcal{P}$ について，アフィン座標系及び重み付き射影座標系におけるバイナリ法，符号付バイナリ法及びウィンドウ法の演算コストの比較を行う．さらに，パステーブルに必要なメモリの容量を試算する．

## 2 パステーブル法

### 2.1 パステーブルを用いたスカラー倍算

最近 $5P$ や $5P \pm Q$ など，比較的小さい整数 $\lambda$ と $\mu$ に対して，$\lambda P + \mu Q$ をアフィン座標系で直接求める計算公式が発表されている．これらの演算は $P+Q$ と $2P$ のみの演算より低いコストになることが示されている．

ここでは，Ciet ら [5] や溝添ら [6] の結果を利用して $3P$, $4P$, $5P$, $2P \pm Q$, $3P \pm Q$, $4P \pm Q$, $5P \pm Q$, $5P \pm 2Q$ の基本演算を利用できるものとする．

† 国立吉備高原職業リハビリテーションセンター
‡ 東北職業能力開発大学校附属 青森職業能力開発短期大学校
* 職業能力開発総合大学校 電子情報システム工学科

表 1: $\lambda P + \mu Q$ の演算とその計算コスト

| $\lambda P + \mu Q$ | Complexity | Ref. |
|---|---|---|
| $2P$ | $1i$+$2s$+$2m$ | |
| $3P$ | $1i$+$4s$+$7m$ | [5] |
| $4P$ | $1i$+$9s$+$9m$ | [5] |
| $5P$ | $1i$+$12s$+$13m$ | [6] |
| $P \pm Q$ | $1i$+$1s$+$2m$ | |
| $2P \pm Q$ | $1i$+$2s$+$9m$ | [5] |
| $3P \pm Q$ | $2i$+$4s$+$9m$ | [5] |
| $4P \pm Q$ | $2i$+$4s$+$11m$ | [5] |
| $5P \pm Q$ | $1i$+$11s$+$16m$ | [6] |
| $5P \pm 2Q$ | $1i$+$13s$+$20m$ | [6] |

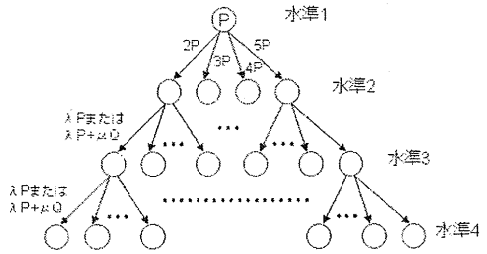



図 1: 複数の基本演算を利用する加算連鎖の木

表 1 に，Ciet ら [5]，および溝添ら [6] により発表された演算とその計算コストを示す．ここで，$m, s, i$ は，それぞれ，定義された有限体上での乗算，二乗算，逆元計算にかかる計算量を表す．

複数の基本演算を用いる場合，$k$ に対する加算連鎖は，整数列

$$1 = a_0, a_1, a_2, \cdots, a_r = k \tag{1}$$

で表される [13]．ただし，すべての $j = 1, 2, \cdots, r$ について，

$$a_j = \lambda a_{j-1} + \mu a_l \ \ (l < j - 1) \tag{2}$$

と合成される．

図 1 は表 1 の複数の基本演算を利用する加算連鎖の木を示す．ルートとなる有理点を $\mathcal{P}$ と表記する．この木において，$P$ は作業中のノードを表し，根から $P$ までの一意に定まる経路上にあるすべてのノードが $Q$ の候補となる．生成可能なすべてのノードを木に持たせる場合には，各ノードからすべての演算に対応する枝を伸ばすことになり，深さを下げるにつれて $Q$ の候補が増えることから，急激にノード数が増加する．$\mathcal{P}$ のノードを水準 1 とし，第 $t$ 水準のノード数を $n(t)$ とすると，初期値は

$$n(1) = 1 \tag{3}$$

$$n(2) = 4 \tag{4}$$

表 2: 各基本演算の計算コスト

| $\lambda P + \mu Q$ | $i/m = 13$ | $i/m = 11$ | $i/m = 9$ | $i/m = 7$ |
|---|---|---|---|---|
| $2P$ | $16.6m$ | $14.6m$ | $12.6m$ | $10.6m$ |
| $3P$ | $23.2m$ | $21.2m$ | $19.2m$ | $17.2m$ |
| $4P$ | $29.2m$ | $27.2m$ | $25.2m$ | $23.2m$ |
| $5P$ | $35.2m$ | $33.6m$ | $31.6m$ | $29.6m$ |
| $P \pm Q$ | $15.8m$ | $13.8m$ | $11.8m$ | $9.8m$ |
| $2P \pm Q$ | $23.6m$ | $21.6m$ | $19.6m$ | $17.6m$ |
| $3P \pm Q$ | $38.2m$ | $34.2m$ | $30.2m$ | $26.2m$ |
| $4P \pm Q$ | $40.2m$ | $36.2m$ | $32.2m$ | $28.2m$ |
| $5P \pm Q$ | $37.8m$ | $35.8m$ | $33.8m$ | $31.8m$ |
| $5P \pm 2Q$ | $43.4m$ | $41.4m$ | $39.4m$ | $37.4m$ |

