Efficient Multiplication Beyond Optimal Normal Bases

Arash Reyhani-Masoleh, *Member*, *IEEE*, and M. Anwar Hasan, *Senior Member*, *IEEE*

Abstract—In cryptographic applications, the use of normal bases to represent elements of the finite field $\mathrm{GF}(2^m)$ is quite advantageous, especially for hardware implementation. In this article, we consider an important field operation, namely, multiplication which is used in many cryptographic functions. We present a class of algorithms for normal basis multiplication in $\mathrm{GF}(2^m)$. Our proposed multiplication algorithm for composite finite fields requires a significantly lower number of bit level operations and, hence, can reduce the space complexity of cryptographic systems.

Index Terms—Finite fields, multiplication, normal bases, composite fields, optimal bases.

1 Introduction

Many cryptographic functions, such as key exchange, signing, and verification, require significant amount of computations in the finite field $\mathrm{GF}(2^m)$. The elements of such a field can be represented in different ways. The choice of the representation plays an important role in determining the complexity of a finite field arithmetic unit and, consequently, that of a cryptographic system. Among the various ways one can represent field elements, the use of normal bases has drawn significant attention, especially for implementing cryptographic functions in hardware [1].

In a normal basis representation, squaring can be performed simply by a cycle shift of the coordinates of an element and, hence, in hardware, it is almost free of cost. Such a cost advantage often makes the normal basis a preferred choice of representation. However, a normal basis multiplication is not so simple. In [10], Massey and Omura proposed a normal basis multiplication scheme which can be implemented in bit-parallel fashion using m identical logic blocks whose inputs are cyclically shifted from one another [25]. Although this normal basis multiplier offers modularity, its space complexity[1] is quite high.

In the recent past, considerable efforts have been made, for example, [13], [23], [6], [9], and [20], to reduce the space complexity of the normal basis multiplier. In [13], two special types of normal bases were reported which are known as type-I and type-II optimal normal bases. In [5], it was shown that these two types are all the optimal normal bases in $\mathrm{GF}(2^m)$. The use of these optimal normal bases can considerably reduce the complexity of the multiplier [23], [6], [3], and [20].

In this article, we first present an algorithm for multiplication in $\mathrm{GF}(2^m)$. This algorithm is quite generic in the sense that it is not restricted to any special type of normal bases. Compared to other generic algorithms for normal basis multiplication in $\mathrm{GF}(2^m)$, the proposed one requires fewer bit level multiplications. Although this is achieved at the expense of extra bit level additions, the total number of GF(2) operations is the same as that of the best known generic algorithm. Unlike the existing normal basis multiplication algorithms, our algorithm is highly suitable for software implementation on general purpose processors and we give the number of main instructions needed by such processors for multiplication over $\mathrm{GF}(2^m)$.

Our algorithm is then applied to the type-I optimal normal basis to further reduce the number of bit level operations. We then present an algorithm for normal basis multiplication in composite finite fields. This algorithm significantly reduces bit level operations, in terms of both addition and multiplication over GF(2). To show the advantage of the proposed algorithms, we compare our results with those of the best known normal basis multipliers.

The organization of the rest of this article is as follows: In the next section, we briefly review the conventional normal basis multiplication scheme, which relies on inner product operations over the ground field. In Section 3, first we prove a number of results for the normal basis *multiplication matrix* and then derive an algorithm for multiplication over $\mathrm{GF}(2^m)$. We also give the computational complexity of the algorithm in terms of the number of bit level operations needed. This algorithm is then adapted for its easy software implementation on general purpose processors. In Section 4, we apply the above algorithm to a special class of normal bases, namely, the type-I optimal normal basis, and we give an exact analysis for this case and compare our results with those of existing schemes. Then, in Section 5, we consider finite fields $\mathrm{GF}(2^m)$ where m is a composite number. For such composite finite fields, we give a multiplication

1. Conventionally, the space complexity of the $\mathrm{GF}(2^m)$ multiplier is given in terms of the number of logic gates, namely XOR and AND gates, which correspond to GR(2) (i.e., bit level) addition and multiplication, respectively.

- *A. Rayhani-Masoleh is with the Centre for Applied Cryptographic Research, Combinatorics and Optimization Department, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. E-mail: areyhani@math.uwaterloo.ca.*
- *M.A. Hasan is with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. E-mail: ahasan@ece.uwaterloo.ca.*

Manuscript received 17 May 2002; revised 25 Nov. 2002; accepted 25 Nov. 2002.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 117864.

algorithm, its complexity, and comparison results. Finally, we make a few concluding remarks in Section 6.

2 Preliminaries

2.1 Normal Basis Representation

It is well-known that there exists a normal basis (NB) in the field $GF(2^m)$ over $GF(2)$ for all positive integers m. By finding an element $\beta \in GF(2^m)$ such that $\{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$ is a basis of $GF(2^m)$ over $GF(2)$, any element $A \in GF(2^m)$ can be represented as

$$A = \sum_{j=0}^{m-1} a_j \beta^{2^j} = a_0\beta + a_1\beta^2 + \cdots + a_{m-1}\beta^{2^{m-1}}, \tag{1}$$

where $a_j \in GF(2)$, $0 \le j \le m-1$, is the jth coordinate of A. In short, this normal basis representation of A will be written as $A = (a_0, a_1, \cdots, a_{m-1})$. In vector notation, (1) will be written as

$$A = \underline{a} \cdot \underline{\beta}^T = \underline{\beta} \cdot \underline{a}^T, \tag{2}$$

where $\underline{a} = [a_0, a_1, \cdots, a_{m-1}]$, $\underline{\beta} = [\beta, \beta^2, \cdots, \beta^{2^{m-1}}]$, and T denotes vector transposition.

