NOVEMBER 1987

LIDS-P-1716

A NEW ALGORITHM FOR MULTIPLICATION IN FINITE FIELDS

by

Antonio Pincin

ABSTRACT

This note presents a new algorithm for computing the product of two elements in a finite field $\bar{F}$ by means of sums and products in a fixed subfield F and $\bar{F}$ (ex. $\bar{F} = GF(2^m)$ and $F = GF(2)$). The algorithm is based on a normal basis representation of fields and assumes that the dimension m of $\bar{F}$ over F is a highly composite number. A very fast parallel implementation and a considerable reduction in the number of computations is allowed, in comparison with some methods discussed in the literature.

I. INTRODUCTION

In recent years there has been a considerable interest in VLSI architectures and algorithms for computing multiplications in finite fields [17], [20], [21]. Finite field computations are widely used, e.g. in error correcting codes [11], digital signal processing [10], pseudo-random numbers generation [4], [6], [9] and cryptographic protocols [2], [3], [5], [16].

The purpose of this note is to present a new algorithm for evaluating the product of two elements in a finite field $\bar{F}$ by means of sums and products in a subfield F of $\bar{F}$.

Multiplications in $\bar{F}$ are represented in terms of bilinear forms in F, referring to a normal basis representation of fields. This technique, which underlays the remarkable algorithm proposed by Massey and Omura [9], [17], is naturally associated with a matrix theoretic treatment of all the matter.

The basic step of our algorithm exploits some properties of the bilinear forms representing the product with respect to a noraml basis representation. The computational savings introduced in the basic step are then exploited and magnified if the dimension m of the field $\bar{F}$ over the subfield F is a highly composite number.

The algorithm allows a very fast implementation with concurrent use of many processing elements.

In the remaining part of this introduction basic algebraic facts are recalled. An explicit bilinear representation of the multiplication problem is given in Section II and in Section III the new algorithm is presented. Section IV and VI deal with some computational aspects of the algorithm and associated properties. In Section V two examples illustrate computational gainings and speed together with a detailed description of the method in a specific case.

In the sequal $\bar{F}$ is a finite field, F a subfield of $\bar{F}$, "m" the dimension of $\bar{F}$ as a vector space over F, $B_m = \{v_o, v_1, \ldots, v_{m-1}\}$ a generic basis of $\bar{F}$ over F in which $v_o, v_1, \ldots, v_{m-1} \in \bar{F}$ are the linearly independent vectors of the basis. Once a basis B_m for $\bar{F}$ over F has been given, any β in $\bar{F}$ is represented by a row vector with m elements in F:

$$\beta = (b_o, b_1, \ldots, b_{m-1})$$

Assume that p and p^t are the characteristic and the cardinality of F, respectively (p a prime number). An F-automorphism of $\bar{F}$ is an automorphism of $\bar{F}$ which leaves every element of F fixed [8], The set of the F-automorphisms of $\bar{F}$ is a group (the "Galois group" of $\bar{F}$ over F) consisting of m distinct elements G_o, $G_1, \ldots, G_{m-1}$

$$G_i : \bar{F} \to \bar{F} \quad : \quad \alpha \to \alpha^{p^{ti}} = \alpha G_i, \quad \alpha \in \bar{F},$$

$$G_i = G_1^i \; , \quad G_1^m = G_1^o = G_o = I$$

(I the identity automorphism).

A basis $\{v_o, v_1, \ldots, v_{m-1}\}$ is "normal" (for $\bar{F}$ over F) if $v_i = \alpha\, G_i$ for some α in $\bar{F}$ (a normal basis generated by α). Such a basis will be denoted as

$$\{\alpha, \alpha^{p^t}, \ldots, \alpha^{p^{t(m-1)}}\}.$$

It can be shown that a normal basis always exists [8]. The following theorem constitutes the keystone of the algorithm presented in the next section [11], [8]:

Theorem 1. Let $\bar{F}$ contain p^n elements. Then $\bar{F}$ contains a subfield F of p^t elements iff t divides n.