であり，第 $t-1$ 水準 $(t \geq 2)$ の各ノードからは $4+12(t-2)$ 本の枝が伸びるため，

$$n(t) = (4 + 12(t-2))n(t-1) \tag{5}$$

で与えられる．ただし，文献 [8] では探索コストと最適性のトレードオフを考慮して，最小コストの加算連鎖にかかわらないと考えられるノードを除外し，より深い水準のノードを生成する方法について検討している．

最大水準を 1 から 9 まで変えて，生成されるそれぞれのべき木において，コストを求めることが可能な整数 $k$ の最大値 $(kk_{max})$ と，連続してコストを求めることができる $k$ の最大値 $(k_{max})$ は，

第 1 水準 $kk_{max} = 1$, $k_{max} = 1$
第 2 水準 $kk_{max} = 5$, $k_{max} = 5$
第 3 水準 $kk_{max} = 27$, $k_{max} = 27$
第 4 水準 $kk_{max} = 145$, $k_{max} = 137$
第 5 水準 $kk_{max} = 779$, $k_{max} = 687$
第 6 水準 $kk_{max} = 4,185$, $k_{max} = 3,466$
第 7 水準 $kk_{max} = 22,483$, $k_{max} = 18,056$
第 8 水準 $kk_{max} = 120,785$, $k_{max} = 95,402$
第 9 水準 $kk_{max} = 648,891$, $k_{max} = 499,012$

である．第 9 水準において $k_{max}/kk_{max} = 0.769$ となる．

## 2.2 最小コストのパステーブル

表 2 は，$s$ を $0.8m$ とし，$i$ を $13m$,$11m$, $9m$, $7m$ とした場合のそれぞれの基本演算の計算コストを表したものである．

パステーブルは、$s = 0.8m$ と仮定して，表 1 の基本演算を用いて，文献 [9] に示された複数のプログラムにより求めたパス $(1 \leq k \leq 10^5)$ の中で最小コストのパスを選ぶものとした。今回は $i/m = 13, 11, 9, 7$ のそれぞれの場合について，パステーブル $(1 \leq k \leq 10^4)$ を生成し，それらのコストを比較した．ただし，$i/m = 11$ に

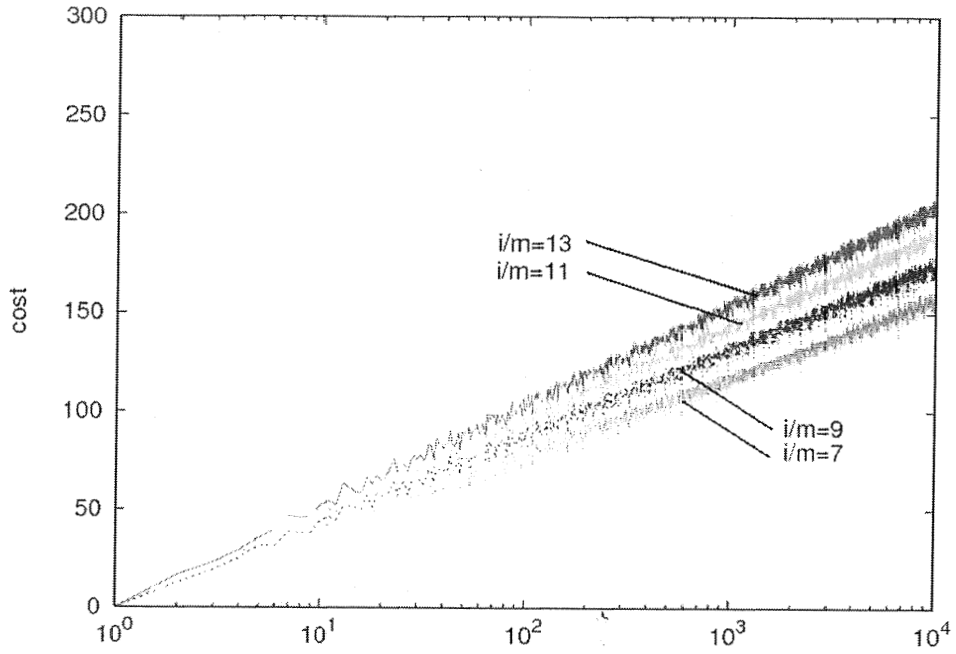



図 2: パステーブル法のコスト ($i/m = 13, 11, 9, 7$)

ついては上記のようにして求めたパステーブルの一部を用い，$i/m = 13, 9, 7$ については，枝刈りを全く行わずに水準 9 までのべき木を生成して最小コストパスを探索した．

