マルチメディア通信と分散処理　106－35
コンピュータセキュリティ　16－35
（２００２．２．１５）

消費電力解析を考慮したスカラー倍演算について

中尾 浩† 藤尾 光彦‡

† 九州工業大学大学院 情報工学研究科
‡ 九州工業大学 情報工学部
†‡ 〒820－8502 福岡県飯塚市川津 680－4 Tel. 0948－29－7740
E-mail: †naka@flab.ces.kyutech.ac.jp , ‡fujio@ces.kyutech.ac.jp

概要

楕円曲線暗号においては楕円曲線上のスカラー倍演算が重要な役割を果たすが，演算時の消費電力の解析により処理が特定されると，暗号手法自体を危険にさらすことになる．本研究ではWindow法を使用する際の予備演算テーブルの演算量を削減することで，消費電力解析対策を施すことによる処理速度低下を抑えたWindow法を提案する．

The Scalar multiplication considerated for Power consumption analysis

Hiroshi Nakao† and Mitsuhiko Fujio‡

†Graduate School of Computer Science and Systems Engineering,
Kyusyu Institute of Technology
‡Faculty of Computer Science and Systems Engineering,
Kyusyu Institute of Technology
†‡Kawazu 680-4, Iizuka, Fukuoka 820-8502, Japan Tel. 0948－29－7740
E-mail: †naka@flab.ces.kyutech.ac.jp , ‡fujio@ces.kyutech.ac.jp

Abstract

The scalar multiplication on a elliptic curve, which plays a significant role in Elliptic Curve Criptosystem, may be discriminated by Power Consumption Analysis (PCA). In this article, we propose a new Window method which prevents PCA and suppresses the fall of processing speed by cutting down the amount of operations in the preliminary operation table of the Window method.

1 はじめに

楕円曲線を用いた暗号や署名を実装する際には，楕円曲線上の点のスカラー倍を高速に計算できることが，その暗号や署名の処理速度を高めることに直接つながる．しかし，現在楕円曲線暗号として利用されているもののひとつであるWeierstrauss型楕円曲線は，消費電力解析をはじめとするパワーアタックに対して弱く，そのままでは秘密情報を盗まれる危険性があることが指摘されている[1]．またパワーアタックへの対処を施したアルゴリズムを用いると，処理速度の低下がおこるという問題点がある．本研究では既存のWeierstrass型楕円曲線を用いて処理速度の低下を抑えつつ，電力解析に耐えうる暗号化の方法について述べる．特にWindow法を用いる際の予備演算によるテーブル作成時の演算量を減らすことにより高速化を行い，耐パワーアタックの処理を施した際の処理速度の低下を抑える方法を提案し，既存の方法との演算回数の比較結果について報告する．

2 背景

まず楕円曲線について説明する．以下で p は $p>3$ の素数とし，F_p を素体とする．楕円曲線とは

$$E: y^2 = x^3 + ax + b \qquad (1)$$

で表される曲線のことであり，ここで $a, b \in F_p$ は $4a^3 + 27b^2 \neq 0 \pmod p$ を満たす整数で無限遠点と呼ばれる特異点 $\mathcal{O}$ をも合わせ持つものとする (ここで，すべての算術演算は F_p 上で行なわれる)．

楕円曲線上の加法は次のように定義される．$P=(x_1,y_1), Q=(x_2,y_2)$ を E 上の点とする．もし $x_2=x_1$ かつ $y_2=-y_1$ ならば，$P+Q=\mathcal{O}$ とし，そうでないなら $P+Q=(x_3,y_3)$ とする．ここで

$$x_3 = \lambda^2 - x_1 - x_2 \quad , \quad y_3 = \lambda(x_1 - x_3) - y_1$$
$$P \neq Q \text{ のとき } \lambda = (y_2 - y_1)(x_2 - x_1)^{-1}$$
$$P = Q \text{ のとき } \lambda = (3x_1^2 + a)(2y_1)^{-1}$$

である．また，E 上のすべての点 P に対し以下のように定義する．

$$P + \mathcal{O} = \mathcal{O} + P = P$$

これらの演算は図 1 のように求められる．

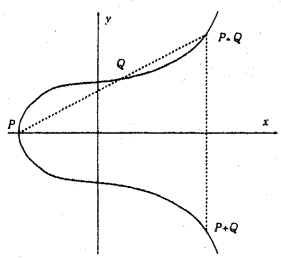

図 1: $P \neq Q$ の場合の楕円曲線の加算

2 点 P, Q を結ぶ直線を引き，曲線と交わるもう一点を $P*Q$ とする．その点から x 軸に関して対称な点を $P+Q$ とする (図 1)．また，すべての $P=(x,y) \in E$ について $-P$ は x 軸に対して対称な座標 $(x,-y)$ となる．

