社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－CSEC－42（34）
2008 / 7 / 25

ペアリング暗号に効果的な拡大体上べき乗算に関する一考察

吉田 知輝† 加藤 英洋† 根角 健太† 野上 保之† 森川 良孝†

† 岡山大学大学院自然科学研究科 〒700-8530 岡山県岡山市津島中 3-1-1
E-mail: †{yoshida,kato,nekado,nogami,morikawa}@trans.cne.okayama-u.ac.jp

あらまし 近年，グループ署名 [1] や ID-based 暗号 [3] といった楕円曲線上の双線形写像（ペアリング）[2] に基づく暗号方式が注目されている．これらの暗号方式は暗号化・復号・署名などの処理の際に，ペアリング計算後の拡大上の乗法群 $\mathbb{G}_T$ における演算（べき乗算）を必要とする．暗号に用いられるべき乗算は指数部が大きく計算に時間を要するため，本稿では Barreto-Naehrig（BN）曲線 [9] とよばれる，埋め込み次数 $k = 12$ をもつ非超特異なペアリングフレンドリ曲線を用いた場合に組織的に与えられる整数 χ で指数部を展開することでべき乗算を高速化する手法を提案する．加えて，本稿ではこれまでに提案されている Binary 法や NAF 法，Window 法，Avanzi 法 [12] [15] といった拡大体上のべき乗算アルゴリズムと提案法を組み合わせた実装および検証を行う．とくに NAF 法との組み合わせによってべき乗する際の指数部のハミング重みに大きく依存せず高速に計算できることを示す．
キーワード ペアリング，べき乗算，拡大体，Barreto-Naehrig 曲線

A Consideration on Efficient Exponentiation in Extension Field for Pairing-based Cryptography

Tomoki YOSHIDA†, Hidehiro KATO†, Kenta NEKADO†,
Yasuyuki NOGAMI†, and Yoshitaka MORIKAWA†

† Graduate School of Natural Science and Technology, Okayama University 3-1-1, Tsushima-naka, Okayama, Okayama 700-8530, Japan
E-mail: †{yoshida,kato,nekado,nogami,morikawa}@trans.cne.okayama-u.ac.jp

Abstract In recent years, pairing-based cryptographies [2] such as ID-base cryptography [3] and group signature [1] have been studied. These cryptographies require exponentiations in multiplicative group $\mathbb{G}_T$. In this paper, the author proposes an efficient exponentiation method for the case of Barreto-Naehrig (BN) curve [9] . Additionally, the author shows implementation results of the proposed method with conventional techniques such as binary method, window method, NAF method and Avanzi method [12] [15]. Then, the author shows the proposed method carries out exponentiation fast.
Key words pairng, exponentiation, extension field, Barreto-Naehrig curve

1. まえがき

近年，グループ署名 [1] や ID-based 暗号 [3] といった楕円曲線上の双線形写像（ペアリング）[2] に基づく暗号方式が注目されている．これらの暗号方式は暗号化・復号・署名などの処理の際に，素数位数の楕円曲線上の有理点群 $\mathbb{G}_1$，$\mathbb{G}_2$ およびペアリング計算後の拡大上の乗法群 $\mathbb{G}_T$ における演算を必要とする．とくに乗法群 $\mathbb{G}_T$ では拡大体におけるべき乗算を必要とし，暗号に用いられるべき乗算は指数部が大きく計算に時間を要するため，本稿ではこれを効率よく行うための手法を提案する．

本稿では Barreto-Naehrig（BN）曲線 [9] とよばれる，埋め込み次数 $k = 12$ をもつ非超特異なペアリングフレンドリ曲線を用いた Ate ペアリングについて取り扱う．BN 曲線は定義体の標数 p およびフロベニウスのトレース t が整数 χ を用いた多項式 $p(\chi)$ および $t(\chi)$ で組織的に与えられる．これまでに $\mathbb{G}_T$ 上のべき乗算に対して $t-1$ 乗がフロベニウス写像 1 回，つまりビットシフトのみで行うことができる性質を利用し，指数部を $t-1$ 進展開することで高速に計算する手法が提案されてきた [13]．本稿ではこの手法を改良し，BN 曲線の組織的に与えられる整数 χ により指数部を展開することでさらなる高速化を図る．加えて，本稿ではこれまでに提案されている Binary 法や NAF 法，Window 法，Avanzi 法 [12] [15] といった拡大体上

