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

2006－CSEC－34（14）
2006／7／20

表参照を用いた有限体乗算アルゴリズムの高速化

羽田間洋一[†]　中村　大輔[†]　廣友　雅徳[††]　森井　昌克[†††]

† 神戸大学大学院自然科学研究科　〒657-8501 神戸市灘区六甲台町 1-1
†† ひょうご情報教育機構　〒650-0044 神戸市中央区東川崎町 1-3-3
神戸ハーバーランドセンタービル 17 階
††† 神戸大学工学部　〒657-8501 神戸市灘区六甲台町 1-1
E-mail: †{030t240n,061t238n}@stu.kobe-u.ac.jp, ††mhirotomo@cmuj.jp, †††mmorii@kobe-u.ac.jp

あらまし　有限体 GF(2^m) 上の高速演算法の一つとして，表参照を用いた乗算アルゴリズムが M.A. Hasan によって与えられている．Hasan 法は GF(2^m) の元のビット列をいくつかのグループに分割し，表参照を用いてグループごとに演算することで高速な乗算が実行できる．Hasan 法においては表参照回数や表のサイズなどの性能が乗算の法に用いる既約多項式の形に依存するため，既約多項式の選び方によって乗算の高速化が図れる．本稿では，m 次 m 項既約多項式を利用した，表参照に基づく乗算アルゴリズムを提案する．提案する方法は m 次 m 項既約多項式を変換して得られる四項式を法として表参照に基づく乗算アルゴリズムを実行する．提案手法を用いることによって，Hasan 法では得られないような表参照回数や表のサイズを小さくする四項式を作ることができ，高速な GF(2^m) の乗算アルゴリズムを実現することができる．

キーワード　有限体，乗算アルゴリズム，表参照，m 次 m 項既約多項式

An Improved Mehtod for the Table Lookup Multiplication Algorithm over GF(2^m)

Yoichi HADAMA[†], Daisuke NAKAMURA[†], Masanori HIROTOMO[††], and Masakatu MORII[†††]

† Graduate School of Science and Technology, Kobe University
1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan
†† Hyogo Institute of Information Education Foundation
Kobe Harbor-land Center Building 17F
1-3-3 Higashikawasaki-cho, Chuo-ku, Kobe 650-0044
††† Faculty of Engineering, Kobe University
1-1, Rokkodai-cho, Nada-ku, Kobe, 657-8501 Japan
E-mail: †{030t240n,061t238n}@stu.kobe-u.ac.jp, ††mhirotomo@cmuj.jp, †††mmorii@kobe-u.ac.jp

Abstract　As effective arithmetic methods in the finite field GF(2^m), M.A. Hasan has presented a look-up table-based algorithm for GF(2^m) multiplication. In Hasan's method, the number of memory access and the table size are depend on the form of the irreducible polynomial which is used in the GF(2^m) multiplication. In this paper, we propose a lookup-up table-based algorithm which can be performed as an efficient GF(2^m) multiplication using the irreducible polynomial which has m degree and m terms. In this method, the algorithm is performed as the multiplication modulo quadnomial which is obtained by the irreducible polynomial which has m degree and m terms. Using this method, the number of memory access and the table size in the algorithm is smaller than that in Hasan's method, and GF(2^m) multiplication is faster than Hasan's method.

Key words　finite field, multiplication algorithm, look-up table, irreducible polynomial which has m degree and m terms

1. まえがき

現在用いられている暗号系には GF(2^m) 上の演算が用いられており，暗号系では安全性の観点から大きい m が用いられる．有限体上の乗算については，m が大きいほど乗算に時間を要する．そのため，有限体上の乗算の高速化が課題となっており，様々な研究がなされている．

有限体上の乗算の高速化を目的とした手法の 1 つに，表参照を用いた有限体乗算アルゴリズムがある．1992 年に Harper,Menezes,Vanstone らにより GF(2^m) の部分体 GF(2^8) における対数表および真数表を用いたアルゴリズムが提案された [1]．また，1997 年に Guajardo と Paar は対数表と真数表を用い，さらに部分体の乗算に Karatsuba-Ofman アルゴリズム [2] を用いた手法を提案した [3]．1998 年には Koc と Acar が 2 つの表と Montgomery の乗算アルゴリズムを用いた手法を提案した [4]．さらにこれらの手法よりも有限体上の乗算が高速に行われる手法として，2000 年 Hasan 法が提案された [5]．

Hasan 法は乗数をいくつかのグループに分割し，1 グループずつ乗算を行う．グループごとの乗算結果をあらかじめ作成された 2 つの表を参照することで乗算の高速化を図る．また，表参照を用いた手法においては，表を格納するメモリサイズと，表を参照する際のメモリアクセスが必要である．Hasan 法は，メモリサイズとメモリアクセスが，乗算の法として用いる既約多項式の 2 番目に大きい次数 d に依存する．

本稿では，m 次 m 項既約多項式を利用して，表参照に基づく乗算アルゴリズムの処理回数を削減する手法を提案する．提案手法では，m 次 m 項既約多項式から d が小さくなる四項式を生成し，その四項式を法として乗算アルゴリズムを実行することで処理回数の削減を図る．提案手法を Hasan 法との性能比較を行った結果，処理回数が削減でき，提案手法を用いることで，Hasan 法より効率的な乗算アルゴリズムが実現できることを示した．