2.1 座標系

楕円曲線の表し方の代表的なものに，Affine 座標系と射影座標系を用いたものがある．Affine 座標系を使用した場合には楕円曲線上の加算および 2 倍算は定義体上の除算 (逆元演算) を必要とする．除算は乗算と比較すると処理時間がかかるので，今回は除算演算が不要な射影座標系を用いる．射影座標系には，(1) を $(x,y)=(X/Z, Y/Z)$ と変換する事によって得られる Projective 座標系や $(x,y)=(X/Z^2, Y/Z^3)$ の変換により得られる Jacobian 座標系などがある．加算に関しては Projective 座標系の方が高速であり，2 倍算に関しては Jacobian 座標系の方が高速である [2]．ここで Affine 座標系から Jacobian 座標系へ変形した楕円曲線が (2) となる．

$$E^J: Y^2 = X^3 + aXZ^4 + bZ^6 \qquad (2)$$

Jacobian 座標系において $P=(X_1,Y_1,Z_1)$, $Q=(X_2,Y_2,Z_2)$, $P+Q=R=(X_3,Y_3,Z_3)$ とすると，加算公式は以下のようになる．

- 加算公式 $(P \neq Q)$

$$X_3 = -H^3 - 2U_1H^2 + r^2$$
$$Y_3 = -S_1H^3 + r(U_1H^2 - X_3)$$
$$Z_3 = Z_1Z_2H$$

このとき各パラメータは

$$U_1 \leftarrow X_1{Z_2}^2 \ , \ U_2 \leftarrow X_2{Z_1}^2$$
$$S_1 \leftarrow Y_1{Z_2}^3 \ , \ S_2 \leftarrow Y_2{Z_1}^3$$
$$H \leftarrow U_2 - U_1 \ , \ r \leftarrow S_2 - S_1$$

また，2 倍算の公式は次のようになる．

- 2 倍算公式 $(R=2P)$

$$X_3 = -2s + m^2$$
$$Y_3 = -8{Y_1}^4 + m(s - X_3)$$
$$Z_3 = 2Y_1Z_1$$

このとき各パラメータは

$$s \leftarrow 4X_1{Y_1}^2, \ m \leftarrow 3{X_1}^2 + a{Z_1}^4$$

ここで Jacobian 座標系において，2 倍算の演算量を減らす為に (X,Y,Z,aZ^4) としたものを Modified Jacobian 座標系と呼ぶ．これら座標系による演算量の違いを示したものが表 1 である．ここで A は Affine，Pro は Projective，J は Jacobian，J^m は Modified Jacobian 座標系を表し，M は乗算，S は自乗，I は逆元演算を指す．

2.2 スカラー倍演算

$$Q = kP \qquad (3)$$

$$kP = \overbrace{P + P + \cdots + P}^{k \text{ 回}} \qquad (4)$$

表 1: 座標系の違いによる演算量比較

座標系	定義体上の演算	
	加算	2 倍
A	$2M+S+I$	$2M+2S+I$
Pro	$12M+2S$	$7M+5S$
J	$12M+4S$	$4M+6S$
J^m	$13M+6S$	$4M+4S$

P と Q は (1) 上の点で，$P(x_p, y_p)$, $Q(x_q, y_q)$ のように表されるとする．楕円曲線暗号ではスカラー値 k が秘密鍵として用いられる．(3) の kP は楕円曲線上の点 P を整数 k で k 倍するという通常の乗算ではなく，楕円曲線上の点で定義した加算を使って，楕円曲線上の点 P を整数 k で k 回足し合わせることを意味する．この加算を使うことで，元の点は (1) 上の別の点に移動する．この与えられた k, P を使用してスカラー倍点 kP を求める計算がスカラー倍演算である．主なスカラー倍演算法には 2 進法 (Binary Method), Window 法 (Window Method), NAF(non-adjacent form) を用いた符号付き 2 進法などがある．以下に 2 進法，NAF，符号付き 2 進法，Window 法のアルゴリズムを示す．

Algorithm 1: 2 進法

Input P, $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0,1\}$.
Output $Q = kP$.
1. $Q[0] \leftarrow P$.
2. for $i = \ell - 2$ to 0 do :
3. $\quad Q[0] \leftarrow 2Q[0]$.
4. $\quad$ if $k_i = 1$ then $Q[0] \leftarrow Q[0] + P$.
5. Return $Q[0]$.