The main advantage of the NB representation is that an element A can be easily squared by a cyclic shift of its coordinates since

$$\begin{aligned} A^2 &= (a_{m-1}, a_0, \cdots, a_{m-2}) \\ &= a_{m-1}\beta + a_0\beta^2 + \cdots + a_{m-2}\beta^{2^{m-1}}. \end{aligned} \tag{3}$$

2.2 Normal Basis Multiplication

Let A and B be any two elements of $GF(2^m)$ and be represented with respect to (w.r.t.) the NB as $A = \sum_{i=0}^{m-1} a_i\beta^{2^i}$ and $B = \sum_{j=0}^{m-1} b_j\beta^{2^j}$, respectively. Let C denote their product, i.e.,

$$C = A \cdot B = (\underline{a} \cdot \underline{\beta}^T) \cdot (\underline{\beta} \cdot \underline{b}^T) = \underline{a} \cdot \mathbf{M} \cdot \underline{b}^T, \tag{4}$$

where the multiplication matrix $\mathbf{M}$ is defined by

$$\begin{aligned} \mathbf{M} &= \underline{\beta}^T \cdot \underline{\beta} = \left[\beta^{2^i+2^j}\right] \\ &= \begin{bmatrix} \beta^{2^0+2^0} & \beta^{2^0+2^1} & \cdots & \beta^{2^0+2^{m-1}} \\ \beta^{2^1+2^0} & \beta^{2^1+2^1} & \cdots & \beta^{2^1+2^{m-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{2^{m-1}+2^0} & \beta^{2^{m-1}+2^1} & \cdots & \beta^{2^{m-1}+2^{m-1}} \end{bmatrix}. \end{aligned} \tag{5}$$

All entries of $\mathbf{M}$ belong to GF(2^m) and if they are written w.r.t. the NB, then the following is obtained

$$\mathbf{M} = \mathbf{M}_0\beta + \mathbf{M}_1\beta^2 + \cdots + \mathbf{M}_{m-1}\beta^{2^{m-1}}, \tag{6}$$

where $\mathbf{M}_i$s are $m \times m$ matrices whose entries belong to $GF(2)$. Substituting (6) into (4), the coordinates of C are found as follows:

$$\begin{aligned} c_i &= \underline{a} \cdot \mathbf{M}_i \cdot \underline{b}^T \quad 0 \le i \le m-1 \\ &= \underline{a}^{(i)} \cdot \mathbf{M}_0 \cdot \underline{b}^{(i)^T} \quad 0 \le i \le m-1, \end{aligned} \tag{7}$$

where $\underline{a}^{(i)}$ is the i-fold left cyclic shift of $\underline{a}$ and the same is for $\underline{b}^{(i)^T}$ [6].

Example 1. Consider the finite field $GF(2^5)$ generated by the irreducible polynomial $F(z) = z^5 + z^2 + 1$ and let α be its root, i.e., $F(\alpha) = 0$. We choose $\beta = \alpha^3$, then $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$ is a normal basis. Then, using Table 1 in [13], we have

$$\mathbf{M}_0 = \begin{bmatrix} 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Let A and B be two elements in $GF(2^5)$, whose representations w.r.t. the normal basis are

$$A = (a_0, a_1, \cdots, a_4) = \sum_{i=0}^{4} a_i\beta^{2^i}$$

and

$$B = (b_0, b_1, \cdots, b_4) = \sum_{i=0}^{4} b_i\beta^{2^i}.$$

Thus, using (7), the coordinates of C are computed as

$$\begin{aligned} c_i = {} & a_{-i}(b_{2-i} + b_{3-i} + b_{4-i}) + a_{1-i}(b_{3-i} + b_{4-i}) \\ & + a_{2-i}(b_{-i} + b_{3-i}) + a_{3-i}(b_{-i} + b_{1-i} + b_{2-i} + b_{4-i}) \\ & + a_{4-i}(b_{-i} + b_{1-i} + b_{3-i} + b_{4-i}), \quad 0 \le i \le 4, \end{aligned}$$

where subtractions in subscripts are performed modulo 5.

Definition 1. *The numbers of 1s in all $\mathbf{M}_i$s are equal. Let us define this number by*

$$C_N = H(\mathbf{M}_i), \quad 0 \le i \le m-1, \tag{8}$$

which is known as the complexity of the NB [13]. In (8), $H(\mathbf{M}_i)$ refers to the Hamming weight, i.e., the number of 1s, in $\mathbf{M}_i$.

3 A New Multiplication Scheme

3.1 Multiplication Matrix Revisited

In (5), the multiplication matrix $\mathbf{M}$ is symmetric, i.e., $\mathbf{M} = \mathbf{M}^T$ and its diagonal entries are the elements of the NB. Denoting $\mathbf{M}$ as $[\mu_{i,j}]_{i,j=0}^{m-1}$, where $\mu_{i,j} = \mu_{j,i} = \beta^{2^i+2^j}$, it is easy to see that

$$\mu_{i,j} = \mu_{i-1,j-1}^2, \quad 0 < i, j \le m-1.$$

Thus, given the m entries of the 0th row of $\mathbf{M}$, the generation of all its other entries (except the leftmost entries) require at most some squaring operations, which are, however, essentially free of cost in a normal basis representation. Now, if we let

$$\delta_j = \beta^{1+2^j} \qquad j = 0,\ 1,\ \cdots,\ v, \tag{9}$$

where $v = \lfloor \frac{m}{2} \rfloor$, then the entries of $\mathbf{M}$ can be conveniently obtained from δ_js, as stated in the following lemma.