のべき乗算高速化アルゴリズムと提案法を組み合わせた実装を行う．とくに NAF 法との組み合わせによってべき乗する際の指数部のハミング重み(注1)に依存せず高速に計算できることを示す．以下，とくに断らない限り，$\#_{\mathrm{MUL}}$ および $\#_{\mathrm{SQ}}$ はそれぞれ $\mathbb{G}_T$ 上での乗算回数および二乗算の回数を，$H_w(n)$ をビット列 n のハミング重みを表す．

2. 数学的準備

本章では Ate ペアリング [4] と，BN 曲線 [9] について簡単に説明した後，べき乗算アルゴリズムについて復習する．

2.1 フロベニウス写像

q を素数 p のべき乗とし，楕円曲線の有理点 $P=(x,y)$ に対するフロベニウス写像 ϕ_{q^i} は以下の式で定義される．

$$\phi_{q^i}:(x,y)\mapsto(x^{q^i},y^{q^i}), \tag{1}$$

定義体が $\mathbb{F}_{p^k}$ の場合には，$\phi_{q^k}(P)=P$ となる．$\mathbb{F}_{q^k}$ の基底として正規基底を用いた場合，$\mathbb{F}_{q^k}$ の元 x に関するフロベニウス写像 $\psi(x)\mapsto x^q$ は計算なしに求めることができる．

2.2 Ate ペアリング

本稿では，埋め込み次数 $k=12$ の Barretto-Naehrig (BN) 曲線を用いた Ate ペアリング [4] を扱う．$E(\mathbb{F}_q)$ を有限体 $\mathbf{F}_q$ 上で定義される楕円曲線上の有理点が成す加法群，$E[r]$ を位数が素数 r である有理点の集合とする．$[i]$ を有理点を i 倍する写像とする．このとき，Ate ペアリング e は非退化な双線形写像として次式で定義される．

$$\begin{aligned}&\mathbb{G}_1=E[r]\cap\mathrm{Ker}(\phi_q-[1])\\&\mathbb{G}_2=E[r]\cap\mathrm{Ker}(\phi_q-[q])\\&e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T.\end{aligned} \tag{2}$$

ただし，$\mathbb{G}_T$ は k 次拡大体乗法群の位数 r の部分群である．

2.3 BN 曲線

Barreto-Naehrig (BN) 曲線 [9] とよばれる，埋め込み次数 12 をもつ非超特異なペアリングフレンドリ曲線が知られている [9]．BN 曲線は $y^2=x^3+a$, $a\in\mathbb{F}_p$ の式で与えられ，標数 p およびフロベニウスのトレース t は，χ を整数として以下の式で与えられる．

$$\begin{aligned}p(\chi)&=36\chi^4-36\chi^3+24\chi^2-6\chi+1\\t(\chi)&=6\chi^2+1,\end{aligned} \tag{3}$$

整数 χ は負の数でもよい．本稿では主に素数位数 $\#E(\mathbb{F}_p)$ をもつ BN 曲線を用いた Ate ペアリングの $\mathbb{G}_T$ におけるべき乗算の高速化について考える．

2.4 Binary 法

べき乗算の高速化アルゴリズムとしてよく知られているものに Binary 法がある [12]．これは指数部を 2 のべき乗の和で表し，計算過程に現れる 2 のべき乗を使いまわすことで高速化するものである．Binary 法はビット列を見る向きによって右向き Binary 法，左向き Binary 法がある．例えば A^{77} は左向き Binary 法を用いると (4) のように求められる．

$$A^{77}=A^{1001101_2}=A^{2^6+2^3+2^2+2^0}=A^{64}\cdot A^8\cdot A^4\cdot A \tag{4}$$

同様に右向き Binary 法を用いると (5) のように求められる．

$$A^{77}=A^{1001101_2}=((((((A)^2)^2)^2\cdot A)^2\cdot A)^2)^2\cdot A \tag{5}$$

本節以降に説明する Binary 法を応用したアルゴリズムでは，どちらか一方のみ利用できるものもある．この方法では $\lfloor\log_2 n\rfloor$ 回の二乗算と $Hw(n)-1$ 回の乗算でべき乗を行うことができる．そのため，指数部が同じビット数でもハミング重みに応じて乗算回数が増える．