Let F_1, $F_2, \ldots, F_{s+1}$ be finite fields and assume that F_{i+1} is a subfield of F_i, $i = 1, \ldots, s$, m_{s-i+1} the dimension of F_i over F_{i+1}. Then F_1, $F_2, \ldots, F_{s+1}$ (in the order) constitute a "descending chain of fields". We summarize these facts by the following notation

$$F_1 \underset{m_s}{>} F_2 \underset{m_{s-1}}{>}, \ldots, \underset{m_2}{>} F_s \underset{m_1}{>} F_{s+1} \tag{1}$$

As a corollary of the previous theorem we have that if $n = m_s m_{s-1} \cdots m_1$, $m_i > 1$, positive integers, then there exists a descending chain of fields

$$\bar{F} = F_1 \underset{m_s}{>} F_2, \ldots, \underset{m_1}{>} F_{s+1} = F.$$

The same is true if $m = m_s m_{s+1} \ldots m_1$ is the dimension of $\bar{F}$ over F.

II. THE MASSEY-OMURA ALGORITHM

Let $\bar{F}$ be represented as a row vector space over F, each row consisting on the coordinates of an element of $\bar{F}$ with respect to a given basis B_m of $\bar{F}$ over F. Let

$\gamma = (c_o, c_1, \ldots, c_{m-1}) \in \bar{F}$

$\beta = (b_o, b_1, \ldots, b_{m-1}) \in \bar{F}$

$\pi = \gamma\beta = (d_o, d_1, \ldots, d_{m-1}) \in \bar{F}$

Therefore the problem of obtaining the product of γ and β is transformed into the problem of computing the components d_i of its representative vector and reduces to the evaluation of m symmetrical bilinear forms over F. In fact, let $a_{h,k}^{(i)} \in F$ denote the projection of $v_h . v_k$ on the vector v_i (i.e. the i-th component of the element $v_h v_k$ represented on the basis B_m) and introduce the following matrices

$$A^{(i)} = ||a_{h,k}^{(i)}||_{h,k\ =\ 0,\ldots,m-1} \qquad i = 0,1,\ldots,m-1$$

Then, for any β and γ in $\bar{F}$, we have

$$d_i = \gamma\ A^{(i)}\beta' \qquad i = 0,1,\ldots,m-1 \tag{2}$$

In the case when $B_m = N_{m,\alpha}$ is a normal basis the symmetrical matrices $A^{(i)}$ are connected each other in a very simple way.

Dropping the superscript m-1 both in $A^{(m-1)}$ and in $a_{h,k}^{(m-1)}$,

$$A = A^{(m-1)} = ||a_{hk}||_{h,k=-0,\ldots,m-1},$$

we have

$$A_{m-i-1} = S^i A S'^{\,i} = ||a_{((h+i)),((k+i))}||_{h,k=0,\ldots,m-1}$$

$$d_{m-i-1} = \sum_{h,k=0,\ldots,m-1} a_{((k+i)),((h+i))} b_k c_h \tag{3}$$

$$i = 0,1,\ldots,m-1$$

where

$$S = \begin{bmatrix} 0 & 1 & 0 & \cdots\cdots & 0 \\ 0 & 0 & 1 & 0 \cdots & 0 \\ \vdots & & & & \vdots \\ \vdots & & & & 0 \\ 0 & & & & 1 \\ 1 & 0 & & & 0 \end{bmatrix}$$

and $((j))$ means j mod m.

Note that S induces a single step cyclic right shift into the components of a row vector.

Equations (2) and (3) provide a compact representation of the Massey-Omura multiplier. The structure of the A matrix, defining the multiplication in $\bar{F}$, must satisfy some restrictions. It can be shown [12] [18] that the sum of the elements in a row (column) of the symmetrical matrix A is zero, with the exception of the m-th row (column). In complete generality we assume that this sum is one (normal bases generated by an element with unitary trace).

III. A NEW ALGORITHM

We are now in a position to introduce the basic step of the multiplication algorithm.