Lemma 1. *For the multiplication matrix $\mathbf{M} = [\mu_{i,j}]_{i,j=0}^{m-1}$, the following holds:*

$$\mu_{i,j} = \mu_{j,i} = \begin{cases} \delta_{j-i}^{2^i}, & 0 < j-i \leq v, \\ \delta_{m+i-j}^{2^j}, & v < j-i \leq m-1. \end{cases} \tag{10}$$

Proof. Since $\mathbf{M}$ is symmetric, $\mu_{i,j} = \mu_{j,i}$. For $0 < i < j \leq v$,

$$\mu_{i,j} = \beta^{2^i+2^j} = \left(\beta^{1+2^{j-i}}\right)^{2^i} = \delta_{j-i}^{2^i}.$$

Now, for $v < i < j \leq m-1$, we have $j - i > v$. Thus, $m - (j-i) \leq v$ and the following holds:

$$\mu_{i,j} = \beta^{2^i+2^j} = \left(\beta^{2^{m+i-j}+1}\right)^{2^j} = \delta_{m+i-j}^{2^j}.$$

□

Noting that

$$\delta_{m-1-v} = \begin{cases} \delta_v & \text{for } m \text{ odd}, \\ \delta_{v-1} & \text{for } m \text{ even}, \end{cases} \tag{11}$$

and

$$\delta_v \equiv \delta_v^{2^v} \quad \text{for } m \text{ even}, \tag{12}$$

the multiplication matrix can be written as

$$\mathbf{M} = \begin{bmatrix} \delta_0 & \delta_1 & \cdots & \delta_v & \delta_{m-1-v}^{2^{v+1}} & \delta_{m-2-v}^{2^{v+2}} & \cdots & \delta_1^{2^{m-1}} \\ \delta_1 & \delta_0^2 & \cdots & \delta_{v-1}^2 & \delta_v^2 & \delta_{m-1-v}^{2^{v+2}} & \cdots & \delta_2^{2^{m-1}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \delta_v & \delta_{m-1-v}^{2^{v+1}} & \cdots & \delta_0^{2^v} & \delta_1^{2^v} & \delta_2^{2^v} & \cdots & \delta_{m-1-v}^{2^v} \\ \delta_{m-1-v}^{2^{v+1}} & \delta_v^2 & \cdots & \delta_1^{2^v} & \delta_0^{2^{v+1}} & \delta_1^{2^{v+1}} & \cdots & \delta_{m-2-v}^{2^{v+1}} \\ \delta_{m-2-v}^{2^{v+2}} & \delta_{m-1-v}^{2^{v+2}} & \cdots & \delta_2^{2^v} & \delta_1^{2^{v+1}} & \delta_0^{2^{v+2}} & \cdots & \delta_{m-3-v}^{2^{v+2}} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \delta_1^{2^{m-1}} & \delta_2^{2^{m-1}} & \cdots & \delta_{m-1-v}^{2^v} & \delta_{m-2-v}^{2^{v+1}} & \delta_{m-3-v}^{2^{v+2}} & \cdots & \delta_0^{2^{m-1}} \end{bmatrix}. \tag{13}$$

Now, we write $\mathbf{M}$ as a sum of m matrices as follows:

$$\mathbf{M} = \mathbf{M}^{(0)} + \mathbf{M}^{(1)} + \cdots + \mathbf{M}^{(m-1)} \tag{14}$$

such that the nonzero entries of $\mathbf{M}^{(l)}$, $0 \leq l \leq m-1$, belong to $\{\delta_0^{2^l}, \delta_1^{2^l}, \cdots, \delta_v^{2^l}\}$. As an example, for $m = 5$, the representation of $\mathbf{M}$ given in (14) is as follows:

$$\mathbf{M} = \begin{bmatrix} \delta_0 & \delta_1 & \delta_2 & 0 & 0 \\ \delta_1 & 0 & 0 & 0 & 0 \\ \delta_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & \delta_0^2 & \delta_1^2 & \delta_2^2 & 0 \\ 0 & \delta_1^2 & 0 & 0 & 0 \\ 0 & \delta_2^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \delta_0^{2^2} & \delta_1^{2^2} & \delta_2^{2^2} \\ 0 & 0 & \delta_1^{2^2} & 0 & 0 \\ 0 & 0 & \delta_2^{2^2} & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & \delta_2^{2^3} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \delta_2^{2^3} & 0 & 0 & \delta_0^{2^3} & \delta_1^{2^3} \\ 0 & 0 & 0 & \delta_1^{2^3} & 0 \end{bmatrix}$$
$$+ \begin{bmatrix} 0 & 0 & 0 & 0 & \delta_1^{2^4} \\ 0 & 0 & 0 & 0 & \delta_2^{2^4} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ \delta_1^{2^4} & \delta_2^{2^4} & 0 & 0 & \delta_0^{2^4} \end{bmatrix}.$$