2.5 Window 法

Binary 法を改良したべき乗算法に Window 法がある [12]．主たるアイデアは Window Size (以下 WS) という値 w を決めて ビット列を WS 毎に分割し，乗算の回数を減らすというものである．2^w-1 までのべき乗を計算し，あらかじめメモリに保存しておく．そして，ビット単位ではなく Window 単位で右向き Binary 法を考える．

例えば $w=2$ のとき A^{157} は，あらかじめ A^{10_2},A^{11_2} をメモリに保存しておき，以下のように求める．

$$A^{147}=A^{10011101_2}=(((A^{10_2})^{2^2}\cdot A)^{2^2}\cdot A^{11_2})^{2^2}\cdot A \tag{6}$$

2.6 Adaptive Window 法

Window 法を改良したべき乗算法に Adaptive Window 法がある．これは Window 内のビット列の先頭が 0 の場合に二乗算一回のみを行い，Window を一つずらし乗算回数およびメモリに保存しておくデータを減らす手法である．

Window 法では Window に 0 から 2^w-1 までの数が入ることを想定するが，Adaptive Window 法では Window には先頭が 1 のものしか入らないため，2^{w-1} から 2^w-1 のべき乗のみを計算してメモリに保存しておく．例えば $w=2$ のとき A^{153} は (7) のように求めることができる．

$$A^{153}=A^{10011001_2}=((((A^{10_2})^{2^2})^2\cdot A^{11_2})^{2^2})^2\cdot A \tag{7}$$

2.7 Avanzi 法

precompute A^{10},A^{11}

		1	0	1	0	1	0	$0_2:n_0$
		1	1	0	1	1	0	$1_2:n_1$

*STEP*1 $A^{10}\cdot A^{11_P}$

1	0	1	0	1	0	1	0	$0_2:n_0$
1	1	0	1	1	0	0	1	$1_2:n_1$

*STEP*2 $(A^{10}\cdot A^{11_P})^{2^2}\cdot A^{10}\cdot A^P$

1	0	1	0	1	0	1	0	$0_2:n_0$
1	1	0	1	1	0	0	1	$1_2:n_1$

*STEP*3 $((A^{10}\cdot A^{11_P})^{2^2}\cdot A^{10}\cdot A^P)^{2^2}\cdot A^{10}\cdot A^{10_P}$

⋮

図 1 Avanzi 法の例 ($w=2$)

(注1)：ビット列中の 1 の数

正規基底表現を用いて表される拡大体上の元のように，フロベニウス写像 ψ を高速に行えるとき，Avanzi の提案したフロベニウス写像 ψ を多用する手法によりべき乗算を効率的に行うことができる [15]．まず，求めるべき乗算の指数部 n を p 進展開する．

$$A^n = A^{n_0} \cdot A^{n_1 \cdot p} \cdots A^{n_{m-1} \cdot p^{m-1}} \tag{8}$$

そして図 1 のように p 進展開をしたべき乗数のビット列(注2)を縦に並べて WS の横方向と p 進展開された n が並ぶ縦方向を包むような Big Window(以下 BW) を用意し，BW 内のべき乗を先に計算した後，二乗算を行う．つまり，$A^{n_0}, A^{n_1}, \cdots, A^{n_{m-1}}$ をそれぞれ Window 法で計算した場合に重複する二乗算を，先に縦方向の乗算を行うことで一つにまとめ高速化する手法である．このとき，通常の Window 法同様にあらかじめ Window の $2^w - 1$ までのべき乗を求めてメモリに入れておく．

2.8 Adaptive Avanzi 法

Avanzi 法では p 進展開後の各べき乗算に対して Window 法を適用し，重複する二乗算を一つにまとめることで高速化を図った．同様にに各べき乗算を Adaptive Window 法で計算し，Avanzi 法同様に重複する二乗算をまとめることで効率的にべき乗算を行うことができる．