2. 有限体上の乗算

2.1 有限体上の演算

有限体 GF(2^m) は 2^m 個の要素からなる集合である．GF(2^m) の要素は，GF(2) の要素 $(0,1)$ を係数とした $m-1$ 次の多項式で表すことができる．これを多項式表現という．例えば，GF(2^m) の要素 a を多項式表現した $A(x)$ は，

$$A(x) = a_{m-1}x^{m-1} + a_{m-2}x^{m-2} + \cdots + a_1x + a_0 \quad a_i \in \mathrm{GF}(2) \tag{1}$$

と表される．有限体の演算は加算と乗算が存在するが，GF(2^m) 上の演算は係数を並べてビット列と見立てて行う．加算は，同じ次数の係数を GF(2) での加算，すなわち XOR をとることで行う．一方，乗算では m 次の既約多項式 $F(x)$ を用いる．既約多項式 $F(x)$ を以下に示す．

$$F(x) = x^m + f_{m-1}x^{m-1} + \cdots + f_1x + 1 \quad f_i \in \mathrm{GF}(2) \tag{2}$$

$F(x)$ を用いた $A(x)$ と $B(x)$ の乗算結果 $P(x)$ を以下に示す．

$$P(x) = A(x)B(x) \bmod F(x) \tag{3}$$

法をとることで $P(x)$ が $m-1$ 次以下になる．有限体上の乗算は加算に比べ計算に時間を要し，m が大きいほど演算時間が長くなる．現在用いられている暗号系には有限体上の演算が用いられており，安全性の観点から大きい m が用いられている．したがって，有限体上の乗算の高速化が課題となっている．

2.2 Hasan 法

有限体 GF(2^m) 上の乗算の高速化を目的として，Hasan は表参照を用いた乗算アルゴリズムを与えた (Hasan 法)．Hasan 法では乗数 $B(x)$ の係数を g ビットずつの s 個のグループに分割し，グループごとに乗算を実行する．このグループごとの演算では，比較的時間を必要とする部分に対して，あらかじめ演算結果を表に格納しておき，演算時にその表を参照することで，演算の高速化を図っている．式 (3) における乗数 $B(x)$ を g ビットずつの s 個のグループに分割した式を次のように表す．

$$B(x) = B_{s-1}(x)x^{(s-1)g} + B_{s-2}(x)x^{(s-2)g} + \cdots + B_1(x)x^g + B_0(x) \tag{4}$$

ただし，$B_i(x) = \sum_{j=0}^{g-1} b_{ig+j}x^j \quad (i = 0, 1, \cdots, s-1)$

このグループに分割した $B_i(x)$ を用いて GF(2^m) 上の乗算を次のように行う．

$$\begin{aligned} P(x) &= A(x)B(x) \bmod F(x) \\ &= \Big(\big(\{A(x)B_{s-1}(x) \bmod F(x)\}x^g \bmod F(x) \\ &\quad + A(x)B_{s-2}(x) \bmod F(x) \big)x^g \bmod F(x) \\ &\quad + \cdots + \\ &\quad + A(x)B_1(x) \bmod F(x) \Big)x^g \bmod F(x) \\ &\quad + A(x)B_0(x) \bmod F(x) \end{aligned} \tag{5}$$

式 (5) の演算方法に基づくと，GF(2^m) 上の乗算は次のようなアルゴリズムになる．

アルゴリズム 1 : Group-Level の乗算アルゴリズム

入力 : $A(x), B(x), F(x)$
出力 : $A(x)B(x) \bmod F(x)$
Step 1.1 $P(x) := A(x)B_{s-1}(x) \bmod F(x)$
Step 1.2 for($i = s-2$ to 0){
$\quad \hat{P}(x) := x^g P(x) \bmod F(x)$
$\quad P(x) := \hat{P}(x) + A(x)B_i(x) \bmod F(x)$
}

Hasan 法ではアルゴリズム 1 のように GF(2^m) 上の乗算を行う．しかし，アルゴリズム 1 には $F(x)$ を法とする乗算が含まれているため，その部分の演算に時間を要する．Hasan 法では，その演算結果をあらかじめ表に格納しておき，アルゴリズム実行時にその表を参照することで，演算時間の短縮を図っている．次に，この表の設計について説明する．

アルゴリズム 1 は次の 2 つの演算で構成することができる.

$$P(x) = x^g P(x) \bmod F(x) \quad (6)$$

$$P(x) = A(x)B_i(x) \bmod F(x) \quad (7)$$

式 (6) はアルゴリズム 1 の Step1.2 の 1 行目の演算に，式 (7) は Step1.1 と Step1.2 の 2 行目の演算に利用できる．Hasan 法では式 (6)，(7) の演算に表参照を利用する．式 (6) の演算に利用する表を M-table，式 (7) の演算に利用する表を T-table と呼ぶ.