From the structure of $\mathbf{M}$ given in (13), it is clear that these nonzero entries of $\mathbf{M}^{(l)}$ exist only along its row l and column l. Since $\mathbf{M} = \mathbf{M}^T$, we have $\mathbf{M}^{(l)} = \left(\mathbf{M}^{(l)}\right)^T$ and, hence, the lth column of $\mathbf{M}^{(l)}$ is the transpose of its lth row. The latter can be obtained by using (12) and (13), and, for m odd, it is given by

$$\mu_{l,*}^{(l)} = \begin{cases} [\underbrace{0,\ 0,\ \cdots,\ 0}_{l \text{ zeros}}, \delta_0^{2^l},\ \delta_1^{2^l},\ \cdots, \delta_v^{2^l}, \\ \underbrace{0,\ 0,\ \cdots,\ 0}_{m-l-v-1 \text{ zeros}}], & 0 \leq l \leq v, \\ [\delta_{m-l}^{2^l},\ \delta_{m-l+1}^{2^l},\ \cdots, \delta_v^{2^l},\ \underbrace{0,\ 0,\ \cdots,\ 0}_{m-v-1 \text{ zeros}}, & v+1 \leq l \leq m-1, \\ \delta_0^{2^l},\ \delta_1^{2^l},\ \cdots, \delta_{m-l-1}^{2^l}], \end{cases} \tag{15}$$

and for m even

$$\mu_{l,*}^{(l)} = \begin{cases} [\underbrace{0,\ 0,\ \cdots,\ 0}_{l \text{ zeros}}, \delta_0^{2^l},\ \delta_1^{2^l},\ \cdots, \delta_v^{2^l}, \\ \underbrace{0,\ 0,\ \cdots,\ 0}_{m-l-v-1 \text{ zeros}}], & 0 \leq l \leq v-1, \\ [\underbrace{0,\ 0,\ \cdots,\ 0}_{m-v \text{ zeros}},\ \delta_0^{2^l},\ \delta_1^{2^l},\ \cdots, \delta_{v-1}^{2^l}], & l = v, \\ [\delta_{m-l}^{2^l},\ \delta_{m-l+1}^{2^l},\ \cdots, \delta_{v-1}^{2^l}, \\ \underbrace{0,\ 0,\ \cdots,\ 0}_{m-v \text{ zeros}}, \delta_0^{2^l},\ \delta_1^{2^l},\ \cdots, \delta_{m-l-1}^{2^l}], & v+1 \leq l \leq m-1. \end{cases} \tag{16}$$

Thus, the following lemma holds.

Lemma 2. *For $0 \leq i, l \leq m-1$, let us denote the number of nonzero entries of the ith row and the ith column of $\mathbf{M}^{(l)}$ as $H(\mu_{i,*}^{(l)})$ and $H(\mu_{*,i}^{(l)})$, respectively. If $i \neq l$, then $H(\mu_{i,*}^{(l)}) = H(\mu_{*,i}^{(l)}) \in \{0,\ 1\}$. For $i = l$, there are two cases depending on m. If m is odd,*

$$H(\mu_{l,*}^{(l)}) = H(\mu_{*,l}^{(l)}) = v+1, \quad \forall l,$$

and, for m even,

$$H(\mu_{l,*}^{(l)}) = H(\mu_{*,l}^{(l)}) = \begin{cases} v+1, & 0 \le l \le v-1, \\ v, & v \le l \le m-1. \end{cases}$$

Corollary 1. *Let $H(\mathbf{M}^{(l)})$, $0 \le l \le m-1$, denote the number of nonzero entries of $\mathbf{M}^{(l)}$. Then, for m odd,*

$$H(\mathbf{M}^{(l)}) = 2v+1, \ \forall l,$$

and, for m even,

$$H(\mathbf{M}^{(l)}) = \begin{cases} 2v+1, & 0 \le l \le v-1, \\ 2v-1, & v \le l \le m-1. \end{cases}$$

Proof. We note that $\mu_{l,l}^{(l)} = \delta_0^{2^l}$. Since the nonzero entries of $\mathbf{M}^{(l)}$ lie only in its row l and column l, we have

$$H(\mathbf{M}^{(l)}) = H(\mu_{l,*}^{(l)}) + H(\mu_{*,l}^{(l)}) - H(\mu_{l,l}^{(l)}).$$

The proof then follows from Lemma 2. □

Now, we give another lemma which will be useful in our algorithm formulation presented in the next section.

Lemma 3. *For δ and v as defined above, the following holds:*

$$\sum_{j=1}^{v}\left(\delta_j + \delta_j^{2^{((-j))}}\right) = \begin{cases} \delta_0 + \delta_0^{2^{-1}} & \text{for } m \text{ odd}, \\ \delta_0 + \delta_0^{2^{-1}} + \delta_v & \text{for } m \text{ even}, \end{cases} \tag{17}$$

where $((x))$ indicates x modulo the degree of the field under consideration (i.e., m).

Proof. From (9), we have

$$\delta_j^{2^{((-j))}} = (\beta^{1+2^j})^{2^{m-j}} = \beta^{2^{m-j}+2^m} = \beta^{1+2^{m-j}}.$$

Thus,

$$\text{L.H.S.} = \sum_{j=1}^{v}(\delta_j + \delta_j^{2^{((-j))}}) = \sum_{j=1}^{v}(\beta^{1+2^j} + \beta^{1+2^{m-j}})$$
$$= \beta\sum_{j=1}^{v}(\beta^{2^j} + \beta^{2^{m-j}})$$
$$= \beta(\beta^2 + \beta^{2^2} + \cdots + \beta^{2^v} + \beta^{2^{m-1}} + \beta^{2^{m-2}} + \cdots + \beta^{2^{m-v}}).$$

For the normal basis $\{\beta, \beta^2, \cdots, \beta^{2^{m-1}}\}$, one has $\sum_{i=0}^{m-1}\beta^{2^i} = 1$. Now, noting that

$$m - v = m - \left\lfloor\frac{m}{2}\right\rfloor = \begin{cases} m - \frac{m-1}{2} = v+1 & \text{for } m \text{ odd}, \\ m - \frac{m}{2} = v & \text{for } m \text{ even}, \end{cases}$$

we can write:

$$\text{L.H.S.} = \begin{cases} \beta\sum_{i=1}^{m-1}\beta^{2^i} = \beta^2 + \beta = \delta_0 + \delta_0^{2^{-1}}, & \text{for } m \text{ odd}, \\ \beta\left(\beta^{2^v} + \sum_{i=1}^{m-1}\beta^{2^i}\right) = \beta^2 + \beta + \beta^{1+2^v} \\ = \delta_0 + \delta_0^{2^{-1}} + \delta_v, & \text{for } m \text{ even}. \end{cases}$$