Let $\underline{a}_k$ be the k+1-th row of A. Then

$$a_{m-i-1} = \beta\ S^i A S'^i \gamma' = \sum_{k=0,m-1} b_{((k-i))}(\underline{a}_k s'^i \gamma') \tag{4}$$

As a consequence of the rows structure of A, we have

$$\sum_{k=0,m-2} \underline{a}_k + \underline{a}_{m-1} = (0,0,\ldots,0,1) \quad \text{and}$$

$$\sum_{i=1,m-1} \underline{a}_k S'^i + \underline{a}_k = \begin{cases} (0,\ldots,0), & k = 0,1,\ldots,m-2 \\ (1,1,\ldots,1), & k = m-1 \end{cases} \tag{5}$$

$$(0,\ldots 0,1)S'^i\gamma' = c_{m-1-i}$$

Therefore the m-i-1 component of the product can be expressed as

$$d_{m-i-1} = \sum_{k=0,m-2} (b_{((k-i))} - b_{((m-1-i))})\underline{a}_k S'^i\gamma' + b_{((m-1-i))}c_{m-1-i} \tag{6}$$

In order to compute $\underline{a}_k S'^i$, we resort again to the rows structure of A so

(for $k = 0,1,\ldots,m-2$)

$$\underline{a}_k S'^{i\prime} = \sum_{j=0,m-1} a_{k,((j+i))} c_j = \sum_{j=0,m-2} a_{k,((j+i))} (c_j - c_{m-1})$$

and, finally,

$$d_{m-i-1} = \sum_{k=0,m-2} \Big(\Big(\sum_{j=0,m-2} a_{k,((j+i))} (c_j - c_{m-1})\Big)(b_{((k-i))} - b_{((m-1-i))})\Big) +$$

$$+ b_{((m-1-i))} c_{m-1-i} \tag{7}$$

Note that the evaluation of d_{m-1} by means of formula (6) invovles the computation of $\underline{a}_k\gamma'$ $k = 0,1,\ldots,m-2$. In this step no multiplications are needed. In fact, using equations (5), we have

$$\underline{a}_k\gamma' = - \sum_{i=1,m-1} \underline{a}_k S'^{i}\gamma' \qquad k = 0,1,\ldots,m-2 \tag{8}$$

Once the computation of $\underline{a}_k S'^{i}\gamma'$ $k = 0,1,\ldots,m-2$ has been performed, only sums are involved in (8).

This implies that $(m-1)(m-1) = (m-1)^3$ products in F are required for computing the terms $\underline{a}_k S'^{i}\gamma'$ and m^2 products are successively needed for evaluating the coefficients d_{m-i-1} $i=0,1,\ldots,m-1$. Therefore $P_{(m)} = (m-1)^3 + m^2$ products in F are sufficient to compute a generic product $\beta\gamma$ in $\bar{F}$.

The previous procedure implies also that a number $S_{(m)} = (m-1)(m^2-1)$ of sums in F is sufficient.

The computational procedure above will be called the "basic algorithm".

Suppose now m is not a prime interger and let $m = m_2m_1$, m_1 and m_2 greater than 1, be a not trivial factorization of m. By theorem 1, there exists a descending chain of fields $\bar{F} > F_2 > F$, F_2 an intermediate field between $\bar{F}$ and F, with $m_2 = \dim_F \bar{F}$, $m_1 = \dim_F F_2$.

Since a normal basis of a finite field over any subfield always exists, it is possible to split the computation of $\beta\gamma$ in $\bar{F}$ in two steps. In the first step, the basic algorithms between $\bar{F}$ and F_2 is applied, in the second step products in F_2, previously obtained in step one, are computed applying the basic algorithm between F_2 and F.

This procedure is an alternative to the direct application of the basic algorithm between $\bar{F}$ and F. It is easily seen it has a recurrent character.

In fact the first descent along the chain (from $\bar{F}$ to F_2) splits a single multiplication problem in $\bar{F}$, whose solution depends on m-th order bilinear forms over F, into several multiplication problems in F_2, whose solution depends on m_1-th order bilinear forms over F.

The procedure above extends in a natural way to any descending chain of fields between $\bar{F}$ and F and is called a "factorization of the algorithm (along the chain)".

If m is highly composite, the factorization of the algorithm allows a considerable saving in the number of products and sums in F needed for computing the product $\beta\gamma$. This will be shown in the next section.

4. COMPUTATIONAL ASPECTS

1. Consider the factorization of the algorithm along the descending chain $\bar{F} \underset{m_2}{>} F_2 \underset{m_1}{>} F$.