まず，M-table の設計について説明する．M-table は次の演算に利用される.

$$P(x) = x^g P(x) \bmod F(x) \quad (8)$$

式 (8) は $P(x)$ を g ビットシフトして $F(x)$ で法をとる演算である．この演算において，$F(x)$ で法をとる必要がある部分は $x^g P(x)$ の m 次以上である．よって，$F(x)$ の法に関係する部分は $P(x)$ の $m-g$ 次以上であり，$P(x)$ の $m-g-1$ 次以下は関係せず，g ビットシフトするだけで良い．このことから，$P(x)$ を次のように $m-g$ 次以上と $m-g-1$ 次以下に分けて考える.

$$\begin{aligned} P(x) &= \sum_{i=0}^{m-1} p_i x^i \\ &= \sum_{i=m-g}^{m-1} p_i x^i + \sum_{i=0}^{m-g-1} p_i x^i \\ &= P_2(x) + P_1(x) \end{aligned} \quad (9)$$

$P(x)$ のうち $P_2(x)$ のみが $F(x)$ の法に関係するため，Hasan 法では M-table に次の演算結果を格納する.

$$x^g P_2(x) \bmod F(x) \quad (10)$$

M-table の特徴としては，上式より M-table が $F(x)$ のみに依存し，$A(x)$ と $B_i(x)$ に依存しないことから，M-table はシステム起動時に作成し保持しておくことができる．また，$P_2(x)$ には 2^g 通りの多項式を取り得るので，M-table は 2^g 個の乗算結果を格納した表になる.

M-table のメモリサイズについて述べる．$F(x)$ を

$$F(x) = x^m + x^d + \sum_{i=1}^{d-1} f_i x^i + 1 \quad (11)$$

と表し，$F(x)$ の 2 番目に大きい次数を d とすると，M-table に格納されている式 (10) の乗算結果は $g+d$ ビットとなる．格納する乗算結果は 2^g 個存在するので，M-table のメモリサイズは $(g+d)2^g$ である．したがって，M-table を参照する際のメモリアクセスも $g+d$ に依存する.

次に，T-table の設計について説明する．T-table は式 (7) の演算に利用される．したがって，$B_i(x)$ を入力とした式 (7) の右辺の乗算結果を格納する．式 (7) の演算は，$A(x)$ と $F(x)$ に依存する．したがって，T-table は $A(x)$ 入力後に作成する.

T-table の特徴について述べる．$B_i(x)$ は g ビットなので，2^g 通りの多項式を取り得るが，アルゴリズム 1 より，T-table の参照回数は s 回である．T-table は $A(x)$ 入力後の作成となるため，効率面から 2^g と s の大小関係によって T-table の作成する要素が異なる．$2^g \geqq s$ のときはその乗算に用いる s 個の乗算結果だけを求めて表作成し，$2^g < s$ のときは $B_i(x)$ の取り得る 2^g 個すべての乗算結果を求めて表作成を行う.

表 1　アルゴリズム 2 の処理回数

$2^g \geqq s$ のとき	XOR	$(g+s+\frac{gs-5}{2})\lceil\frac{m}{w}\rceil$
	SHIFT	$(g-1)\lceil\frac{m}{w}\rceil+(s-1)\lceil\frac{m-g}{w}\rceil$
	Memory access	$(s-1)\lceil\frac{g+d}{w}\rceil+(\frac{sg}{2}+s)\lceil\frac{m}{w}\rceil$
	Table size	$(g+s)m+(g+d)2^g$
$2^g < s$ のとき	XOR	$(2^g+2s-\frac{g+7}{2})\lceil\frac{m}{w}\rceil$
	SHIFT	$(g-1)\lceil\frac{m}{w}\rceil+(s-1)\lceil\frac{m-g}{w}\rceil$
	Memory access	$(s-1)\lceil\frac{g+d}{w}\rceil+(2^g+s-2)\lceil\frac{m}{w}\rceil$
	Table size	$(m+g+d)2^g$

これまで述べた M-table，T-table を用いる Hasan 法について述べる．Hasan 法では，アルゴリズム 1 で用いられている式 (6)，(7) の演算結果を表参照することで乗算の高速化を図っている．式 (6) においては，表参照する部分としない部分があった．M-table を参照しない部分が式 (9) の $P_1(x)$，M-table を参照するのは式 (9) の $P_2(x)$ である．$P_1(x)$ をシフトした結果を τ_1，$P_2(x)$ に入力し M-table を参照した結果を τ_2 とすると，τ_1, τ_2 は次のように表される.

$$\tau_1 = x^g P_1(x) \quad (12)$$

$$\tau_2 = M[P_2(x)] \quad (13)$$

また，T-table を用いた式 (7) の参照結果を τ_3 とすると，τ_3 は，

$$\tau_3 = T[B_i(x)] \quad (14)$$

と表される．τ_1, τ_2, τ_3 を用いて表参照を用いた乗算アルゴリズムは次のように表される.

アルゴリズム 2：表参照を用いた乗算アルゴリズム

入力：$A(x), B(x), F(x)$,M-table

出力：$A(x)B(x) \bmod F(x)$

Step 2.1　T-table 作成

Step 2.2　$P(x) := T[B_{s-1}(x)]$

Step 2.3　for($i = s-2$ to 0){

　　$\tau_1 := x^g P_1(x)$

　　$\tau_2 := M[P_2(x)]$

　　$\tau_3 := T[B_i(x)]$

　　$P(x) := \tau_1 + \tau_2 + \tau_3$

　　}

アルゴリズム 2 における XOR，SHIFT の処理回数，表へのアクセス回数を表す Memory access および M-table と T-table のメモリサイズの合計を表す Table size を表 1 に示す．ただし，表中の w は CPU のビット数である.