□

3.2 Algorithm Formulation

Lemma 4. *Let A and B be two elements of $GF(2^m)$ and C be their product. Then,*

$$C = \begin{cases} \sum_{i=0}^{m-1} a_i b_i \delta_0^{2^{i-1}} + \sum_{i=0}^{m-1}\sum_{j=1}^{v} y_{i,j}\delta_j^{2^i}, & \text{for } m \text{ odd} \\ \sum_{i=0}^{m-1} a_i b_i \delta_0^{2^{i-1}} + \sum_{i=0}^{m-1}\sum_{j=1}^{v-1} y_{i,j}\delta_j^{2^i} + \sum_{i=0}^{v-1} y_{i,v}\delta_v^{2^i}, & \text{for } m \text{ even}, \end{cases} \tag{18}$$

where

$$y_{i,j} \triangleq (a_i + a_{((i+j))})(b_i + b_{((i+j))}), \ 1 \le j \le v, \ 0 \le i \le m-1. \tag{19}$$

Proof. Here, we present the case of m odd only. The case of m even is similar.

From (4) and (14),

$$C = \underline{a} \cdot \mathbf{M} \cdot \underline{b}^T = \sum_{i=0}^{m-1} \underline{a} \cdot \mathbf{M}^{(i)} \cdot \underline{b}^T.$$

Let $C^{(i)} = \underline{a} \cdot \mathbf{M}^{(i)} \cdot \underline{b}^T$. Using (15), for $0 \le i \le v$, we then have

$$C^{(i)} = [\underbrace{0,\ 0,\ \cdots,\ 0}_{i \text{ zeros}}, \sum_{j=0}^{v} a_{i+j}\delta_j^{2^i},\ a_i\delta_1^{2^i},\ \cdots,$$
$$a_i\delta_v^{2^i}, \underbrace{0,\ 0,\ \cdots,\ 0}_{m-i-v-1 \text{ zeros}}] \cdot \underline{b}^T$$
$$= \sum_{j=0}^{v} a_{i+j}b_i\delta_j^{2^i} + \sum_{j=1}^{v} a_i b_{i+j}\delta_j^{2^i}$$
$$= a_i b_i \delta_0^{2^i} + \sum_{j=1}^{v}(a_i b_{i+j} + a_{i+j}b_i)\delta_j^{2^i}$$

and, for $v+1 \le i \le m-1$,

$$C^{(i)} = [a_i\delta_{m-i}^{2^i},\ a_i\delta_{m-i+1}^{2^i},\ \cdots, a_i\delta_v^{2^i}, \underbrace{0,\ 0,\ \cdots,\ 0}_{m-v-1 \text{ zeros}},$$
$$\sum_{j=0}^{v} a_{((i+j))}\delta_j^{2^i},\ a_i\delta_1^{2^i},\ \cdots, a_i\delta_{m-i-1}^{2^i}] \cdot \underline{b}^T$$
$$= \sum_{j=0}^{v} a_{((i+j))}b_i\delta_j^{2^i} + \sum_{j=1}^{v} a_i b_{((i+j))}\delta_j^{2^i}$$
$$= a_i b_i \delta_0^{2^i} + \sum_{j=1}^{v}(a_i b_{((i+j))} + a_{((i+j))}b_i)\delta_j^{2^i}.$$

Noting that $i+j = ((i+j))$ for $0 \le i,j \le v$, we then have

$$
\begin{aligned}
C &= C^{(0)} + C^{(1)} + \cdots + C^{(m-1)} \\
&= \sum_{i=0}^{m-1} a_i b_i \delta_0^{2^i} + \sum_{i=0}^{m-1}\sum_{j=1}^{v} (a_i b_{((i+j))} + a_{((i+j))} b_i)\delta_j^{2^i} \\
&= \sum_{i=0}^{m-1} a_i b_i \delta_0^{2^i} + \sum_{i=0}^{m-1}\sum_{j=1}^{v} \left(a_i b_i + a_{((i+j))} b_{((i+j))}\right)\delta_j^{2^i} \\
&\quad + \sum_{i=0}^{m-1}\sum_{j=1}^{v} y_{i,j}\delta_j^{2^i} \quad \text{(using (19))}.
\end{aligned}
$$

After expansion and reindexing, one can verify that

$$
\sum_{i=0}^{m-1}\sum_{j=1}^{v} a_{((i+j))} b_{((i+j))}\delta_j^{2^i} = \sum_{i=0}^{m-1}\sum_{j=1}^{v} a_i b_i \delta_j^{2^{((i-j))}}.
$$

Now, we can write

$$
\begin{aligned}
C &= \sum_{i=0}^{m-1} a_i b_i \delta_0^{2^i} + \sum_{i=0}^{m-1}\sum_{j=1}^{v}\left(a_i b_i \delta_j^{2^i} + a_i b_i \delta_j^{2^{((i-j))}}\right) + \sum_{i=0}^{m-1}\sum_{j=1}^{v} y_{i,j}\delta_j^{2^i} \\
&= \sum_{i=0}^{m-1} a_i b_i \left(\delta_0 + \sum_{j=1}^{v}\left(\delta_j + \delta_j^{2^{((-j))}}\right)\right)^{2^i} + \sum_{i=0}^{m-1}\sum_{j=1}^{v} y_{i,j}\delta_j^{2^i}.
\end{aligned}
$$