The basic algorithm between $\bar{F}$ and F_2, $\bar{F} \underset{m_2}{>} F_2$, requires $P_{(m_2)}$ products in F_2. In turn, applying the basic algorithm between F_2 and F, $F_2 \underset{m_1}{>} F$, each product in F_2 requires $P_{(m_1)}$ multiplications in F. Therefore, in order to compute the product $\beta\gamma$

$$P_{(m_2,m_1)} = P_{(m_2)}P_{(m_1)} = ((m_2-1)^3 + m_2^2)((m_1-1)^3 + m_1^2)$$

multiplications in F are sufficient. Simple calculations show that

$$P_{(m_2,m_1)} < P_{(m)}, \quad m = m_1m_2 \qquad m_1,m_2 > 1$$

proving that the factorization of the algorithm reduces the maximum number of multiplications in F.

In m_1,m_2 aren't prime integers, it is possible to resort to a finer factorization of the algorithm. Let

$$m = m_s m_{s-1},\ldots,m_1 m_o, \quad m_i>1 \qquad i=1,\ldots,s \qquad m_o=1 \tag{9}$$

be a factorization of the integer m and

$$\bar{F} = F_1 \underset{m_s}{>} F_2 \underset{m_{s-1}}{>} \cdots > F_s \underset{m_1}{>} F_{s+1} = F \tag{10}$$

a descending chain of fields associated with it. Using the basic algorithm between F_i and F_{i+1}, $i = 1,2,\ldots,s$ in the order (factorization of the algorithm), the number of F-multiplications needed for computing $\beta\cdot\gamma$ in $\bar{F}$ is given by

$$P_{(m_s,m_{s-1},\ldots,m_1)} = \prod_{i=1,s} ((m_i-1)^3 + m_i^2) \tag{11}$$

2. (10) is an upper bound on the number of F-multiplications, when using the factorization of the algorithm along the descending chain (10). There exist cases in which the upper bound (11) is reached (see example 1 in the next section).

3. Suppose now the algorithm is factorized along the chain (10) and let $S_i = S_{(m\ ,m_{i-1},\ldots,m\ )}$ be the number of sums in F that are sufficient for applying the factorized algorithm along the descending chain

$$F_{s-i+1} \underset{m_i}{>} F_{s-i+2} \underset{m_{i-1}}{>} \cdots > F_s \underset{m_1}{>} F_{s+1} = F \tag{12}$$

Then

$$S_i = K_i S_{i-1}\ u_i \qquad i = s,s-1,\ldots,1 \tag{13}$$

where

$$u_i = h_i S_{(m_i)} , \qquad k_i = P_{(m_i)}, \qquad h_i = m_{i-1} \dots m_1 m_o,$$

$$m_o = 1 \qquad , \qquad S_o = 0$$

In fact, in order to apply the basic algorithm between F_{s-i+1} and F_{s-i+2}, $F_{s-i+1} \underset{m_i}{>} F_{s-i+2}$, $S_{(m_i)}$ sums in F_{s-i+2} are sufficient and these sums are computed in F with $h_i S_{(m_i)}$ sums (h_i is the dimension of F_{s-i+2} over F). Moreover the basic algorithm between F_{s-i+1} and F_{s-i+2} induces the computation of $P_{(m_i)}$ product in F_{s-i+2}. Such computation in turn requires no more than S_{i-1} sums in F, we have to take into account in the evaluation of S_i.

4. Let σ_j be any permutation of the numbers 1,2,...,s. For every such permutation, there exists a descending chain of fields

$$F_1 \underset{m_{(s)\sigma_j}}{>} F_2 \underset{m_{(s-1)\sigma_j}}{>} \dots F_s \underset{m_{(1)\sigma_j}}{>} F$$

Then there are at most s! descending chains of fields, that are different from one another in the ordering of the factors $m_1, m_2, \dots, m_s$. The factorization of the algorithm along these chains does not change the upper bound (11) of the number of F-multiplications. On the contrary, changing these chains affects the number of sums in F

$S_i(=S_{(m_{(s)\sigma_j}, m_{(s-1)\sigma_j}, \ldots, m_{(s)\sigma_j})})$. For instance, in the case $s = 2$,

$s_{(t,q)} < S_{(q,t)}$ if and only if $t<q$ [12]. In general a factorization of the algorithm along the descending (10) chain is optimal (i.e. it minimizes the maximum number S_s of sums in F) if the factors of the chain (10) satisfy the condition $m_s \leq m_{s-1} \leq \ldots \leq m_1$.