2.3　Hasan 法の特徴

表 1 より，Memory access と Table size が $F(x)$ の 2 番目

表 2 d と $k+1$ の比較

m	d			$k+1$
	既約三項式	既約五項式	任意の既約多項式	
719	150	12	10	2
737	5	5	5	3
767	168	8	8	5
775	93	7	7	5
959	143	13	12	9

に大きい次数 d に依存することがわかる．このため，d の小さい法多項式で Hasan 法を用いれば，Memory access と Table size を削減することができる．また，一般に GF(2^m) 上の乗算においては，$F(x)$ には m 次既約三項式がよく用いられるが，d の小さい既約三項式が存在しない次数がある．以上のことから，d を小さくして Memory access と Table size の削減を図る手法を次章で提案する．

3. 提案手法

本章では，m 次 m 項既約多項式を利用して，表参照に基づく乗算アルゴリズムの処理回数を削減する手法を提案する．提案手法では，m 次 m 項既約多項式から d が小さくなる四項式を生成し，その四項式を法として乗算アルゴリズムを実行することで処理回数を削減している．

3.1 m 次 m 項既約多項式

本節では，提案手法で用いる m 次 m 項既約多項式について述べる．m 次 m 項既約多項式は x^m から x^0 の係数のうち 1 つだけが 0，他はすべて 1 の既約多項式である．k 次の係数だけが 0 になる m 次 m 項既約多項式 $F_m(x)$ は次式のようになる．

$$F_m(x) = x^m + x^{m-1} + x^{m-2} + \cdots + x^{k+1} + x^{k-1} + \cdots + x + 1 \quad (15)$$

$F_m(x)$ に $x+1$ をかけた $F'_m(x)$ は次のようになる．

$$F'_m(x) = F_m(x)(x+1) = x^{m+1} + x^{k+1} + x^k + 1 \quad (16)$$

$F'_m(x)$ は既約ではないが，$F'_m(x)$ を法とする乗算においても Hasan 法を用いることができる．$F'_m(x)$ の 2 番目に大きい次数は $k+1$ である．したがって，$k+1 < d$ となるような m 次 m 項既約多項式が存在する m では，$F'_m(x)$ を法として Hasan 法を用いることで処理回数を削減できる．$k+1 < d$ となるいくつかの例を表 2 に示す．ただし，表中の各項目の d および $k+1$ は，その次数における最小のものである．表 2 より，式 (16) のように m 次 m 項既約多項式 $F_m(x)$ から四項式 $F'_m(x)$ を作ることで，$F'_m(x)$ の 2 番目の次数 $k+1$ を既約多項式の 2 番目の次数 d より小さくできることがわかる．提案手法ではこの特徴を利用して，表参照を用いたアルゴリズムの処理回数を削減する．

3.2 提案アルゴリズム

前節では，m 次 m 項既約多項式 $F_m(x)$ から得られる四項式 $F'_m(x)$ の 2 番目の次数 $k+1$ が小さくなることを示した．$F'_m(x)$ を法としてアルゴリズム 2 を用いれば，乗算を高速に行うことができる．提案手法では，m 次 m 項既約多項式 $F_m(x)$ を有限体上の乗算の法として用い，アルゴリズム 3 を用いるための前処理として式 (16) により $F_m(x)$ から $F'_m(x)$ を生成する．$F'_m(x)$ を法としてアルゴリズム 2 を用いた後，後処理として $F_m(x)$ を法とした乗算結果を求める．まず，前処理と後処理について述べる．

前処理について述べる．前処理は，式 (16) を用いて m 次 m 項既約多項式 $F_m(x)$ から四項式 $F'_m(x)$ を生成する処理である．$F_m(x)$ はシステムの起動時に決まっているので，$F_m(x)$ から容易に求められる $F'_m(x)$ もシステムの起動時に決定できる．したがって，前処理はアルゴリズム 3 を用いるたびに行う必要がない．

次に後処理について述べる．後処理は，$F'_m(x)$ を法とした乗算結果 $\hat{P}(x)$ から $F_m(x)$ を法とした乗算結果 $P(x)$ を求める処理である．$\hat{P}(x)$ と $P(x)$ の間には以下の関係式が成り立つ．

$$\begin{aligned} P(x) &= A(x)B(x) \bmod F_m(x) \\ &= \{A(x)B(x) \bmod F'_m(x)\} \bmod F_m(x) \\ &= \hat{P}(x) \bmod F_m(x) \end{aligned} \quad (17)$$

式 (17) より，$\hat{P}(x)$ の次数は m 次以下である．$\hat{P}(x)$ の m 次の係数を p_m とおくと，$p_m = 0$ のときは $P(x) = \hat{P}(x)$ である．したがって，$p_m = 0$ のときは後処理は必要ない．$p_m = 1$ のときは，$P(x)$ と $F_m(x)$ の m 次の係数がともに 1 であるから，両者の XOR 演算により $P(x)$ を求めることができる．