Then, using Lemma 3, the proof is complete. □

Let h_j, $1 \le j \le v$, be the number of nonzero coordinates of the normal basis representation of δ_j, i.e., $h_j = H(\delta_j)$, and let $w_{j,1}, w_{j,2}, \cdots, w_{j,h_j}$ denote the positions of such coordinates, i.e.,

$$
\delta_j = \sum_{k=1}^{h_j} \beta^{2^{w_{j,k}}},\ 1 \le j \le v, \tag{20}
$$

where $0 \le w_{j,1} < w_{j,2} < \cdots < w_{j,h_j} \le m-1$. Also, for even values of m, we have $v = \frac{m}{2}$ and $\delta_v = \delta_v^{2^{\frac{m}{2}}}$. This implies that, in the normal basis representation of δ_v, its ith coordinate is equal to its $((\frac{m}{2}+i))$-th coordinate. Thus, h_v is even and we can write

$$
\delta_v = \sum_{k=1}^{\frac{h_v}{2}}\left(\beta^{2^{w_{v,k}}} + \beta^{2^{w_{v,k}+v}}\right), \quad v = \frac{m}{2}. \tag{21}
$$

Now, substituting (20) and (21) into (18) and noting that $\delta_0^{2^{i-1}} = \beta^{2^i}$, we have the following theorem.

Theorem 1. *Let A and B be two elements of $GF(2^m)$ and C be their product. Then,*

$$
C = \begin{cases}
\sum\limits_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum\limits_{j=1}^{v}\sum\limits_{k=1}^{h_j}\left(\sum\limits_{i=0}^{m-1} y_{((i-w_{j,k})),j}\beta^{2^i}\right), & \text{for } m \text{ odd} \\
\sum\limits_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum\limits_{j=1}^{v-1}\sum\limits_{k=1}^{h_j}\left(\sum\limits_{i=0}^{m-1} y_{((i-w_{j,k})),j}\beta^{2^i}\right) + F, & \text{for } m \text{ even,}
\end{cases} \tag{22}
$$

where

$$
F = \sum_{k=1}^{\frac{h_v}{2}}\sum_{i=0}^{v-1} y_{((i-w_{v,k})),v}\left(\beta^{2^i} + \beta^{2^{i+v}}\right) \text{ and } v = \frac{m}{2}.
$$

Note that, for a normal basis, the representation of δ_j is fixed and so is $w_{j,k}$, $1 \le j \le v$, $1 \le k \le h_j$. Theorem 1 is valid for any normal basis of GF(2^m) over GF(2). A bit level version of (22) has recently been reported in [3] for the special case of type-II optimal normal bases. Based on (22), now we have the following algorithm for low complexity normal basis (LCNB) multiplication.

Algorithm 1. (Low Complexity Normal Basis Multiplication over $GF(2^m)$

Input: $A, B \in GF(2^m)$, $w_{j,k}$, $1 \le j \le v$, $1 \le k \le h_j$

Output: $C = AB$

1. Generate $y_{i,j} = (a_i + a_{((i+j))})(b_i + b_{((i+j))})$, $1 \le j < v$, $0 \le i \le m-1$, where $y_{i,j} \in GF(2)$.
2. Initialize $c_i := a_i b_i$, $0 \le i \le m-1$, $C := (c_0, c_1, \cdots, c_{m-1})$
3. For $j = 1$ to $v-1$ {
4. $T := (t_0, t_1, \cdots, t_{m-1}) = 0$
5. For $k = 1$ to h_j {
6. $r_i := y_{((i-w_{j,k})),j}$, $0 \le i \le m-1$, $R := (r_0, r_1, \cdots, r_{m-1})$
7. $T := T + R$
8. }
9. $C := C + T$
10. }
11. $T := 0$
12. If m is odd,
13. $s := h_v$, $t := m$
14. else $s := \frac{h_v}{2}$, $t := \frac{m}{2}$
15. Generate $y_{i,v} = (a_i + a_{((v+i))})(b_i + b_{((v+i))})$, $0 \le i \le t-1$,
16. If m is even $y_{i+v,v} = y_{i,v}$, $0 \le i \le \frac{m}{2}-1$
17. For $k = 1$ to s
18. $r_i := y_{((i-w_{v,k})),v}$, $0 \le i \le t-1$
19. If m is even,
20. $r_{i+\frac{m}{2}} := r_i$, $0 \le i \le \frac{m}{2}-1$, $R := (r_0, r_1, \cdots, r_{\frac{m}{2}-1}, r_0, r_1, \cdots, r_{\frac{m}{2}-1})$
21. $T := T + R$
22. }
23. $C := C + T$

Example 2. To illustrate the operation of the above algorithm, we again use the field GF(2^5) and its normal basis, as described in Example 1. Here, $m = 5$ and $v = \lfloor \frac{5}{2} \rfloor = 2$. Using Table 1 in [13], one has

$$
\begin{aligned}
&\delta_1 = \beta^3 = \beta^2 + \beta^4 + \beta^8, \\
&h_1 = 3, \\
&[w_{1,k}]_{k=1}^{h_1} = [1,\ 2,\ 3], \\
&\delta_2 = \beta^5 = \beta + \beta^2 + \beta^4 + \beta^{16}, \\
&h_2 = 4, \\
&[w_{2,k}]_{k=1}^{h_2} = [0,\ 1,\ 2,\ 4].
\end{aligned}
$$

Let $A = \beta^2 + \beta^4 + \beta^8 = (01110)$ and $B = \beta + \beta^4 + \beta^{16} = (10101)$ be two field elements. The generation of $y_{i,j}$s in line 1 of the LCNB multiplication algorithm is shown in Table 1a. Table 1b shows contents of variables R and C in the order they are updated by the execution of the algorithm. In this table, the row with j being "-" indicates the initialization step (i.e., line 2) of the algorithm. The final contents of C represent the product of A and B.