2 進法 例えば $100P$ を求めるとき，$100P = 2(2(P + 2(2(2(P + 2P)))))$ と表す．これによりの加算 100 回の演算が加算 2 回と 2 倍算 6 回で計算できる．

Algorithm 2: NAF

Input $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0,1\}$.
Output $k' = \sum_{i=0}^{\ell} k'_i 2^i$, $k'_i \in \{-1,0,1\}$.
1. $c_0 \leftarrow 0$.
2. for $i = 0$ to ℓ do :
3. $\quad c_{i+1} \leftarrow \lfloor (k_i + k_{i+1} + c_i)/2 \rfloor$,
4. $\quad k'_i \leftarrow k_i + c_i - 2c_{i+1}$.
5. Return $(k'_\ell k'_{\ell-1} \cdots k'_0)$.

NAF 楕円曲線上の演算において加算と減算は演算のコストが同じであるため，符号付き 2 進法にする方が計算効率がよくなる．NAF はスカラー値 $k' = (k'_l k'_{l-1} \cdots k'_1 k'_0)_2$ のすべての $i \geq 0$ に対して $k'_i \times k'_{i+1} = 0$ となるような，k の符号付き 2 進表記である．また，非 0 bit が約 1/3 となる性質もある．

Algorithm 3: 符号付き 2 進法

Input P, $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0,1\}$.
Output $Q = kP$.
Precomputation.
1. *compute* $k' = \sum_{i=0}^{\ell} k'_i 2^i$, $k'_i \in \{-1,0,1\}$.
Main
2. $Q[0] \leftarrow P$.
3. for $i = \ell - 1$ to 0 do :
4. $\quad Q[0] \leftarrow 2Q[0]$.
5. $\quad$ if $k_i = 1$ then $Q[0] \leftarrow Q[0] + P$.
6. $\quad$ if $k_i = -1$ then $Q[0] \leftarrow Q[0] - P$.
7. Return $Q[0]$.

符号付き 2 進法 まず NAF を使用し符号付き 2 進表示にする．はじめに 2 倍算を行う点は 2 進法と同じである．次に k'_i $(i = 1, 2, \ldots, \ell)$ に注目する．k'_i の値が 1 の時は加算を実行し，k'_i の値が -1 の時は減算を実行する．k'_i の値が 0 のときは何もしない．

Algorithm 4: Window 法

Input P, $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0,1\}$.
Output $Q = kP$.
Precomputation.
1. $P_1 \leftarrow P$, $P_2 \leftarrow 2P$.
2. for $i = 1$ to $2^{w-1} - 1$ do: $P_{2i+1} = P_{2i-1} + P_2$.
3. $Q[0] \leftarrow P_{h_i}$ $\left(h_i = (k_{\ell-1} k_{\ell-2} \cdots k_{\ell-w})\right)$.
Main
4. for $i = (\ell - 1) - w$ to 0 by $-w$ do :
5. $\quad Q[0] \leftarrow 2^w Q[0]$.
6. $\quad h_i \leftarrow (k_i k_{i-1} \cdots k_{i-w+1})$.
7. $\quad$ if $h_i \neq 0$ then
8. $\quad\quad$ if P_{h_i} *is already computed* then *Read* P_{h_i}.
9. $\quad\quad$ else *compute* P_{h_i}.
10. $\quad Q[0] \leftarrow Q[0] + P_{h_i}$.
11. Return $Q[0]$.

Window 法 Window 法を使用するためには，まず予備演算のためのテーブルを作成する．1. と 2. がこれに当たる部分で 1 から $2^w - 1$ までを計算して

いる．次に Main loop 部分である．まず 5. で w 回の 2 倍算を行い，6. で $h_i \leftarrow (k_i k_{i-1} \cdots k_{i-w+1})$ を求める．$h_i \neq 0$ の時，すでに P_{h_i} が求められているなら，テーブルより読み込む．テーブルに存在しない場合には計算して求め，それをテーブルに格納する．10. で求めた P_{h_i} との加算を行う．$h_i = 0$ の場合は，何も処理をせずに 4. に戻る．

3 消費電力解析

電力解析は消費電力の変化に着目して鍵や処理内容を解析しようという方法であり，大きく分けて単純電力解析 (Simple Power Analysis: SPA) と電力差分解析 (Differential Power Analysis: DPA) の 2 つがある．

3.1 単純電力解析 (SPA)

電力の消費量を観測することで，視覚的に特性を見つける手法であり，アルゴリズム中に使用されている加算と乗算の違いなどを判別できる．つまりアルゴリズム中にデータに依存した処理が行われている場合は，SPA が適用される危険性がある．次に，2 進法 (Binary Method) を改良して単純電力解析 (SPA) 対策を施したアルゴリズムを示す [1].