提案アルゴリズムについて述べる．提案アルゴリズムでは $F_m(x)$ に前処理を施した $F'_m(x)$ を乗算の法に用いる．提案アルゴリズムを以下に示す．

アルゴリズム 3：提案アルゴリズム

入力：$A(x), B(x), F'_m(x)$,M-table
出力：$A(x)B(x) \bmod F(x)$
Step 3.1　T-table 作成
Step 3.2　$\hat{P}(x) = T[B_{s-1}(x)]$
Step 3.3　for($i = s-2$ to 0){
　　$\tau_1 := x^g P_1(\hat{x})$
　　$\tau_2 := M[P_2(\hat{x})]$
　　$\tau_3 := T[B_i(x)]$
　　$\hat{P}(x) := \tau_1 + \tau_2 + \tau_3$
　　}
Step 3.4　if $p_m = 1$　$P(x) := \hat{P}(x) + F_m(x)$

Step 3.1 から Step 3.3 はアルゴリズム 2 と同じ処理であり，Step 3.4 が後処理である．

アルゴリズム 3 の処理回数について述べる．Step 3.1 から Step 3.3 はアルゴリズム 2 と同じ処理であるが，アルゴリズム 3 では $F'_m(x)$ を法として用いているため，表 1 の m を $m+1$ に置き換える必要がある．また，$F'_m(x)$ の 2 番目に大きい次数は $k+1$ なので，d を $k+1$ に置き換える．

表 3　アルゴリズム 3 の処理回数

$2^g \geq s$ のとき	XOR	$(g+s+\frac{gs-4}{2})\lceil\frac{m+1}{w}\rceil$
	SHIFT	$(g-1)\lceil\frac{m+1}{w}\rceil+(s-1)\lceil\frac{m-g+1}{w}\rceil$
	Memory access	$(s-1)\lceil\frac{g+k+1}{w}\rceil+(\frac{sg}{2}+s)\lceil\frac{m+1}{w}\rceil$
	Table size	$(g+s)(m+1)+(g+k+1)2^g$
$2^g < s$ のとき	XOR	$(2^g+2s-\frac{g+6}{2})\lceil\frac{m+1}{w}\rceil$
	SHIFT	$(g-1)\lceil\frac{m+1}{w}\rceil+(s-1)\lceil\frac{m-g+1}{w}\rceil$
	Memory access	$(s-1)\lceil\frac{g+k+1}{w}\rceil+(2^g+s-2)\lceil\frac{m+1}{w}\rceil$
	Table size	$(m+g+k+2)2^g$

後処理である Step 3.4 は，$p_m = 1$ のときに限り XOR 演算を行う．m 次つまり $m+1$ 項の XOR 演算であるため，$\lceil\frac{m+1}{w}\rceil$ 回の XOR 演算を必要とするが，$p_m = 1$ となる確率を 1/2 として $\frac{1}{2}\lceil\frac{m+1}{w}\rceil$ 回の XOR 演算とする．この処理回数がアルゴリズム 2 の処理回数に加えられる．以上をまとめてアルゴリズム 3 の処理回数を表 3 に示す．表 3 と表 1 を比較すると，評価式ではほとんど変化ないものの，d にあたる $k+1$ を小さくすることで Memory access と Table size が削減できる．

4. 性能評価

本章では，提案手法の性能を評価する．乗数 $B(x)$ の区切るビット数 g について，表 3 に示した提案手法の処理回数を最も少なくする値を数値実験によって示す．次に，その処理回数を最小にする g の値を用いて提案手法と Hasan 法の処理回数を比較し，Hasan 法より高速な乗算アルゴリズムが実現できることを示す．

4.1 g の最適化

Hasan 法では，乗数 $B(x)$ を g ビットずつ s 個のグループに分けて乗算を行うため，s がループ回数のパラメータである．したがって，法多項式が同じであっても，$B(x)$ の次数によってループ回数が変わり処理回数も変化するので，$B(x)$ の次数を変えて議論する必要がある．

$B(x)$ のとりうる多項式について述べる．法多項式を m 次とするとき，$B(x)$ は $m-1$ 次以下の多項式である．一般に，GF(2^m) 上で t 次の多項式は，$t-1$ 次以下の t 項の係数がそれぞれ 0 と 1 の 2 通りあるので，2^t 個存在する．すなわち，法多項式を m 次としたときの $B(x)$ のとりうる多項式の個数は $\sum_{t=0}^{m-1} 2^t$ 個である．$B(x)$ が t 次のときの提案アルゴリズムの XOR の処理回数を $XOR(t)$ とおくと，$XOR(t)$ は表 3 の XOR の処理回数において $t = gs-1$ の関係にある関数である．すると法多項式を m 次としたときの XOR の平均処理回数 XOR_{ave} は，$XOR(t)$ を用いて次の式で求められる．

$$XOR_{ave} = \frac{\sum_{t=0}^{m-1}(2^t \cdot XOR(t))}{\sum_{t=0}^{m-1} 2^t} \qquad (18)$$

同様にして SHIFT，Memory access，Table size の平均処理回数 $SHIFT_{ave}, Memory_{ave}, Table_{ave}$ を求める．