TABLE 1
(a) Generation of $y_{i,j}$ in Line 1 of Algorithm 1; (b) Contents of R and C during the Execution of Algorithm 1 with $A = (01110)$ and $B = (10101)$

i	$y_{i,j}$ $j=1$	$y_{i,j}$ $j=2$
0	1	0
1	0	0
2	0	0
3	1	1
4	0	1

(a)

j	h_j	k	$w_{j,k}$	R	C
-	-	-	-	-	00100
1	3	1	1	01001	01101
		2	2	10100	11001
		3	3	01010	10011
2	4	1	0	00011	10000
		2	1	10001	00001
		3	2	11000	11001
		4	4	00110	11111

(b)

3.3 Complexity and Comparison

Lemma 5. [9]. *For h_j as defined above, the complexity of the normal basis N is*

$$C_N = 2\left(\sum_{j=1}^{v-1} h_j + \epsilon h_v\right) + 1, \tag{23}$$

where

$$\epsilon \triangleq \begin{cases} 1 & \text{for } m \text{ odd} \\ 0.5 & \text{for } m \text{ even.} \end{cases} \tag{24}$$

Theorem 2. *For the LCNB multiplication algorithm, let $\#Mult_{LCNB}$ and $\#Add_{LCNB}$ denote the numbers of bit level multiplications and additions, respectively. Then,*

$$\#Mult_{LCNB} = \frac{m(m+1)}{2}, \tag{25}$$

$$\#Add_{LCNB} = \frac{m}{2}(C_N + 2m - 3 - (1-\epsilon)(h_v - 2)). \tag{26}$$

Proof. The number of bit level multiplications in lines 1, 2, and 15 of Algorithm 1 are $m(v-1)$, m, and t, respectively. Thus, the total number of such multiplications is $mv + t = \frac{m(m+1)}{2}$. The number of additions consists of two parts: 1) the bit level additions of lines 1 and 15, which are $2m(v-1)$ and $2t$, respectively, and 2) the word level additions of lines 7, 9, 21, and 23. The bit level additions of lines 7 and 9 without considering the first addition of line 7 with $T = 0$ is $m\sum_{j=1}^{v-1} h_j$. Similarly, the bit level additions of line 23 is m. For line 21, the number of bit additions is $(s-1)t$ because, for even values of m, half of the bits of R (and, hence, T) are the same as the other half bits. Thus, the total number of bit level additions is

$$\#\mathrm{Add}_{\mathrm{LCNB}} = 2m(v-1) + 2t + m\sum_{j=1}^{v-1} h_j + m + (s-1)t. \tag{27}$$

Using (23) and noting that $s = \epsilon h_v$, $t = \epsilon m$, (27) gives the proof. □

TABLE 2
Comparison of Normal Basis Multipliers

Multipliers	#Mult	#Add	Total bit operations
MO [25]	m^2	$m(C_N - 1)$	$m(C_N + m - 1)$
RR_MO [20]	m^2	$\leq \frac{m}{2}(C_N + m - 2)$	$\leq \frac{m}{2}(C_N + 3m - 2)$
LCNB	$\frac{m(m+1)}{2}$	$\leq \frac{m}{2}(C_N + 2m - 3)$	$\leq \frac{m}{2}(C_N + 3m - 2)$

Remark 1. *In order to have a bit-parallel implementation of Algorithm 1, one needs to generate all $y_{i,j}$s and a_ib_is using $\frac{m(m+1)}{2}$ two input-AND gates and $m(m-1)$ two-input XOR gates and the corresponding time delay is $T_A + T_X$, where T_A and T_X are time delays due to an AND gate and an XOR gate, respectively. In lines 6 and 18 of the algorithm, when we add r_is and a_ib_is to obtain c_is, we need a total of $\sum_{j=1}^{v-1} h_j + \epsilon h_v = \frac{C_N - 1}{2}$ XOR gates. If these gates are arranged in a binary tree fashion, then the corresponding time complexity is $\lceil \log_2 \frac{C_N+1}{2} \rceil T_X = (\lceil \log_2(C_N+1) \rceil - 1)T_X$. Thus, the overall time complexity of the bit-parallel structure is $T_A + \lceil \log_2(C_N+1) \rceil T_X$. Since C_N is an odd integer, one has $\lceil \log_2(C_N+1) \rceil = \lceil \log_2 C_N \rceil$. Thus, the time complexity is simplified to*

$$Time\ delay = T_A + \lceil \log_2 C_N \rceil T_X. \tag{28}$$

Table 2 compares the number of bit level operations of the LCNB algorithm with those of the Massey-Omura (MO) multiplier of [25] and the reduced redundancy Massey-Omura (RR_MO) multiplier of [20]. The multipliers of [25] and [20] are used for comparison as they appear to be the first and the most recently reported work in this area and it seems the total number of bit level operations of [20] is the least among the existing normal basis schemes. All the multipliers in Table 2 have the same time delay $T_A + \lceil \log_2 C_N \rceil T_X$ in bit-parallel implementation. As can be seen from the table, the total number of bit level operations of our new LCNB algorithm matches that of [20]. More importantly, the LCNB algorithm has the least number of bit level multiplications that meets the lower bound on the number of bit level multiplications determined in [3]. Since the bit level multiplication corresponds to the multiplication in the ground field GF(2), if the algorithm is extended to a ground field of degree more than one, where a multiplication is more expensive than an addition operation, the use of the LCNB algorithm will be advantageous. This is investigated in Section 5 of this paper.