以下では表記を簡単にするため，表 1 における乗算コスト $m$ を 1 としてスカラー倍算 $k\mathcal{P}$ のコストを表現する．

図 2 は，これらのパステーブルに示されるスカラー倍算 $k\mathcal{P}$ $(1 \leq k \leq 10^4)$ の演算コストを示したものである．

$k\mathcal{P}$ のコストは $\log k$ にほぼ比例して増大するが，その傾きは $i/m$ の値に依存する．また，図 2 から，$i/m$ の値が小さくなると，パステーブルの平均コストも下がることがわかる．これは，パステーブル法がアフィン座標系での演算によるもので，表 1 に示されるように，各基本演算に逆元計算が 1 回以上含まれていることによる．

### 2.3 パステーブルのメモリ量

パステーブルに記憶される情報は，各 $k$ に対する最小コストの加算連鎖パスである．例えば，$i/m = 11$ とすると，$k = 8,809$ のパス，すなわち $8,809\mathcal{P}$ を求めるパスは，$3P \to 3P \to 5P - Q$ $(Q = \mathcal{P}) \to 2P \to 4P \to 5P \to 5P + Q$ $(Q = 9\mathcal{P})$ の 7 段の基本演算からなり，コストは 189.4 となる．

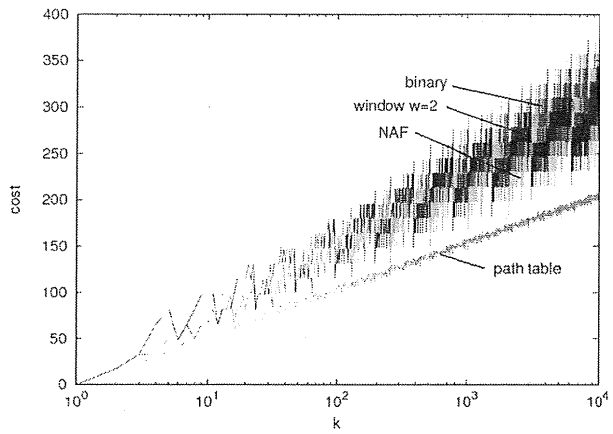
各パスの最初の演算には 4 種類の公式が用いられ，2 段目以降の演算には 16 種類の公式が用いられる．用いられた公式が $\pm Q$ を含む場合には，どの水準のノードが $Q$ として用いられているかを公式と合わせて表現する必要がある．ただし，$Q$ として取りうるノードの数は，段数が大きくなると増加する．この結果，一つのパスを表現するのに，最初の演算には 2 ビット，2 段目の演算には 4 ビット，3 段目には 5 ビット，4,5 段目には 6 ビット，6〜9 段目には 7 ビット，10〜17 段目には 8 ビットが必要となる．パステーブルに記憶された加算連鎖パスのなかで最長のものを演算回数 10(べき木の水準 11) とすると，一つのパスを表現するのに 59 ビット (約 8 Byte) が必要になる．

$k$ の値が 100,000 までのパステーブルを作るには，$k$ をアドレスとして 8 Byte を 1 word とする ROM で，約 800K Byte の容量があれば実現できる．

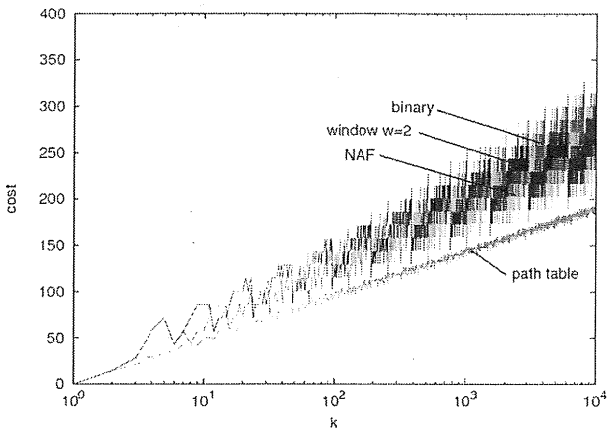
## 3 アフィン座標系でのコスト比較

ここでは，表 1 の二倍算 $2P$ と加算 $P+Q$ のコストを仮定し，アフィン座標系でのバイナリ法，符号付バイナリ法[1]，ウィンドウ法およびパステーブル法の計算コストを求める．ウィンドウ法ではテーブルのウィンドウ幅に応じてテーブル作成の計算コストを最初に加算する．このとき，ウィンドウ幅を $w$bit とすると，$2P, 3P, 5P, \ldots, (2^w - 1)P$ を計算し，これらのコストの和をテーブル作成コストとした [14]．ただし，スカラー倍算 $kP$ の係数がウィンドウ幅 $w$ より小さい場合は，最初のテー