5. The factorization of the algorithm makes it possible to reduce the number of coefficients in F necessary to assign the A matrices of the bilinear forms, defining the multiplication algorithm at each step (see example 1). In fact, the application of the algorithm (factored or not) requires an a priori knowledge of the coefficients of the bilinear forms (2), (3). They are the elements of the symmetrical matrices A, used in each step of the algorithm (one $m_i x m_i$ matrix for every application of the basic algorithm between F_{s-i+1} and F_{s-i+2}). In the single step case, the A matrix is completely defined by $(m+1)m/2 - 2$ coefficients in F. This depends on the symmetry of A and on the constraints on the first row and the last column of A.

If $m=m_1m_2$, $m_1,m_2>1$, and the algorithm is factorized along the chain

$F \underset{m_1}{>} F_2 \underset{m_2}{>} F$, $m_2(m_1(m_1+1)/2-2) + m_2(m_2+1)/2- 2$ coefficients in F are

sufficient, less than in the single step. The same conclusion holds in the general case.

V. EXAMPLES

Example 1: We compare two algorithms, referring to the maximum number of sums and products needed for computing m bilinear forms over F.

The reference algorithm, Λ_1, is the factorized algorithm described in this note. The maximum number of sums and products for Λ_1 is given by (13) and (11).

The second algorithm, Λ_2, computes the bilinear form $\beta A\gamma'$ by evaluating first the vector $\theta' = A\gamma'$ (m^2 products and m(m-1) sums are sufficient) and then $\beta\theta'$ (m products and m-1 sums). Since the number of forms to be computed is m, this algorithm needs

$$m^3 + m^2 \quad \text{products in F}$$

$$m(m^2-1) \quad \text{sums in F}$$

Assume $m=2^s$ and suppose that the factors of the descending chain (10) are $m_i = 2$, $i = 1,2,\ldots,s$. We have $P_{(2)} = 5,\ldots,P_{(2,2,\ldots,2)} = 5^s \cong 2^{2.32\ s}$

The number of F-multiplications is reduced from an order m^3 (in Λ_2) to an order $m^{2.32}$ (in Λ_1), which is a remarkable reduction if m is large. Notice that in this situation the factorized algorithm is in its most efficient form (in particular, since $m_i = 2$, there are no residual symmetries to exploit in the 2x2 A matrices and only one coefficient is used in a single step of the factorized algorithm). If $2 = \dim_{F\ +1} F_i$ in general it is impossible to compute a product in F_i with less than five multiplications in F_{i+1} (notice that the fundamental results of [19] cannot be straightforwardly applied to this problem).

The Λ_1 columns of the table list the number of sums and products needed by the factorized algorithm while the Λ_2/Λ_1 columns provide the ratios between the maximum number of sums and products in Λ_2 and Λ_1 respectively.

The remarkable computational advantage of the factorization may be immediately appreciated.

s	m	Λ_1 products	Λ_1 sums	Λ_2/Λ_1 products	Λ_2/Λ_1 sums
2	4	25	35	3.2	1.7
3	8	125	195	4.6	2.6
4	16	625	1.015	7	4
5	32	3.125	5.155	10.8	6.4
6	64	15.625	25.935	17	10
8	256	390.625	650.615	43	26
10	1024	9.765.625	16.274.335	110	66
12	4096	224.140.625	406.894.215	281	169
16	65536	----	----	1829	1107

Notice that the number of coefficients in F necessary to define the algorithm is $m(m+1)/2$ in Λ_2 and $1+2+\ldots+2^{s-1} = 2^s-1 = m-1$ in Λ_1.