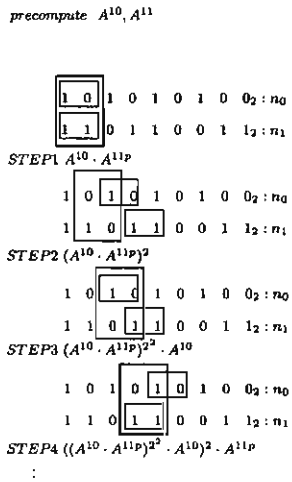

図 2 Adaptive Avanzi 法の例 $w = 2$

この方法では，Adaptive Window 法同様に 2^{w-1} から $2^w - 1$ のべき乗を計算し，メモリに保存した後に，図 2 の例のように p 進展開をしたべき乗数を縦に並べてそれぞれのビット列に Small Window(以下 SW) という Window を用意する．この SW は大きさが WS で Adaptive Window 法の Window と同様に掛け合わせる処理をした後，右に WS 分シフトする．一方 Big Window(以下 BW) は大きさが WS で一回の処理ごとに右に ひとつシフトしていく Window である．この BW が一つシフトするごとに二乗算を一回行う．BW の中に SW がいる場合にはあらかじめ計算してメモリに入れた値に対して適切なフロベニウス写像 ψ を行った後に掛け合わせる．その後 SW を WS 分右にシフトする．ただし，SW 内の先頭ビットが 1 でない場合はさらに右にシフトする．この処理を BW が LSB にたどり着くまで繰り返すことによってべき乗算を求めることができる，

(注2)：図では 2 列で $A = A^{n_0} \cdot A^{n_1 p}$ の場合

3. 提案するべき乗算法

本章では，ペアリング計算後の G_T 上のべき乗演算を高速に行う手法を提案する．これまでにべき乗の指数部を $t-1$ 進展開し，$t-1$ 乗をフロベニウス写像 ψ で置き換え，高速化する手法が提案されている．[13] 本稿では，用いる楕円曲線が BN 曲線の場合において，指数部を BN 曲線のパラメータ χ によって展開することでより高速なべき乗算を行う手法を提案する．

3.1 $t-1$ 進展開

楕円曲線上の有理点の位数 r は標数 p およびフロベニウスのトレース t を用いて $r = p + 1 - t$ のように表されることから，

$$p \equiv t - 1 \pmod{r}. \tag{9}$$

G_T の元を A とすると，G_T の位数は r であり，$A^r = 1$ より，

$$A^{t-1} = A^p = \psi(A) \tag{10}$$

となる．Hasse の定理より $|t-1| < 2\sqrt{p}$ であることから，この関係を用いて $t-1$ 乗を 1 回のフロベニウス写像 ψ で置き換えることで，べき乗算を効率的に行うことができる．

3.2 提案法： 6χ 進展開

用いる楕円曲線が BN 曲線の場合，指数部を BN 曲線のパラメータ χ を用いて展開することにより，さらに効率的にべき乗算を行うことができる．

$p^{12} - 1$ を以下のように因数分解する．

$$p^{12} - 1 = (p^6 - 1)(p^6 + 1) \tag{11a}$$
$$= (p^6 - 1)(p^2 + 1)(p^4 - p^2 + 1) \tag{11b}$$

本稿では埋め込み次数 12 の場合を取り扱っているため，i を $i < 12$ として p と r の間に以下の関係が成り立つ．

$$r \mid (p^{12} - 1),\ r \nmid (p^i - 1) \tag{12}$$

すなわち以下の式が成り立つ．

$$r \mid p^4 - p^2 + 1 \tag{13a}$$
$$p^4 - p^2 + 1 \equiv 0 \pmod{r} \tag{13b}$$

また，(9) と (3) より，$t-1$ は以下のように表せる．

$$p \equiv t - 1 \equiv 6\chi^2 \pmod{r} \tag{14}$$

さらに，(13b) を用いて以下の式を得る．

$$p^2(1-p)(1+p) \equiv 1 \pmod{r}$$
$$(1+p)^{-1} \equiv p^2(1-p) \pmod{r} \tag{15}$$

ここで (3) の関係より，

$$p \equiv p^2 - 6\chi(p+1) + 4p + 1 (\mathrm{mod}\ r), \tag{16}$$

$$6\chi \equiv (p^2 + 3p + 1)(p+1)^{-1} \ (\mathrm{mod}\ r), \tag{17}$$

$$6\chi \equiv \{(p+1)^2 + p\}(p+1)^{-1} (\mathrm{mod}\ r), \tag{18}$$

(15) より，以下の式を得る．

$$6\chi \equiv p + 1 + p^3(1-p) \pmod{r}. \tag{19}$$

$6\chi \approx \sqrt{|t-1|}$ であり，この関係を用いて 6χ 乗を 3 回のフロベニウス写像と 3 回の乗算で置き換えることでべき乗算を効率的に行うことができる．ただし，(19) の右辺に 1 回の逆元演算を必要とする．拡大体の元同士の逆元演算は拡大体上乗算数回分の演算コストを要するため，(19) をそのまま適用した場合，十分な効果が得られない．そこで，(19) を逆元演算を必要としない形に変形することを考える．