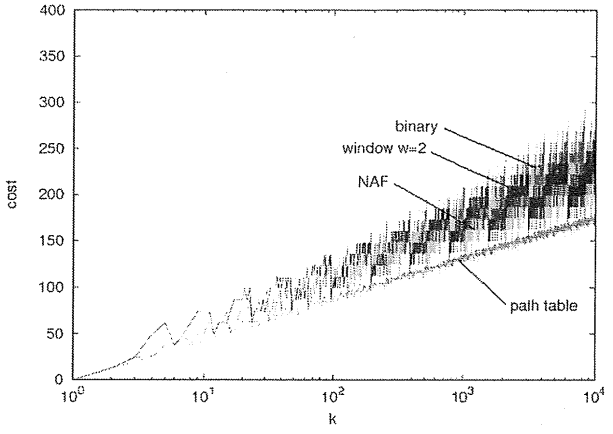
[1]ここでは，符号付バイナリとして付録に示す Non-Adjacent Form (NAF) を用いた．



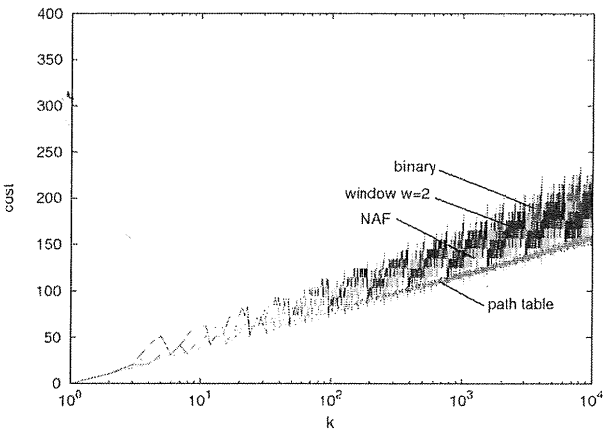

(a) $i/m = 13$ の場合




(b) $i/m = 11$ の場合




(c) $i/m = 9$ の場合




(d) $i/m = 7$ の場合

図 3: パステーブル法とアフィン座標系でのバイナリ法，ウィンドウ法とのコスト比較

ブル作成におけるアルゴリズムの途中で計算を終了できると仮定した．また，$k \leq 10^4$ では，ウィンドウ法のコストは $w = 2$ とする場合が最も平均コストが低いため，$w = 2$ として比較を行う．$i/m = 13, 11, 9, 7$，$w = 2$ とした場合のそれぞれの計算コストを図 3 に示す．

バイナリ法では桁数が $\lfloor \log_2 k \rfloor + 1$ であるため，スカラー倍算 $kP$ のコストは $(1i + 2s + 2m)\lfloor \log_2 k \rfloor$ と $(2i + 3s + 4m)\lfloor \log_2 k \rfloor$ の間にあり，平均はほぼ $(3i + 5s + 6m)\lfloor \log_2 k \rfloor/2$ となる[2]．一方，符号付バイナリ法のコストの平均は，およそ $(4i + 7s + 8m)\lfloor \log_2 k \rfloor/3$ となる．

図 3 より，$i/m$ が小さくなると，どの計算法のコストも下がることが示される．しかし，バイナリ法やウィンドウ法に対してパステーブル法を利用した場合のコスト低減効果は，$i/m$ が大きいほど高いことがわかる．その理由は，$P + Q$ と $2P$ 以外の基本演算 ($3P, 3P \pm Q$ など) は，アフィン座標系で逆元計算を減らすような工夫をしてコストを下げており，$i/m$ が大きいほどその効果が大きく現れるためである．$i/m$ が非常に小さい値の場合には，$P + Q$ と $2P$ 以外の基本演算を用いてもコストを下げる効果があまり得られないため，パステーブル法のコストがバイナリ法のコストに近くづくことが予想される．また，パステーブル法は，$k$ の値によるコストの分散が非常に小さいという特徴がある．

表 3 は，1 以上 10,000 以下の整数 $k$ の各々について，アフィン座標系でのバイナリ法，符号付バイナリ法およびウィンドウ法のうち最小値のコストを $C_{bw}^{(A)}(k)$ とし，パステーブル法のコスト $C_p(k)$ を比較して $C_{bw}^{(A)}(k) < C_p(k)$，$C_{bw}^{(A)}(k) = C_p(k)$，$C_{bw}^{(A)}(k) > C_p(k)$ となる $k$ の頻度を表したものである．この表より，$i/m = 13, 11, 9, 7$ のすべての場合に，99% 以上の $k$ 値においてパステーブル法のコストの方が低いことが示される．