Example 2: We give here a detailed description of the factorized algorithm presented in section III. In addition some properties of the representation (2), (3) are pointed out. Let $\bar{F} = GF(2^6)$ $F_2 = GF(2^3)$ $F = GF(2)$, and consider the factorization of the algorithm along the chain

$GF(2^6) \supset GF(2^3) \supset GF(2)$. A root σ of the polynomial $g(x) = x^3 + x^2 + 1$ generates a normal basis $N_{3,\sigma} = \{\sigma, \sigma^2, \sigma^4\}$ for $GF(2^3)$ over $GF(2)$ and a root α of $p(x) = x^2+x+1$ generates a normal basis $N_{2,\alpha} = \{\alpha, \alpha^2\}$ for $GF(2^2)$ over $GF(2)$ and also for $GF(2^6)$ over $GF(2^3)$ (this is a particular case of a more general one, [12], [13]).

To apply the algorithm the matrix A of the bilinear representation (3) has to be found. Let A_3 (A_2) be that matrix when the product is between elements of $GF(2^3)$ (of $GF(2^6)$) represented over $GF(2)$ (over $GF(2^3)$) on the normal basis $N_{3,\sigma}$ $(N_{2,\sigma})$.

Simple computations give the representations of the elements σ^2, $(\sigma^2)^2$, $(\sigma^4)^2$, $\sigma\sigma^2$, $\sigma\sigma^4$ in the normal basis $N_{3,\sigma}$:

$$\sigma^2 = \sigma^2 \qquad (\sigma^2)^2 = \sigma^4 \qquad (\sigma^4)^2 = \sigma$$

$$\sigma\sigma^2 = \sigma+\sigma^4 \qquad \sigma\sigma^4 = \sigma^2+\sigma^4 \qquad \sigma^2\sigma^4 = \sigma+\sigma^2$$

The symmetric matrix A_3 is therefore the following:

$$A_3 = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

Similarly it is found that $A_2 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$.

Denote with $\gamma = (c_0, c_1)$ $\beta = (b_0, b_1)$ $\rho = (d_0, d_1)$ elements of $GF(2^6)$ represented over $GF(2^3)$ in the basis $N_{2,\alpha}$ and $\gamma' = (c_0', c_1', c_2')$ $\beta' = (b_0', b_1', b_2')$ $\rho' = (d_0', d_1', d_2')$ elements of $GF(2^3)$ represented over $GF(2)$ in the basis $N_{3,\sigma}$. If $\rho = \gamma\beta$ the application of steps (6), (7) between $GF(2^6)$ and $GF(2^3)$ gives

$$d_0 = (b_0 \oplus b_1) * (c_o \oplus c_1) \oplus b_0 * c_0$$
$$d_1 = (b_0 \oplus b_1) * (c_0 \oplus c_1) \oplus b_1 * c_1 \qquad (14)$$

(* and $\oplus$ denote multiplication and addition in $GF(2^3)$). If $\rho' = \gamma'\beta'$ applying steps (6), (7) between $GF(2^3)$ and $GF(2)$ the following is obtained

$$d_0' = (b_1'+b_0')(c_0'+c_2') + (b_2'+b_0')(c_1'+c_2') + b_0'c_0'$$
$$d_1' = (b_2'+b_1')(c_0'+c_1') + (b_0'+b_1')(c_0'+c_2') + b_1'c_1' \qquad (15)$$
$$d_2' = (b_0'+b_2')(c_1'+c_1') + (b_1'+b_2')(c_0'+b_1') + b_2'c_2'$$

where the operation are in the binary field $GF(2)$. The algorithm compute d_0, d_1 by evaluating every product * in (14) by using (15) after the computation of the sums $b_0 \oplus b_1$ and $c_0 \oplus c_1$. If $d_i = (d_{i0}, d_{i1}, d_{i2})$ $c_i = (c_{i0}, c_{i1}, c_{i2})$, $b_i = (b_{i0}, b_{i1}, b_{i2})$ are the components of d_i, c_i, b_i in the basis $N_{3,\sigma}$ the explicit expression of d_{00} is

$$d_{00} = ((b_{01}+b_{11}) + (b_{00}+b_{10}))((c_{00}+c_{10}) + (c_{02}+c_{12})) +$$
$$+ ((b_{02}+b_{12}) + (b_{00}+b_{10}))((c_{01}+c_{11}) + (c_{02}+c_{12})) + b_{00}+b_{10})(c_{00}+c_{10}) +$$
$$+ (b_{01}+b_{00})(c_{00}+c_{02}) + (b_{02}+b_{00})(c_{01}+c_{02}) + b_{00}c_{00}$$