(11a) および (12) より以下の式を得る．

$$p^6 + 1 \equiv 0 \pmod{r} \tag{20}$$

よって，

$$A^{p^6+1} \equiv 1 \pmod{r}$$

$$A^{-1} \equiv A^{p^6} \pmod{r} \tag{21}$$

のように変形できる．この (21) に表される関係を用いて (19) を変形すると，

$$A^{6\chi} = A^{p+1+p^3(1+p^7)} \tag{22}$$

となり，逆元演算が不要となる．さらに $p^{12} \equiv 1 \pmod{r}$ の関係を用い (22) を式変形する．

$$A^{6\chi} = A^{(1+p^3)^1+p^{10}} \tag{23}$$

(23) のように，6χ 乗を 2 回のフロベニウス写像と 2 回の乗算で置き換えることで高速化できる．

3.3 Binary 法の適用

指数部を以下のように $t-1$ 進展開する．

$$A^n = A^{n_0} \cdot A^{n_1(t-1)}. \tag{24}$$

そして A^{n_0}，A^{n_1} をさらに 6χ 進展開し，Binary 法を適用することで乗算回数を減らす方法を示す．

$$A^n = A^{N_0} \cdot A^{N_1 \cdot 6\chi} \cdot A^{N_2(t-1)} \cdot A^{N_3 \cdot 6\chi(t-1)} \tag{25}$$

$$(n_0 = N_0 + N_1 \cdot 6\chi,\ n_1 = N_2 + N_3 \cdot 6\chi)$$

まず，$A^{N_0} \sim A^{N_3}$ をかけあわせる順番を入れ替え，6χ 乗する回数を 1 回に減らす．

$$A^n = A^{N_0} \cdot (A^{N_1} \cdot A^{N_2 \cdot (t-1)})^{6\chi} \cdot A^{N_2(t-1)} \tag{26}$$

そして $A^{N_0} \sim A^{N_3}$ を求める際に Binary 法を適用する．$A^{2^i} (1 \leq i \leq \lfloor \log_2(6\chi) \rfloor - 1)$ を求める部分については 1 回目の処理のときにメモリに保存する．最後に (23) の関係を用い，6χ 乗をフロベニウス写像 2 回と乗算 2 回に置き換える．

3.4 NAF 法

Non-adjacent form 法（NAF 法）は拡大体の A の逆元 A^{-1} を効果的に用いることにより Binary 法を改良した手法である．はじめにべき乗の指数部を 2 進数展開した際に，1 が続くビット列を以下のように符号付きで表現する．

$$\{1110101_2\} \rightarrow \{100\bar{1}0101_2\} \tag{27}$$

そして A の逆元 A^{-1} を求め，右向き Binary 法を用いて $\bar{1}$ がたっているビットに関しては A ではなく A^{-1} を乗ずる．A の逆元は (21) の関係より，フロベニウス写像 ψ を用いて容易に求めることができる．この手法により，ハミング重みを小さくすることで乗算回数を減らし，べき乗算を高速に行うことができる．本稿ではこの NAF 法を次節で示す Adaptive Avanzi 法に適用し，ハミング重みに大きく依存せず高速な実装を行う．

3.5 Adaptive Avanzi 法の適用

べき乗算の指数部を (25) のように展開する．その後前章で説明した Adaptive Avanzi 法を適用する．

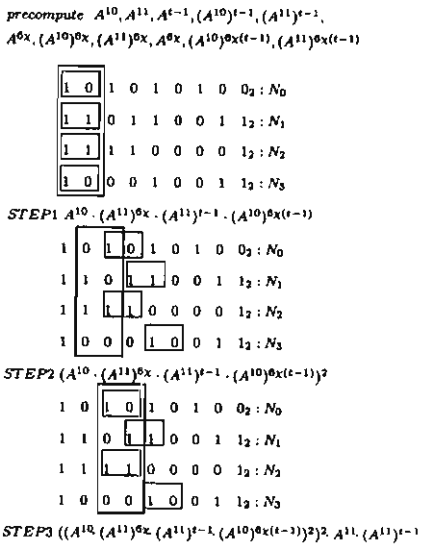

図 3　提案法+Adaptive Avanzi 法の例 ($w = 2$)