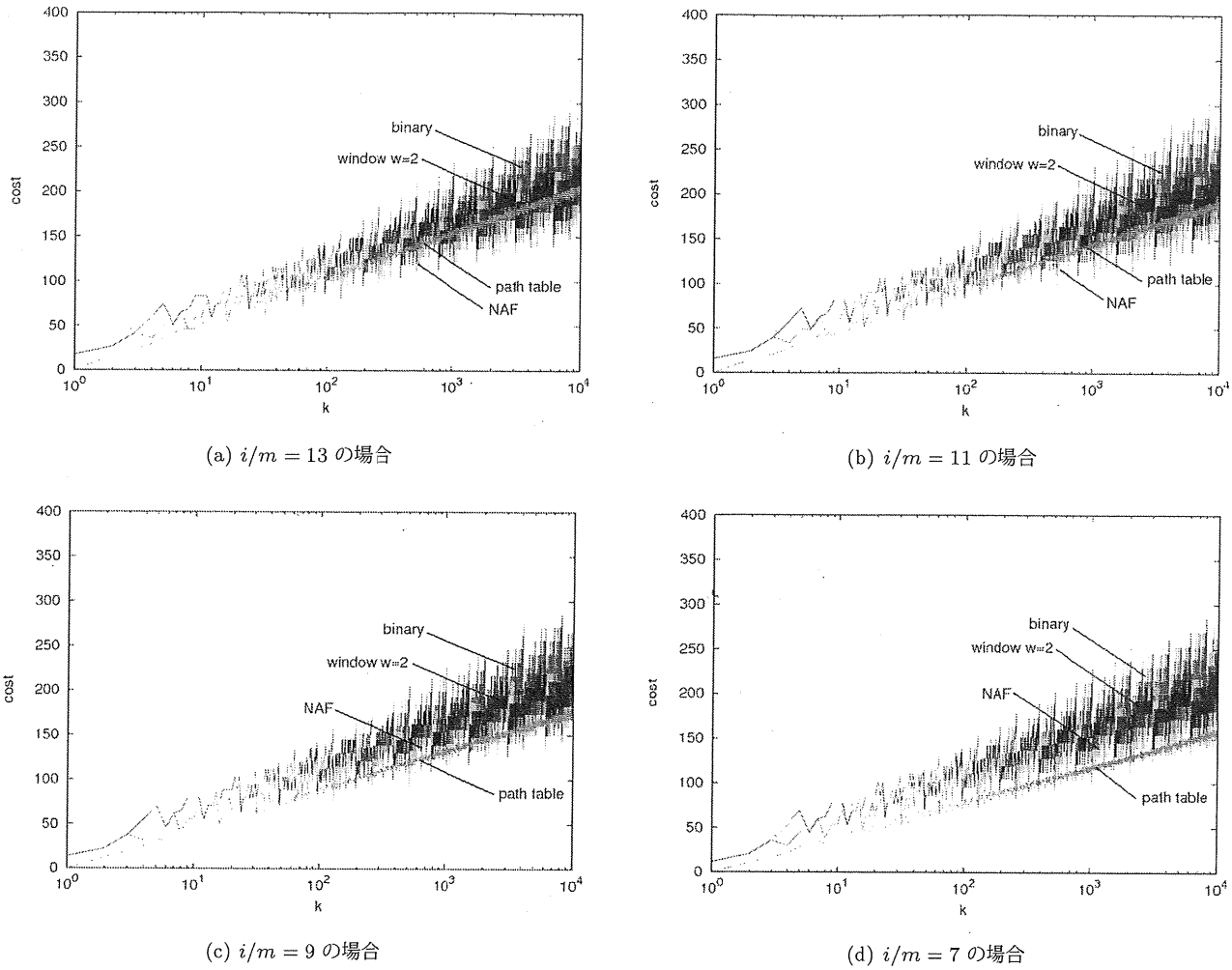
[2] $\lfloor x \rfloor$ は実数 $x$ を超えない最大の整数を表す．

表 3: アフィン座標系でのバイナリ法，ウィンドウ法とパステーブル法との比較

| $i/m$ | $k$ | $C_{bw}^{(A)}(k) < C_p(k)$ となる $k$ の頻度 | | $C_{bw}^{(A)}(k) = C_p(k)$ となる $k$ の頻度 | | $C_{bw}^{(A)}(k) > C_p(k)$ となる $k$ の頻度 | |
|---|---|---|---|---|---|---|---|
| 13 | 1〜100 | 0 | (0.00%) | 2 | (2.00%) | 98 | (98.00%) |
| | 101〜1000 | 0 | (0.00%) | 0 | (0.00%) | 900 | (100.00%) |
| | 1001〜10000 | 0 | (0.00%) | 0 | (0.00%) | 9000 | (100.00%) |
| | total | 0 | (0.00%) | 2 | (0.02%) | 9998 | (99.98%) |
| 11 | 1〜100 | 0 | (0.00%) | 2 | (2.00%) | 98 | (98.00%) |
| | 101〜1000 | 0 | (0.00%) | 0 | (0.00%) | 900 | (100.00%) |
| | 1001〜10000 | 0 | (0.00%) | 0 | (0.00%) | 9000 | (100.00%) |
| | total | 0 | (0.00%) | 2 | (0.02%) | 9998 | (99.98%) |
| 9 | 1〜100 | 0 | (0.00%) | 7 | (7.00%) | 93 | (93.00%) |
| | 101〜1000 | 0 | (0.00%) | 3 | (0.33%) | 897 | (99.67%) |
| | 1001〜10000 | 0 | (0.00%) | 4 | (0.04%) | 8996 | (99.96%) |
| | total | 0 | (0.00%) | 14 | (0.14%) | 9986 | (99.86%) |
| 7 | 1〜100 | 0 | (0.00%) | 7 | (7.00%) | 93 | (93.00%) |
| | 101〜1000 | 1 | (0.11%) | 2 | (0.22%) | 897 | (99.67%) |
| | 1001-10000 | 59 | (0.66%) | 16 | (0.18%) | 8925 | (99.17%) |
| | total | 60 | (0.60%) | 25 | (0.25%) | 9915 | (99.15%) |