The "nested" basis (see also remark 4 of section VI) associated to the algorithm is

$$\sigma\alpha,\ \sigma^2\alpha,\ \sigma^4\alpha,\ \sigma\alpha^2,\ \sigma^2\alpha^2,\ \sigma^4\alpha^2 \tag{16}$$

which is still a normal basis $N_{6,\sigma\alpha}$ for $GF(2^6)$ over $GF(2)$, after a permutation of the basis vectors. In the basis $N_{6,\sigma\alpha}$ the matrix A of the Massey-Omura multiplier has 15 non zero elements, so the Massey-Omura algorithm compute a product in $GF(2^6)$ represented in $N_{6,\sigma\alpha}$ with 90 multiplications and 84 additions in GF(2). According to section III the factorized algorithm computes (14) and (15) with 36 multiplications and 90 additions. Other simmetries of (14) and (15) can be exploited: the first terms of the sums in (14) and also the first of d_0' and the second of d_1', the first of d_2' and the second of d_0', the first of d_1' and the second of d_2' are pairwise equal, therefore only 18 multiplications and 48 additions in GF(2) are needed. Without taking into account the time for input/output operations a completely parallel realization of the Massey-Omura multiplier (with elementary processors capable of the binary GF(2) operations between two operators) multiplies in five clock pulses, the factorized algorithm in six (but with a greater communication complexity).

VI. SOME REMARKS

1. An important part of the algorithmic principle presented in section II is the factorization along the chain of fields (10). This principle can be applied to algorithms different from the one considered in Section III. For example the algorithm presented in [20] is intrinsically sequential and

factoring it along the chain (10) gives a much more parallel procedure.

2. Consider the basic algorithm between F_{s-i+1} and F_{s-i+2} in (12). The coefficients $a_{k,((j+i))}$ in (7) are fixed element of F_{s-i+2} so they induce a linear transformation into F_{s-i+2} which can be computed in no more than $(m_{i-1}\ldots m_1)^2$ operations in F. With this modification in the case $m=2^s$ (example 1) the factorized algorithm requires no more than $m^2(1+s/4)$ multiplications in F.

3. Algorithms based on the bilinear representation of section II allow a highly parallel implementation which computes a product in a finite field $GF(2^n)$ in time $\log_2 n$. This is true also for the factorized algorithm of section III with the modification of remark 2. It is worthwhile to notice that multiplication algorithms derived from efficient multiplication and division algorithms for polynomials (as the FFT and the Schonhage-Strassen algorithms [1], [10], [14], [15]) do not allow a parallel implementation running in time linear in $\log_2 n$.

4. The basis of $\bar{F}$ over F resulting from the algorithm factorization exhibits a "nested structure" as in [16]. In fact, let N_m = $(v_{o,i}, v_{1,i}, \ldots, v_{m-1,i})$ be the normal basis for F_{s-i+1} over F_{s-i+2} $i = s, s-1, \ldots, 1$. The basis for F_{s-i+1} over F, associated with the factorization of the algorithm, consists of $n_i = m_i m_{i-1}, \ldots, m_1$ elements of F_{s-i+1}, given by

the products

$v_{j_i,i} v_{j_{i-1},i-1}, \ldots, v_{j_1,1}$, $0 \leq j_k \leq m_k - 1$ $1 \leq k \leq i$. It is worthwhile to notice that, in general, it is not a normal basis for F_{s-i+1} over F [13].

5. The matrix $A^{(5)}$ of the bilinear representation (2) associated to the basis (16) in example 2 has the following block structure:

$$A^{(5)} = \begin{bmatrix} A_3 & A_3 \\ A_3 & 0 \end{bmatrix}$$

If the basis has the nested structure of remark 4 it can be seen that the matrices of the bilinear representation (2) present a block structure. In the above case $A^{(5)}$ can be described as the "tensor product" [7] of the matrices A_2 and A_3 of example 2 (this is a particularization of the more general case).