従来の Adaptive Avanzi 法ではフロベニウス写像 ψ が高速に行えるという前提の元に高速化を図っていた．しかし提案法でこの手法をそのまま用いるとフロベニウス写像 1 回と同様の $t-1$ 乗の他にフロベニウス写像 2 回，乗算 2 回に相当する 6χ 乗を多数行わなければならず，最大限の効果が期待できない．そこで図 3 に示すように，あらかじめメモリに 2^{w-1} から $2^w - 1$ のべき乗に加え，それらを $t-1$ 乗，6χ 乗，$6\chi(t-1)$ 乗したものをフロベニウス写像および乗算により求め，保存しておく．そして必要な際に乗ずることで高速化を図ることができる．

4. コスト評価とシミュレーション

本章では提案法および従来法を組み合わせたべき乗法をソフトウェア実装した結果を示す.

4.1 べき乗算に必要なコスト

160 ビットのランダムな整数 n に対し, G_T 上の元を n 乗するときに必要な拡大体上乗算および二乗算の回数を計測した. 具体的なパラメータ設定は表 1 に示す.

まず図 4 に従来のべき乗算高速化アルゴリズムを用いた際に必要な乗算および二乗算の回数を指数部の全体のビット数に対するハミング重みの割合毎に計測したものを示す. $\#MUL$, および $\#SQ$ はそれぞれ一回のべき乗算に必要な拡大体上の乗算および二乗算の回数を表す. Binary 法を用いた場合には, ハミング重みが大きくなるのに従い乗算回数が増えるが, NAF 法を用いた場合にはハミング重みが大きくなったとしても乗算回数が増えないのが読み取れる.

次に $t-1$ 進展開する手法をこれらのアルゴリズムと組み合わせ, 乗算および二乗算の回数を実測した結果を図 5 に示す. 従来法と比較すると, 二乗算の回数が半分となる. さらに, Widnow 法を応用した Avanzi 法, Adaptive Avanzi 法により, 乗算回数も提案法適用前と比較しても増えていない.

最後に図 6 に示すように, 提案法を用いた場合, 二乗算の回数は $t-1$ 進展開する手法と比較して半分に削減できる. さらに NAF 法と Adaptive Avanzi 法との組み合わせにより, 指数部のハミング重みに大きく依存することなく乗算, 二乗算の回数を減らすことができる. また, 図 4, 図 5, 図 6 のそれぞれにおいて各手法の二乗算の回数は $\log_2 n$ 回, $\log_2(t-1)$ 回, $\log_2(6\chi)$ 程度であったので, 一つにまとめている.

4.2 ソフトウェア実装と検証

従来のべき乗算高速化アルゴリズムおよび提案法を組み合わせソフトウェア実装を行った結果を示す. 実装には Pentium4(3.6GHz), C++言語の多倍長ライブラリである NTL を用いて実装を行った [8]. 表 2 に拡大体上乗算および二乗算一回にかかった時間を示す.

表 2 G_T 上の乗算・二乗算一回あたりの計算時間

	二乗算	乗算
time[sec]	0.0078	0.0085

実装環境 :Pentium4 3.6GHz, C++, NTL Library 使用

n を 160 ビットの整数とし, ランダムに 10000 万回値を変えた A^n を計算するのににかかった時間を平均し比較した. まず, 従来のべき乗算高速化アルゴリズムを実装した結果を図 7 に示す. Binary 法ではハミング重みが大きくなると計算時間が増えるが, NAF 法によって計算時間を平均化できている. また, Window 法や Adaptive Window 法を用いたことで乗算回数を削減されたことから計算時間が減っている. 次に $t-1$ 進展開した後に従来のべき乗算アルゴリズムと組み合わせ, 実装を行った結果を図 8 に示す. 前節で示したように二乗算の回数が半分になったため, 全体的に高速化できているのがわかる.