Algorithm 5: 2 進法 (SPA 対策)

Input P, $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0, 1\}$.
Output $Q = kP$.
1. $Q[0] \leftarrow P$.
2. for $i = \ell - 2$ to 0 do :
3. $\quad Q[0] \leftarrow 2Q[0]$.
4. $\quad Q[1] \leftarrow Q[0] + P$.
5. $\quad Q[0] \leftarrow Q[k_i]$.
6. Return $Q[0]$.

アルゴリズムを見ればわかるように，スカラー値 k のすべての bit (ここでは k_i) に対して，楕円曲線上の 2 倍算と加算を行っているので，演算の手順は固定されており，bit の値による演算の違いは生まれない．しかしそれは，加算回数の増加を意味することとなる．

3.2 電力差分解析

電力差分解析とは消費電力量のデータを多く取り，統計処理を行うことで秘密情報を推定しようとする手法である．楕円曲線暗号系に対する電力差分解析への対処方法としては，乱数を有効に利用する三つの手法が提案されている [1].

3.2.1 秘密鍵のランダム化を用いた対処法

$\#E$ を楕円曲線上の点の数とする．$Q = kP$ の計算を次のように行うことで電力差分解析を防ぐ．

1. 20 ビット程度の乱数 r_1 を選ぶ．
2. $k' = k + r_1 \#E$ を計算．
3. $Q = k'P$ を計算．$\#EP = 0$ であるから，$Q = kP$ となる．

この手法を用いた場合，毎回異なる k' を用いて $k'P$ を計算することになるため電力差分解析の適用は不可能であり，攻撃を防ぐことができる．

3.2.2 P のブラインド化を用いた対処法

1. ランダムに楕円曲線上の点 R を選び，$S = kR$ を計算する．$Q' = k(R + P)$ も計算しておく．
2. スカラー倍演算は $Q = Q' - S$.
3. R と S はスマートカード内に収められており，新しい計算の度に $R \leftarrow (-1)^b 2R, S \leftarrow (-1)^b 2S$ を計算し R, S を更新 (b はランダム 1 ビット)．

攻撃者は $P' = P + R$ の値を知らないため電力差分解析を適用できない．つまり，秘密鍵 k を特定できない．

3.2.3 射影座標のランダム化を用いた対処法

$P = (x, y)$ の射影座標 (X, Y, Z) は次のように得る．

$$x = X/Z,\ y = Y/Z$$

ここで，有限体の元 λ ($\lambda \neq 0$) に対して，$(\lambda X, \lambda Y, \lambda Z)$ は $P = (X, Y, Z)$ の射影座標となっている．このことを用い，定義体上の演算を実行するたびにランダムに λ に対する P の射影座標を求め，その後の計算に利用する．また Jacobian 座標系を用いようとする場合は $(\lambda^2 X, \lambda^3 Y, \lambda Z)$ となる．この手法により P の射影座標の特定のビットは λ によってランダム化されているので，その演算の電力消費量との間の有意な相関は現れなくなり電力差分解析を防ぐことができる．

3.3 タイミング解析

タイミング解析は電力解析の手法ではないが，サイドチャネル攻撃 (実際に暗号が実装されている装置に対して攻撃を行う手法) の一つとして，DPA と同様に危険なものである．暗号化を行う際の演算時間が秘密鍵データに依存して異なる処理をする場合に，その差を統計的に解析して秘密情報（秘密鍵）を推定する方法である．

4 消費電力解析を考慮したスカラー倍演算

4.1 消費電力解析対策

SPA 対策 SPA 対策は Algorithm 5 を改良することで行った．対策として重要な点は bit 値に依存した振る舞いをさせない事で，その点を考慮したのが Algorithm 6 の 8.〜16. である．どの分岐点も，-1 の乗算と定義体上の加算を 1 回と同一処理を行っており，Algorithm 5 と同一の効果が得られる．

DPA 対策 スカラー倍演算を行う際は秘密鍵のランダム化を用いることで対処する．また，演算するたびに射影座標系によるランダム化を行うことで，更なる安全さを確保する．射影座標系としては，Jacobian 座標系と Modified Jacobian 座標系を用いる．