今回は m = 108, 200, 302, 400, 484, 586, 720, 814, 960 の場合を表 3 の評価式に代入し，処理回数を求める．それぞれの m においては，$B(x)$ の次数を変化させ，$XOR_{ave}, SHIFT_{ave}, Memory_{ave}, Table_{ave}$ を求めた．その結果を図 1 から図 4 に示す．4 つの図より，$m \geq 200$ では $g = 4$ で最も効率的に乗算が行われ，$200 \leq m \leq 586$ では $g = 5$ が，$m \geq 586$ では $g = 6$ で最も効率的に乗算が行われるという結果が得られた．

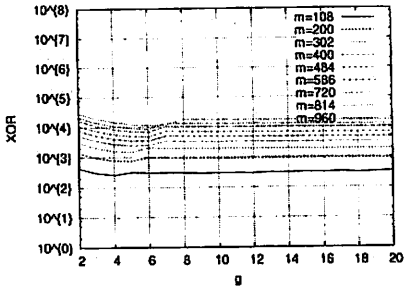

図 1　XOR の変化

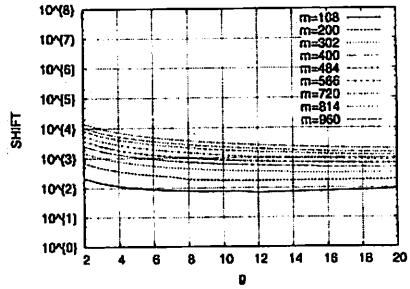

図 2　SHIFT の変化

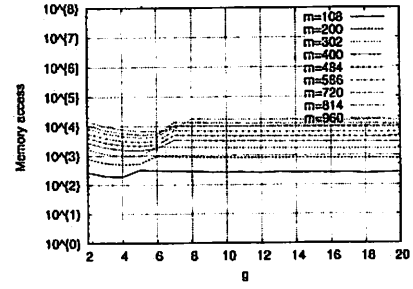

図 3　Memory access の変化

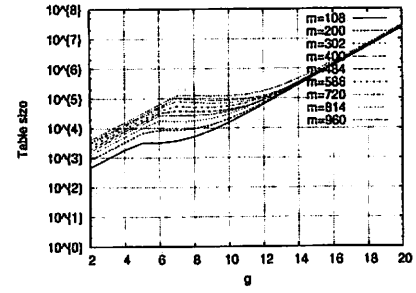

図 4　Table size の変化

4.2 処理回数の比較

本節では，Hasan 法と提案手法の処理回数の比較を行う．一般に GF(2^m) 上の乗算においては，法多項式 $F(x)$ には m 次既約三項式がよく用いられ，m 次既約三項式が存在しない次数

においては，m 次既約五項式が用いられる．そこで本稿では，Hasan 法の処理回数を (i) 既約三項式を用いた場合，(ii) 既約三項式または既約五項式を用いた場合，(iii) 任意の既約多項式を用いた場合の 3 通りに場合分けし，提案手法と Hasan 法の処理回数を比較する．なお今回は，$m = 1$ から 1000 までの範囲で比較を行った．

(i) 既約三項式を用いた Hasan 法との比較

提案手法の処理回数と既約三項式を用いた Hasan 法の処理回数の比較を表 4 に示す．表 4 では，$m = 800$ から 1000 の範囲で m 次 m 項既約多項式 $F_m(x)$ から生成される四項式 $F'_m(x)$ の 2 番目に大きい次数 $k+1$ が d より小さくなったものを示している．表 4 の結果を含め，$k+1$ が d より小さくなるのは $m \leq 1000$ において 99 個存在した．表 4 より提案手法を用いることで，Memory access と Table size を削減し，高速な乗算アルゴリズムが実現できることがわかる．XOR と SHIFT についてはわずかに増加しているが，これは XOR と SHIFT が $k+1$ に依存せず，$F'_m(x)$ の次数が m 次既約多項式に比べて 1 次大きくなることが原因である．

(ii) 既約三項式または既約五項式を用いた Hasan 法との比較

既約三項式または既約五項式で用いた Hasan 法と，提案手法の処理回数を比較した．ここで，Hasan 法の処理回数は既約三項式または既約五項式のうち 2 番目の次数が最も小さい既約多項式を用いて乗算アルゴリズムを実行した場合の処理回数で比較している．$m \leq 1000$ において $d < k+1$ となるものを表 5 に示す．表 5 中の i は，既約多項式において m と d 以外に係数が 1 の次数である．ただし，$i = 0$ はすべての既約多項式の係数が 1 なので省略している．表 5 の $m = 13$ を例にとると $d = 4, i = 3, 1$ なので，この既約多項式は $x^{13}+x^4+x^3+x+1$ である．表 5 のように，41 個の m で $d < k+1$ となる結果が得られた．表 5 からわかるように，提案手法を用いることによって $m = 23, 47, 107, 149, 479, 599, 665$ で d と $k+1$ の差が大きくなり，Memory size を同等にして Table size を削減できた．