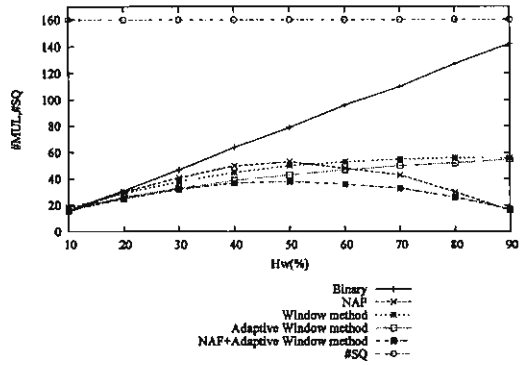

図 4 従来のべき乗算高速化手法に要する乗算・二乗算の回数

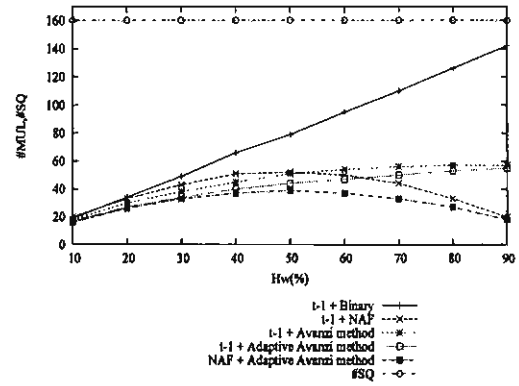

図 5 従来のべき乗算高速化手法+$t-1$ 進展開に要する乗算・二乗算の回数

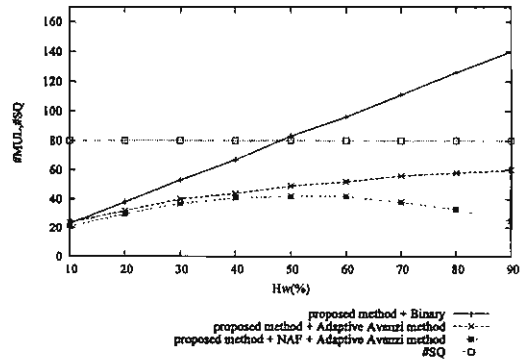

図 6 従来のべき乗算高速化手法+提案法に要する乗算・二乗算の回数

表 1　パラメータ設定

p	2063276713607373024910158007441390334505910272l9
t	454233058419889982668807
χ	275146342401

最後に提案法を従来のべき乗算アルゴリズムと組み合わせ，実装行った結果を図 7 に示す．$t-1$ 進展開を用いた場合よりもさらに二乗算の回数が半分になったことでさらに高速化が達成できている．Binary 法のみの場合にハミング重みが大きくなると計算時間が増えるが，NAF 法との組み合わせによりこれを解決できている．また，Adaptive Avanzi 法により乗算回数も削減されたことで従来法の中でもっとも高速な $t-1$ 進展開に NAF 法と Adaptive Window 法を用いた場合と比較し，約 30% の高速化に成功した．

5. ま と め

本稿ではペアリング計算後の拡大上の乗法群 $\mathbb{G}_T$ におけるべき乗算の高速化手法として埋め込み次数 $k=12$ の BN 曲線を用い，組織的に与えられる整数 χ で指数部を展開する方法を提案した．この手法にこれまでに提案されている Binary 法や NAF 法，Window 法，Avanzi 法といった拡大体上のべき乗算手法を組み合わせることで従来法よりも高速でかつ，ハミング重みに大きく依存することなく高速な実装が可能であることを示した．

6. 謝　　辞

本研究は総務省による受託研究「情報の来歴管理等の高度化・容易化に関する研究開発」の助成を受けて行われた．本研究にご協力いただいた関係各位に感謝する．

文　　献

[1] T. Nakanishi and N. Funabiki, "Verifier-Local Revocation Group Signature Schemes with Backward Unlinkability from Bilinear Maps," Asiacrypt2005, LNCS 3788, Springer-Verlag, pp.443-454, (2005).

[2] R. Sakai, K. Ohgishi and M. Kasahara, "Cryptosystems based on pairings," SCIS 2000, Okinawa, pp.26-28 (2000).

[3] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," CRYPTO2001 Proc., LNCS, Sprlner, vol. 2139, pp. 213-229 (2001).

[4] P. Barreto, B. Lynn and M. Scott, "On the selection of pairing-friendly groups," Selected Areas in Crypt. 2003, LNCS (3006), Springer, pp.17-25 (2004).

[5] P. Barreto, B. Lynn and M. Naehrig, "IEEE P1363.3 submission: pairing-friendly elliptic curves of prime order with embedding degree 12," IEEE Standards Assoc (2006).