4.2 予備演算テーブル高速演算法

今回は Window 幅を $w = 4$ に設定し，そしてスカラー値 k は NAF を用いることとする．通常，Window 法で予備演算テーブルを作る場合は $w = 4$ の時は 1〜15 を調べる必要がある ($P_1, P_3, \ldots, P_{2^w-1}$ を先に調べるが，最終的には 1〜15 のテーブルが必要になる)．ここで NAF の隣り合う bit の乗算は 0 になるという特徴を利用すると，Window 幅は 4 なので NAF を使用した場合は 1 つの Window には $(1\bar{0}1\bar{0})_2$〜$(1010)_2$，つまり -10〜10 しか現れないことがわかる．楕円曲線上の点 $-kP$ の座標が kP の x 軸に関して対象な点であることをより，スカラー値 k は 1〜10 まで計算すればよい．通常の Window 法では予備演算テーブルが $\{P_1, P_2, \ldots P_{15}\}$ だけ必要なのに対し，提案法は $\{P_1, P_2, \ldots P_{10}\}$ と少なくてすむ．ここで，NAF と Jacobian 座標系 (J) と Modified Jacobian 座標系 (J^m) の 2 つの座標系を用いた予備演算テーブルの高速演算法を示す．

1. $P_1 \leftarrow P$ より P_1 を得る．
2. Modified Jacobian 座標系の 2 倍算により $2P, 4P, 8P$ を得る．ただし $2P, 4P$ は $J^m \leftarrow J^m(4M + 4S)$，$8P$ は $J \leftarrow J^m(3M + 4S)$ により求める．
3. $3, 5, 7, 9P$ を $J \leftarrow J^m + J^m(12M + 4S)$ で求める (Jacobian 座標系での加算と同演算量)．
4. $6P \leftarrow 2P_3, 10P \leftarrow 2P_5$ を $J \leftarrow 2J(4M + 6S)$ で求める．

4.3 提案アルゴリズム

Algorithm 6: 提案法

Input P, $k = \sum_{i=0}^{\ell-1} k_i 2^i$, $k_i \in \{0, 1\}$.
Output $Q = kP$.
Precomputation.
1. compute P_i.
2. compute $k' = \sum_{i=0}^{\ell} k'_i 2^i$, $k'_i \in \{-1, 0, 1\}$.
3. $Q[0] \leftarrow P_{h_i}$ $\left(h_i = (k'_\ell k'_{\ell-1} \cdots k'_{\ell-w+1})\right)$.

Main
4. for $i = \ell - w$ to 0 by $-w$ do :
5. $\quad Q[0] \leftarrow 2^w Q[0]$,
6. $\quad h_i \leftarrow (k'_i k'_{i-1} \cdots k'_{i-w+1})$,
7. $\quad Q[1] \leftarrow P_{h_i}$.
8. $\quad$ if $h_i > 0$ then
9. $\qquad\qquad Q[1]_y \leftarrow -1 \times P_{h_i}$,
10. $\qquad\qquad Q[0] \leftarrow Q[0] + P_{h_i}$.
11. $\quad$ else if $h_i < 0$ then
12. $\qquad\qquad Q[1]_y \leftarrow -1 \times Q[1]_y$,
13. $\qquad\qquad Q[0] \leftarrow Q[0] + Q[1]$.
14. $\quad$ else then
15. $\qquad\qquad Q[1]_y \leftarrow -1 \times Q[1]_y$,
16. $\qquad\qquad Q[1] \leftarrow Q[0] + P_1$.
17. Return $Q[0]$.

1. 4.2 より P_i ($i = 1, 2, \ldots, 10$) を求める．
2. NAF を用いて k を符号付き 2 進表記に変換．
3. P_{h_i} を $Q[0]$ に代入．
4. w 回，2 倍算を繰り返す．このとき $w-1$ 回までは $J^m \leftarrow 2J^m$ $(4M + 4S)$ の 2 倍算を用い，最後の 1 回は $J \leftarrow 2J^m$ $(3M + 4S)$ を用いる．
5. h_i を求める，7. P_{h_i} を $Q[1]$ に代入．
8. 8.〜16. はそれぞれの処理を行う．11. で h_i が負の時，$P = (X, Y, Z)$ とおくと $-P = (X, -Y, Z)$ なので，12. で Y 座標のみ変更させている．また，9. 15. 16. などがダミー演