(iii) 任意の既約多項式を用いた Hasan 法との比較

任意の既約多項式で用いた Hasan 法と，提案手法の処理回数を比較した．ここで，Hasan 法の処理回数は，任意の既約多項式のうち 2 番目の次数が最も小さい既約多項式を用いて乗算アルゴリズムを実行した場合の処理回数で比較している．$m \leq 1000$ において $d < k+1$ となるものを表 6 に示す．表 6 のように，36 個の m で $d < k+1$ となる結果が得られた．表 6 からわかるように，提案手法を用いることによって $m = 23, 47, 107, 479, 665$ で d と $k+1$ の差が大きくなり，Memory size を同等にして Table size を削減できた．

5. むすび

本稿では，m 次 m 項既約多項式を利用して，表参照に基づく乗算アルゴリズムの処理回数を削減する手法を提案した．提案手法は，乗算の法多項式に m 次 m 項既約多項式を取り，その多項式に $x+1$ を乗じて得られる四項式を法として Hasan 法を用いることで処理回数の削減を図った．提案手法により，Memory size と Table size の 1 つのパラメータである d を小さくすることができた．GF(2^m) 上の乗算においてよく用いられる既約三項式を法多項式とした場合と比較すると，Memory size と Table size ともにかなりの削減効果が見られた．また，五項以上の既約多項式を比較対象とした場合においても，Memory size を同等にして Table size を削減できることを示し，提案手法の有効性を示した．

文　献

[1] G. Harper, A. Menezes and S. Vanstone, "Public-key cryptosystem with very small key length," Proc. Advanced in Cryptology—EUROCRYPT'92, pp.163–173, 1992.

[2] A. Karatsuba and Y. Ofman, "Multiplication of multidigit numbers on Automata," Sov. Phys. Dokl., vol.7, no.7, pp.595–596, 1963.

[3] J. Guajardo and C. Paar, "Efficient algorithms for elliptic curve cryptosystems," Proc.Advanced in Cryptology—CRYPT'97, pp.342–356, 1997.

[4] C. Koc and T. Acar, "Montgomery multiplication in GF(2^m)," Design, Codes and Cryptography, vol.14, no.1, pp.57–69, 1998.

[5] M. A. Hasan, "Look-up table-based large field multiplication in memory constrained cryptosystems," IEEE Trans. Comput., vol.49, no.7, pp.749–758, July 2002.

表 4　提案手法の処理回数と既約三項式を用いた Hasan 法の処理回数の比較

	Hasan 法					提案手法				
m	d	XOR	SHIFT	Memory access	Table size	$k+1$	XOR	SHIFT	Memory access	Table size
801	217	8463	3449	6019	65536	89	8469	3452	5491	57472
815	333	8577	3638	6631	73856	32	8578	3639	5417	54720
841	144	9125	3888	6148	63424	121	9153	3901	6025	62080
847	136	9179	3915	6180	63296	11	9207	3928	5635	55424
849	253	9920	3935	6768	70912	53	9227	3939	5786	58240
871	378	9743	4172	7523	80320	47	9772	4186	6099	59264
873	352	9786	4193	7554	78784	89	9793	4196	6255	62080
881	78	9852	4226	6288	61760	14	9854	4227	5997	57792
887	147	9908	4254	6613	66560	20	9910	4255	6026	58560
889	127	9911	4256	6614	65408	34	9940	4270	6189	59584
911	204	10494	4522	7261	71744	135	10496	4523	6960	67456
921	221	10599	4575	7479	73472	42	10607	4578	6566	62144
927	403	10657	4604	8284	85504	36	10665	4607	6597	62144
935	417	11096	4798	8707	86912	238	11098	4799	7779	75584
951	260	11265	4882	8042	77888	39	11272	4886	6942	63872
959	143	11336	4918	7342	70912	9	11338	4919	6818	62464
983	230	11962	5206	8308	78016	47	11964	5207	7331	66432
991	39	12027	5239	7364	66304	38	12059	5254	7381	66368