[6] Y. Nogami, A. Saito and Y. Morikawa, "Finite Extension Field with Modulus of All-One Polynomial and Representation of Its Elements for Fast Arithmetic Operations" IEICE Trans., vol. E86-A, no. 9, pp.2376-2387 (2003).

[7] A. Miyaji, M. Nakabayashi and S. Takano, "New explicit conditions of Elliptic Curve Traces for FR-reduction," IEICE Trans., Fundamentals. vol.E84-A(5), pp.1234-1243 (2001).

[8] A Library for doing Number Theory, http://www.shoup.net/ntl/

[9] P. S. L. M. Barreto and M. Naehrig, "Pairing-Friendly Elliptic Curves of Prime Order," SAC2005, LNCS, Vol.3897,

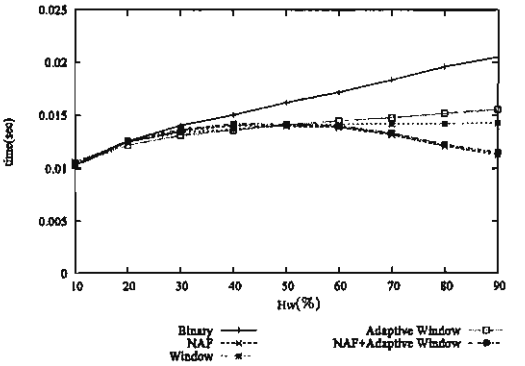

図 7　従来のべき乗算高速化手法の比較

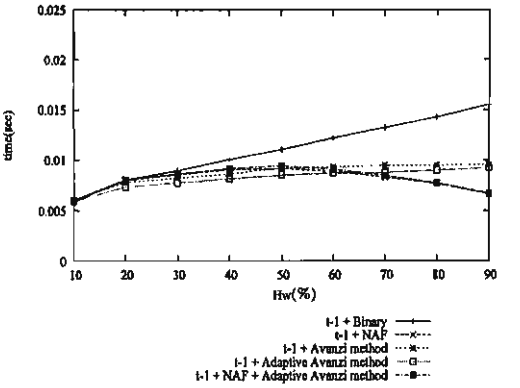

図 8　従来のべき乗算高速化手法+t-1 進展開

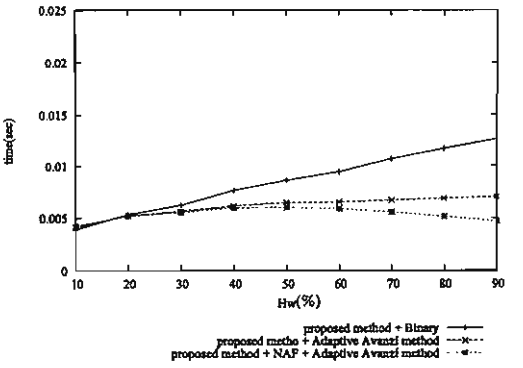

図 9　従来のべき乗算高速化手法+提案法

pp.319-331 (2006).

[10] M. Akane, H. Kato,T.Okimoto, Y. Nogami and Y. Morikawa, "An improvement of Millor's algorithm in Ate palring with Barreto-Naehrig curve, " Proc.CSS2007, pp.489-494 (2007).

[11] M. Akane, H. Kato, T. Okimoto,Y. Nogami and Y. Morikawa, "Efficient parameters for Ate pairing computation with Bareto-Naehrig curve, "Proc.CSS2007, pp.495-500 (2007).

[12] H. Cohen and G. Frey, "*Handbook of elliptic and hyperelliptic curve cryptography*", Chapman & Hall/CRC (2005).

[13] S. D. Galbraith and M. Scott "Exponentiation in pairing-friendly groups using homomorphisms", eprint 2008/117. To appear in Pairing 2008.

[14] R. P. Gallant, R. J. Lambert and S. A. Vanstone, "Faster Point Multiplication on Elliptic Curves with Efficient Endomorphisms," In J. Kilian (Ed.), CRYPTO 2001, Springer LNCS 2139 (2001), 190-200.

[15] R. Avanzi and P. Mihailescu, "Generic Efficient Arithmetic Algorithms for PAFFs (Processor Adequate Finite Fields) and Related Algebraic Structures," Proc. of SAC2003, LNCS 3006, Springer-Verlag, LNCS pp.320-334, 2003.26