表 2: 従来法と提案法の比較

演算手法	定義体上の演算		
	予備演算	加算	2 倍算
2 進法	-	$(12M+4S)\cdot\frac{\ell-1}{2}$	$(4M+6S)\cdot(\ell-1)$
符号付き 2 進法	-	$(12M+4S)\cdot\frac{\ell}{3}$	$(4M+6S)\cdot\ell$
Window 法	$112M+70S$	$(12M+4S)\cdot\frac{\ell-w}{w}$	$(4M+6S)\cdot(\ell-w)$
提案法	$68M+42S$	$(13M+6S)\cdot\frac{\ell+n-w}{w}$	$\left((4M+4S)\cdot(w-1)+(3M+4S)\right)\cdot\frac{\ell+n-w}{w}$

算となっており，実際の処理とは関係ないがこれにより攻撃者を欺くことができる．

17. 最後に $Q \leftarrow Q[0]$ により終了．

5 各手法との比較

5.1 予備演算テーブル演算量比較

まず，Window 法と提案法との予備演算テーブルの演算量比較を行う．Window 幅は 4 である．Window 法は P_1 から P_2 を求め，後は $P_{2i+1} = 2P + P_{2i-1}$ を行っていく．この時点で加算 7 回，2 倍算 1 回．そして残りの P_{2i} $(i = 2,3,\ldots w)$ も計算したとすると全部で加算 7 回，2 倍算 7 回となる．これらは Jacobian 座標系での演算なので $(12M+4S)\cdot 7+(4M+6S)\cdot 7 = 112M+70S(\simeq 168M)$ となる．一方，提案法は 4.2 節で示したとおりに，計算すると，$68M+42S(\simeq 101.6M)$ となる．

5.2 総演算量比較

秘密鍵 秘密鍵 $k = (k_{l-1}k_{\ell-2}\cdots k_1)$ の bit 数は ℓ だが，提案法の場合は 3.2.1 節の秘密鍵のランダム化を用いた対処法を適用するので，$\ell+n$ を bit 数とする．ここで n は 3.2.1 節の乱数 r_1 の bit 数である．また NAF を用いてるものは，$\ell+1$ bit となる．

座標変換 $A \to J$ の変換 $(X = xZ^2, Y = yZ^3)$ には $3M+S$，$J \to A$ の変換 $(x = \frac{X}{Z^2}, y = \frac{Y}{Z^3})$ には $3M+S+I$ かかる．提案法の場合は 3.2.3 節の射影座標のランダム化を用いた対処法を適用するので，乗算が 1 回ずつ多くなる．

5.2.1 総演算量比較

従来法と提案法の定義体上の演算量を表 2 に示す．この表 2 を用いて，Affine 座標系の P，Q において，k と P より $Q = kP$ のスカラー倍演算を求めるのに必要な演算量を表 3 に示す．このとき秘密鍵 k は 160bit $(\ell = 160)$．Window 幅 $w = 4$．秘密鍵のランダム化に用いる r_1 を 20bit $(n = 20)$ とする．$S/M = 0.8$，$I/M = 30$[3] と仮定した場合の乗算 M を基準とした演算量も示す．

表 3: スカラー倍演算量比較

演算手法	スカラー倍演算量	
2 進法	$(1590M+1272S)$ $+(6M+2S+I)$	$2645M$
符号付き 2 進法	$1280M+1173S$ $+(6M+2S+I)$	$2256M$
Window 法	$1204M+1162S$ $+(6M+2S+I)$	$2171M$
提案法	$1300M+1010S$ $+(8M+2S+I)$	$2148M$

6 まとめ

消費電力解析を考慮したスカラー倍演算について提案した．既存の方法との演算量の比較を行った結果，提案した方式は SPA や DPA に耐え，かつ，既存の方法よりも演算回数を削減できていることがわかった．予備演算テーブルに関して言えば，Window 法の演算量を約 40% 削減できていた．

参考文献

[1] J.S. Coron, "Resistance against Differential Power Analysis for Elliptic Curbe Cryptosystems", Proceedings of CHES '99, Springer-Verlag, 1999, pp.292-302.

[2] H. Cohen, A. Miyaji, and T. Ono, "Efficient elliptic curve exponentiation using mixed coordinates", *Advances in Cryptography* - ASIACRYPT '98, LNCS 1514, pp.51-65.

[3] C. H. Lim and H. S. Hwang, "Fast Implementation of Elliptic Arithmetic in GF(p^n)", *Public Key Cryptography* - PKC2000, LNCS 1571, pp.405-421.