表 4: 重み付き射影座標系でのバイナリ法，ウィンドウ法とパステーブル法との比較

| $i/m$ | $k$ | $C_{bw}^{(J)}(k) < C_p(k)$ となる $k$ の頻度 | | $C_{bw}^{(J)}(k) = C_p(k)$ となる $k$ の頻度 | | $C_{bw}^{(J)}(k) > C_p(k)$ となる $k$ の頻度 | |
|---|---|---|---|---|---|---|---|
| 13 | 1-100 | 22 | (22.00%) | 0 | (0.00%) | 78 | (78.00%) |
| | 101-1000 | 436 | (48.44%) | 3 | (0.33%) | 461 | (51.22%) |
| | 1001-10000 | 5768 | (64.09%) | 11 | (0.12%) | 3221 | (35.79%) |
| | total | 6226 | (62.26%) | 14 | (0.14%) | 3760 | (37.60%) |
| 11 | 1-100 | 15 | (15.00%) | 0 | (0.00%) | 85 | (85.00%) |
| | 101-1000 | 221 | (24.56%) | 21 | (2.33%) | 658 | (73.11%) |
| | 1001-10000 | 2945 | (32.72%) | 11 | (0.12%) | 6044 | (67.16%) |
| | total | 3181 | (31.81%) | 32 | (0.32%) | 6787 | (67.87%) |
| 9 | 1-100 | 7 | (7.00%) | 0 | (0.00%) | 93 | (93.00%) |
| | 101-1000 | 55 | (6.11%) | 0 | (0.00%) | 845 | (93.89%) |
| | 1001-10000 | 989 | (10.99%) | 16 | (0.18%) | 7995 | (88.83%) |
| | total | 1051 | (10.51%) | 16 | (0.16%) | 8933 | (89.33%) |
| 7 | 1-100 | 0 | (0.00%) | 0 | (0.00%) | 100 | (100.00%) |
| | 101-1000 | 3 | (0.33%) | 0 | (0.00%) | 897 | (99.67%) |
| | 1001-10000 | 74 | (0.82%) | 9 | (0.10%) | 8917 | (99.08%) |
| | total | 77 | (0.77%) | 9 | (0.09%) | 9914 | (99.14%) |




(a) $i/m = 13$ の場合　　(b) $i/m = 11$ の場合

(c) $i/m = 9$ の場合　　(d) $i/m = 7$ の場合

図 4: パステーブル法と重み付き射影座標系でのバイナリ法，ウィンドウ法とのコスト比較

# 4　重み付き射影座標系でのコスト比較

ここでは，射影座標系の一種である重み付き射影座標系 [4] でのバイナリ法，符号付バイナリ法とウィンドウ法のコストを評価し，パステーブル法と比較する．

有理点 $P$ がアフィン座標系で $(x, y)$ と表されるとき，重み付き射影座標系では $(X, Y, Z)$，$x = X/Z^2, y = Y/Z^3$ と表される．アフィン座標系の点 $P = (x, y)$ は重み付き射影座標系の点 $P = (x, y, 1)$ に自明に変換できる．一方，重み付き射影座標系の点は $1i + 1s + 4m$ のコストでアフィン座標系の点に変換される．