VII. CONCLUSIONS

A new algorithm for multiplication in finite field $\bar{F}$ has been presented. The algorithm is based on the hypothesis that the dimension of the field $\bar{F}$ over the subfield F is a highly composite number and exploits the existence of intermediate fields E between $\bar{F}$ and F. The algorithm allows highly parallel fast computations of products in a finite field with a substantially smaller number of computational elements than in some other methods. The underlying algorithmic principle of exploiting intermediate fields can be extended to other algorithms in order to achieve better performances in term of speed and/or required operations.

REFERENCES

[1] A.V. Aho, J.E. Hopcroft, J.D. Ullman, "The design and Analysis of Computer Algorithm", Reading, MA, Addision-Wesley, 1974.

[2] T. Beth, N. Cot, I. Ingemarson, ed. "Advances in Cryptology: Proceedings of Eurocrypt '84" Lecture Notes in Computer Science n.209, Berlin, Springer Verleg, 1985.

[3] G.R. Blakey, D. Chaum, ed, "Advances in Cryptology: Proceeding of Crypto '84", Lecture Notes in Computer Science n. 196, Berlin, Springer Verlag, 1985.

[4] M. Blum, S. Micali, "How to generate cryptographically strong sequences of pseudo-random bits", SIAM J. Comput. vol. 13, n.4, November 1984.

[5] W. Diffie, M.E. Hellman: "New directions in Cryptography" IEEE Trans. Inf. Theory, Vol. IT22, n.6, p.644-654, November 1976.

[6] S.W. Golomb, "Shift Register Sequences", Holden-Day, San Franciso, 1967.

[7] N. Jacobson, "Lectures in Abstract Algebra", Vol. 2, Princeton (N.J.), D. Van Nostrand, 1959.

[8] N. Jacobson, "Lecture in Abstract Algebra", Vol. 3, Princeton (N.J.), D. Van Nostrand, 1959.

[9] J.L. Massey and J.K. Omura: "Computational method and apparatus for finite field arithmetic", U.S. Patent application, submitted 1981.

[10] J.H. McClellan, C.M. Rader, "Number Theory in Digital Signal Processing", Englewood Cliff, N.J., Prentice Hall, 1979.

[11] F.J. McWilliams, N.J.A. Sloane, "The Theory of Error Correcting Codes", New York; North Holland, 1977.

[12] A. Pincin, "Optimal multiplication algorithms in finite fields", Thesis, Institute of Electrical Eng., Universita degli Studi di Padova, Padova (Italy), July 1986.

[13] A. Pincin, "Bases for finite fields and a canonical decomposition for a normal basis generator", MIT LIDS-P-1713. submitted to "Communications in Algebra", June 1987.

[14] J.M. Pollard, "The fast Fourier Transform in a finite field", Math. Comp. vol. 25, p.365-374, 1971.

[15] A. Schonhage, "Fast multiplication of polynomials over fields of characteristic 2", Acta Informatica vol. 7, p. 395-398, 1977.

[16] P.K.S. Wah, M.Z. Wang, "Realization and application of the Massey-Omura lock", pp. 175-182 in Proc. Intern. Zurich Seminar, March 6-8, 1984.

[17] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutch, J.K. Omura, I.S. Reed: "VLSI architectures for computing multiplications and inverses in $GF(2^m)$", IEEE Transactions on Computers, vol. C-34, no.8, pp. 709-716, August 1985.

[18] C.C. Wang, "Exponentiation in finite field $GF(2^m)$", Ph.D. Disseration, School Eng. Appl. Sci., Univ. Calif., Los Angeles, June 1985.

[19] S.W. Winograd, "On multiplication in algebraic extension fields", Theoretical Computer Science vol. 8, p.359-377, 1979.

[20] C.S. Yeh, I.S. Reed, T.K. Truong, "Systolic multipliers for finite fields $GF(2^m)$", IEEE Trans. Comput., vol. C-33, pp. 357-360.

[21] K. Yiu, K. Peterson, "A single-chip VLSI implementation of the discrete exponentiation public key distribution system", Proc. GLOBCOM 82, IEEE 1982, pp.173-179.