表 5　提案手法の処理回数と既約三項式または既約五項式を用いた Hasan 法の処理回数の比較

	Hasan 法						提案手法				
m	d	i	XOR	SHIFT	Memory access	Table size	$k+1$	XOR	SHIFT	Memory access	Table size
13	4	3,1	62	6	65	1472	2	63	6	66	1472
23	5		66	8	69	2176	2	66	8	69	2112
37	6	4,1	140	15	141	3136	2	142	15	142	3008
47	5		148	24	147	3712	2	148	24	147	3648
49	6	5,4	148	24	147	3904	5	150	25	148	3904
81	4		256	53	240	5824	3	257	54	240	5888
85	8	2,1	258	54	241	6336	2	261	55	243	6080
87	7	5,1	262	56	244	6400	5	263	57	244	6400
95	7	5,1	270	60	249	6912	5	270	60	249	6912
97	6	4,2	360	65	327	6976	5	364	66	329	7040
103	9	4,1	368	84	332	7552	8	372	86	334	7616
107	9	7,4	375	88	337	7808	3	376	88	337	7552
149	10	9,7	539	145	459	10560	7	540	145	459	10496
157	6	5,2	550	150	465	10816	3	555	152	468	10816
159	8	7,4	557	154	469	11072	6	559	154	470	11072
183	8	7,4	717	208	586	12608	2	718	209	587	12352
195	8	3,2	864	225	695	13376	3	866	226	696	13184
199	9	4,1	868	259	697	13696	4	875	262	702	13504
245	6	4,1	1119	359	863	16448	4	1119	360	864	16448
301	9	5,2	1580	540	1169	20224	7	1590	545	1174	20224
341	10	8,6	1891	670	1364	22848	4	1891	671	1365	22592
365	9	6,5	2158	779	1535	24320	2	2159	780	1536	24000
383	9	5,1	2230	815	1574	25472	2	2231	816	1575	25152
399	9	9,2	2489	920	1739	26496	6	2493	921	1741	26432
415	9	4,2	2548	949	1771	27520	2	2561	955	1778	27200
421	5	4,2	2772	967	1917	27648	2	2786	973	1924	27584
463	11	8,7	3180	1215	2161	30720	2	3195	1222	2169	30272
465	8	3,2	3202	1226	2173	30656	5	3206	1228	2175	30592
479	9	7,4	3268	1259	2208	31616	4	3269	1260	2208	31424
483	9	6,4	3512	1276	2364	31872	3	3516	1278	2366	31616
565	11	6,1	4427	1764	2901	37248	2	4446	1773	2910	36800
585	8	3,2	4816	1933	3132	38336	5	4821	1936	3134	38272
599	9	8,6	4900	1975	3176	39296	6	4901	1975	3176	39232
665	11	6,4	5877	2414	3742	43648	3	5879	2414	3742	43264
719	12	9,8	6851	2851	4303	47168	2	6853	2851	4304	46656
737	5		7293	2925	4560	47872	3	7294	2925	4561	47872
767	8	7,4	7533	3166	4685	49984	5	7534	3167	4686	49920
775	7	6,4	7899	3325	4903	50432	5	7925	3337	4916	50432
813	10	3,1	8567	3633	5276	53056	2	8573	3637	5280	52672
847	17	13,3	9179	3915	5621	55680	11	9207	3928	5635	55424
959	13	10,8	11336	4918	6817	62592	9	11338	4919	6818	62464

表 6　提案手法の処理回数と任意の既約多項式を用いた Hasan 法の処理回数の比較

	Hasan 法						提案手法				
m	d	i	XOR	SHIFT	Memory access	Table size	$k+1$	XOR	SHIFT	Memory access	Table size
13	4	3,1	62	6	65	1472	2	63	6	66	1472
23	5		66	8	69	2176	2	66	8	69	2112
37	6	5,4,3,2,1	140	15	141	3136	2	142	15	142	3008
47	5		148	24	147	3712	2	148	24	147	3648
49	6	5,4	148	24	147	3904	5	150	25	148	3904
81	4		256	53	240	5824	3	257	54	240	5888
85	8	2,1	258	54	241	6336	2	261	55	243	6080
87	7	5,1	262	56	244	6400	5	263	57	244	6400
95	6	5,4,2,1	270	60	249	6848	5	270	60	249	6912
97	6		360	65	327	6976	5	364	66	329	7040
107	7	5,3,2,1	375	88	337	7680	3	376	88	337	7552
149	9	8,5,2,1	539	145	459	10496	7	540	145	459	10496
157	6	5,2	550	150	465	10816	3	555	152	468	10816
183	8	7,4	717	208	586	12608	2	718	209	587	12352
195	7	5,4,2,1	864	225	695	13312	3	866	226	696	13184
199	7	6,5,3,2	868	259	697	13568	4	875	262	702	13504
245	6	4,1	1119	359	863	16448	4	1119	360	864	16448
301	8	6,5,2,1	1580	540	1169	20160	7	1590	545	1174	20224
341	8	4,3,2,1	1891	670	1364	22720	4	1891	671	1365	22592
365	9	6,5	2158	779	1535	24320	2	2159	780	1536	24000
383	9	5,1	2230	815	1574	25472	2	2231	816	1575	25152
399	9	6,4,2,1	2489	920	1739	26496	6	2493	921	1741	26432
415	9	4,2	2548	949	1771	27520	2	2561	955	1778	27200
421	5	4,2	2772	967	1917	27648	2	2786	973	1924	27584
463	10	8,7,4,2	3180	1215	2161	30656	2	3195	1222	2169	30272
465	8	3,2	3202	1226	2173	30656	5	3206	1228	2175	30592
479	8	7,6,5,4,3,2	3268	1259	2208	31552	4	3269	1260	2208	31424
483	7	6,4,3,1	3512	1276	2364	31744	3	3516	1278	2366	31616
565	10	5,4,3,2	4427	1764	2901	37184	2	4446	1773	2910	36800
585	8	3,2	4816	1933	3132	38336	5	4821	1936	3134	38272
665	10	9,7,6,5,3,1	5877	2414	3742	43584	3	5879	2414	3742	43264
719	10	8,5,4,3,2,1	6851	2851	4303	47040	2	6853	2851	4304	46656
737	5		7293	2925	4560	47872	3	7294	2925	4561	47872
767	8	7,4	7533	3166	4685	49984	5	7534	3167	4686	49920
775	7	6,4	7899	3325	4903	50432	5	7925	3337	4916	50432
813	8	7,5,2,1	8567	3633	5276	52928	2	8573	3637	5280	52672
959	12	7,5,4,1	11336	4918	6817	62528	9	11338	4919	6818	62464