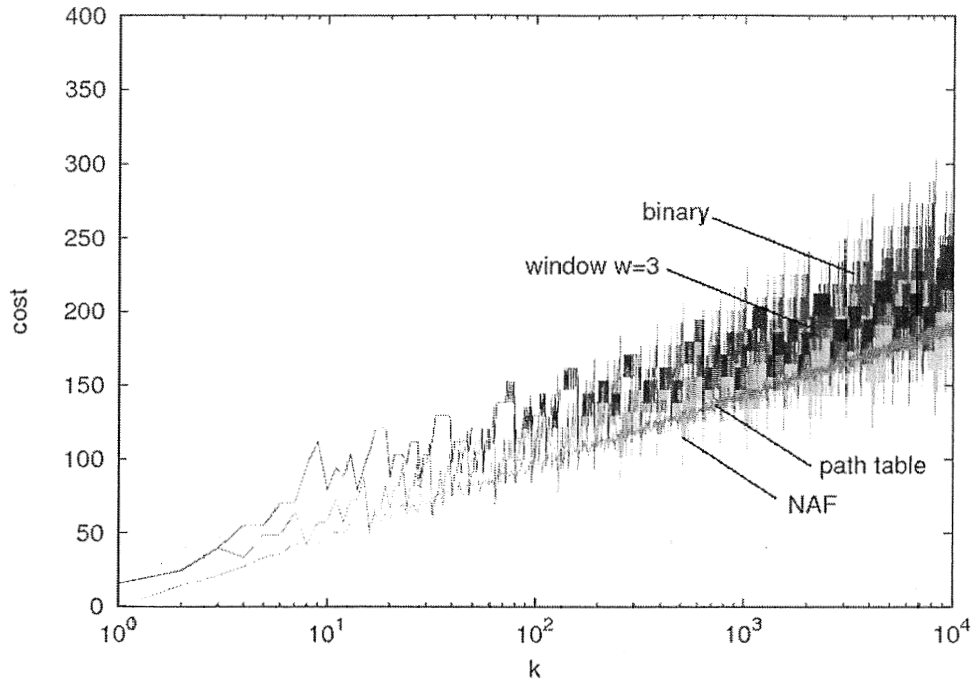
点 $P, Q$ の座標が重み付き射影座標系で与えられたとき，加算 $P + Q$ のコストは $4s + 12m$ で与えられ，二倍算 $2P$ のコストは $6s + 4m$ で与えられる．アフィン座標系と異なり，加算および 2 倍算に逆元計算は含まれない．

射影座標系でのバイナリ法，符号付バイナリ法とウィンドウ法 ($w = 2$) およびパステーブル法の計算コストを求め，図 4 に示す．この図から，アフィン座標系の場合と同様に，パステーブル法は $k$ の値によるコストの分散が非常に小さいことがわかる．

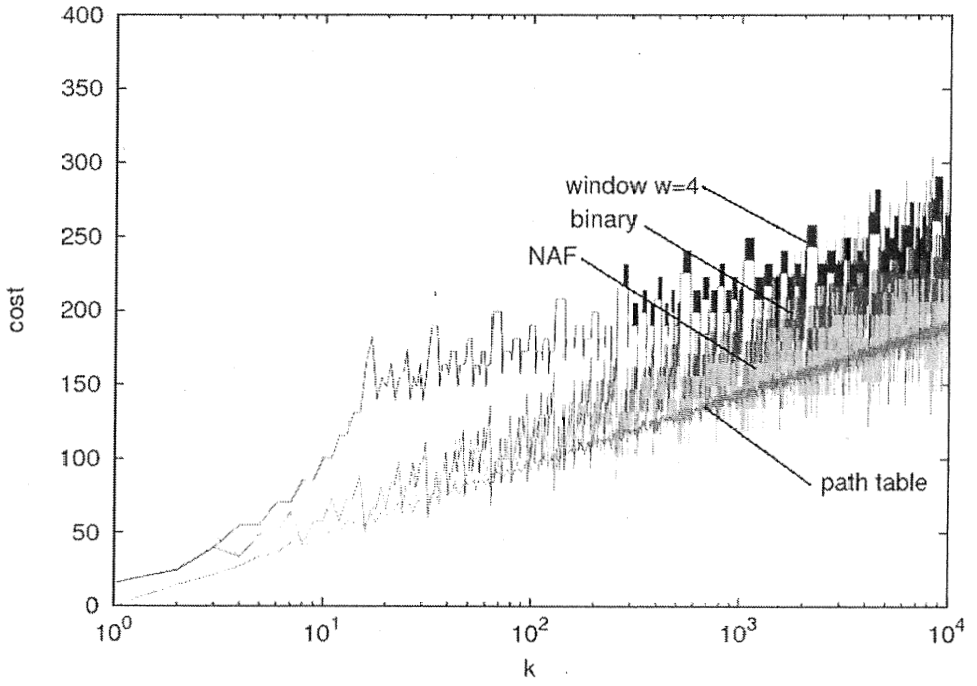
射影座標系の場合，計算の最後でアフィン座標系への変換の際に一度だけ逆元計算が行われるため，$i/m$ の値はコストにわずかにしか影響しない．一方，パステーブル法の平均コストは $i/m$ の値に依存して変わる．このため，$i/m$ が大きいときは，符号付バイナリ法に対してパステーブル法でのコスト低減効果が小さく，$i/m$ が小さいときにパステーブル法での効果が大きくなる．

図 5 に $i/m = 11$ においてウィンドウ法におけるウィンドウ幅を $w = 3, 4$ とした場合のコストを示す．$w = 4$ においてはパステーブル法よりウィンドウ法の傾きが小さいことから，$k$ が非常に大きくなるとウィンドウ法のコストの方が低くなることが予想される．

表 4 は，1 以上 10,000 以下の整数 $k$ の各々について，




(a) $w = 3$ の場合




(b) $w = 4$ の場合

図 5: ウィンドウ法の異なるウィンドウ幅によるコストの比較 $(i/m = 11)$

重み付き射影座標系でのバイナリ法，符号付バイナリ法及びウィンドウ法のうち最小値のコストを $C_{bw}^{(J)}(k)$ とし，パステーブル法のコスト $C_p(k)$ と比較をして $C_{bw}^{(J)}(k) < C_p(k)$, $C_{bw}^{(J)}(k) = C_p(k)$, $C_{bw}^{(J)}(k) > C_p(k)$ となる $k$ の頻度を表したものである．この表より，パステーブル法のコストの方がバイナリ法および符号付バイナリ法のコストより低くなるような $k$ の頻度は，$i/m$ が小さいほど高くなることが示される．

## 5 むすび

本研究では，複数の基本演算が利用できる場合に，ある整数 $k_{max}$ までの各々の $k$ に対して，最小コストを与えるパスをあらかじめ求めてパステーブルに記憶しておき，スカラー倍算を行う際にそれを利用する方法について検討した．逆元計算のコスト $i$ を変えた幾つかの条件において，パステーブルから直接与えられるパスのコストを求め，アフィン座標系でのバイナリ法，符号付バイナリ法とウィンドウ法，および重み付き射影座標系でのバイナリ法，符号付バイナリ法とウィンドウ法のコストと比較を行った．

コスト評価の結果，バイナリ法やウィンドウ法に対して，パステーブル法のコストは，$k$ 値に対する分散が非常に小さいことが示された．また，アフィン座標系でのバイナリ法およびウィンドウ法と比較した場合，$i/m$ の値が 7 以上の場合には $i/m$ の値にかかわらず，99% 以上の $k$ 値に対してパステーブル法のコストの方が低いことが示された．重み付き射影座標系では，$i/m$ が 9 以下の場合には 89% 以上の $k$ 値に対してバイナリ法およびウィンドウ法のコストよりパステーブル法のコストの方が低いが，$i/m$ が大きくなるとその比率が小さくなることが示された．

## 参考文献


[1] N. Kunihiro and H. Yamamoto, "Window and Extended Window Methods for Addition Chain and Addition-Subtraction Chain", IEICE Trans. Fundamentals, vol.E81-A, no.1, Jan. 1998.

[2] Y. Tsuruoka, "Computing Short Lucas Chains for Elliptic Curve Cryptosystems", IEICE Trans. Fundamentals, vol.E84-A, no.5, pp.1227–1233, 2001.

[3] 三宅, 宮地, "楕円曲線暗号におけるスカラー倍算の高速化に関する考察", 信学技報, Vol.101. No.726, pp.69-74, IT2001–86, 2002.

[4] 安達, 浦生, 平田, "直接計算による楕円曲線上のスカラ倍計算の効率化", 信学論, Vol.J88-A, No.1, pp54–61, 2005.

[5] M. Ciet, M. Joye, K. Lauter and P. L. Montgomery, "Trading Inversions for Multiplication in Elliptic Curve Cryptography", in Designs, Codes and Cryptography, 39, pp.189–206, 2006.

[6] 溝添, 宮地, 亀井, "3 つの基底を用いた効率的な楕円ベキ倍算", 信学技報, ISEC2006–130, 2007.

[7] 寺嶋, 松嶋, 足原, "複数基底を用いる楕円スカラー倍算のための最小コスト木に関する検討", 2008 年暗号と情報セキュリティシンポジウム (SCIS2008), 2008.

[8] 寺嶋, 松嶋, 足原, "楕円曲線暗号におけるスカラー倍算の効率化に関する検討", 信学技報, ISEC2007–72, 2008.

[9] 松嶋, 相良, 櫻木, 足原, "楕円曲線暗号におけるスカラー倍算の効率化に関する検討", 職業能力開発総合大学校紀要, 第 40 号 A, pp.77–86, 2011.

[10] 松嶋, 相良, 足原, "パステーブルを用いた楕円スカラー倍算における高階差分演算に関する考察", 信学技報, ISEC2008–11, 2008.

[11] 櫻木, 相良, 松嶋, 足原, "パステーブルを用いる楕円スカラー倍算の計算量に関する検討", 2008 年 情報理論とその応用シンポジウム (SITA2008), 2008.

[12] 相良, 櫻木, 松嶋, 足原, "パステーブルを用いる楕円スカラー倍算の計算量の評価", 2009 年 暗号と情報セキュリティシンポジウム (SCIS2009), 2009.

[13] KNUTH 著, 中川 訳, 準数値算法, サイエンス社, 1981.

[14] D. M. Gordon, "A survey of fast exponentiation methods", J. Algorithms, 27,vol.1, pp.129-146, 1998.


### 付 録

[NAF 変換アルゴリズム]

入力: 整数 $k = \sum_{j=0}^{l-1} k_j 2^j, k_j \in \{0, 1\}$
出力: NAF $s = \sum_{j=0}^{l} s_j 2^j, s_j \in \{-1, 0, 1\}$

1. $j = 0$
2. while $k \geq 1$
3. If $k$ is odd then $s_i = 2 - (k \mod 4), k = k - s_j$
4. else $s_j = 0$
5. $k = k/2, j++$
6. Return $s$