Remark 2. *In Table 2, the numbers of bit level additions (and, consequently, the total operations) are given in terms of C_N. It is well-known that $C_N \geq 2m - 1$ [13]. If a normal basis has* *minimum C_N, i.e., $C_N = 2m-1$, then it is referred to as an optimal normal basis (ONB). There are two types of ONBs, namely, type-I and type-II, which are hereafter also referred to as ONB-I and ONB-II, respectively. The ONBs do not exist for all m. The list in [12] shows that only 23 percent of $m \leq 2{,}000$ have ONBs. For a given m where an ONB exists, the minimum number of bit level additions needed in the LCNB algorithm can be obtained by substituting $C_N = 2m-1$ in (26), i.e., for an ONB we have*

$$\#Add_{LCNB} = 2m(m-1). \tag{29}$$

Recent results on multipliers using the special case of ONB-II include references [23] and [3], which have the same space and time complexities as those presented here. In Section 4, we show that the number of bit level additions can be further reduced by considering ONB-I.

3.4 Multiplication on General Purpose Processors

General purpose processors, such as Intel's Pentium processors, are not usually designed to efficiently add l bits over GF(2), using a single (XOR or such) instruction, even when l is less than the size of the internal registers of the processor. However, the conventional approach[2] to normal basis multiplication relies on inner products over GF(2), as shown in (7), and requires about $\frac{m^2}{2}$ modulo 2 additions, on average, for each coordinate of the product. Hence, this approach is considered not to be very efficient. Below, we present a normal basis multiplication algorithm, which is a variant of the LCNB algorithm and is suitable for software implementation. From (22), we can write

$$C = \begin{cases} \sum_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum_{j=1}^{v} \sum_{k=1}^{h_j} \left(\sum_{i=0}^{m-1} y_{i,j} \beta^{2^i} \right)^{2^{w_{j,k}}}, & \text{for } m \text{ odd} \\ \sum_{i=0}^{m-1} a_i b_i \beta^{2^i} + \sum_{j=1}^{v-1} \sum_{k=1}^{h_j} \left(\sum_{i=0}^{m-1} y_{i,j} \beta^{2^i} \right)^{2^{w_{j,k}}} + D, & \text{for } m \text{ even,} \end{cases} \tag{30}$$

where

$$D = \sum_{k=1}^{\frac{h_v}{2}} \left(\sum_{i=0}^{v-1} y_{i,v} (\beta^{2^i} + \beta^{2^{i+v}}) \right)^{2^{w_{v,k}}} \quad \text{and } v = \frac{m}{2}. \tag{31}$$

Let us define

$$\Delta w_{j,k} \triangleq w_{j,k} - w_{j,k-1},\ 1 \leq j \leq v,\ 1 \leq k \leq h_j,\ w_{j,0} = 0, \tag{32}$$

where $w_{j,k}$s are the positions of 1s in the normal basis representation of δ_j as defined in (20). For a particular normal basis, all $w_{j,k}$s are fixed. Hence, all $\Delta w_{j,k}$s need to be determined only once, i.e., at the time of choosing the basis.

Let $A \odot B$ denote the bitwise AND operations between the coordinates of $A = (a_0, a_1, \cdots, a_{m-1})$ and $B = (b_0, b_1, \cdots, b_{m-1})$, i.e.,

$$A \odot B \triangleq (a_0 b_0, a_1 b_1, \cdots, a_{m-1} b_{m-1}).$$

Let us denote i-fold left and right cyclic shifts of the coordinates of A by $A \lll i$ and $A \ggg i$, respectively.

Based on (30), a software version of LCNB (referred to as S-LCNB) multiplication algorithm can then be stated as follows:

Algorithm 2. (S-LCNB Multiplication over $GF(2^m)$)
Input: $A,\ B \in GF(2^m)$, $\Delta w_{j,k}$, $1 \leq j \leq v$, $1 \leq k \leq h_j$
Output: $C = AB$

```
1.  Initialize C := A ⊙ B, S_A := A, S_B := B
2.  For j = 1 to v − 1
3.    S_A ⋘ 1, S_B ⋘ 1
4.    L_A := A + S_A, L_B := B + S_B
5.    R := L_A ⊙ L_B
6.    For k = 1 to h_j
7.        R ⋙ Δw_{j,k}
8.        C := C + R
9.    }
10. }
11. S_A ⋘ 1, S_B ⋘ 1
12. L_A := A + S_A, L_B := B + S_B
13. R := L_A ⊙ L_B
14. If m is odd, s := h_v
15. else s := h_v/2
16. For k = 1 to s
17.   R ⋙ Δw_{v,k}
18.   C := C + R
19. }
```

Remark 3. *In the above algorithm, shifted values of A and B are stored in S_A and S_B, respectively. In lines 5 and 13, $R \in GF(2^m)$ contains $(y_{0,j}, y_{1,j}, \cdots, y_{m-1,j})$, i.e., $\sum_{i=0}^{m-1} y_{i,j}\beta^{2^i}$. Also, right cyclic shifts of R in lines 7 and 17 correspond to $(\sum_{i=0}^{m-1} y_{i,j}\beta^{2^i})^{2^{w_{j,k}}}$. After the final iteration, C is the normal basis representation of the required product $A \cdot B$. Since, for even values of m, $y_{i+v,v} = y_{i,v}$, $0 \leq i \leq v-1$, where $v = \frac{m}{2}$, hence one may slightly reduce the computational cost of lines 12 and 13 by noting that the $\frac{m}{2}$ bits of each of the upper halves of L_A, L_B, and R are the same as the $\frac{m}{2}$ bits of their respective lower halves.*

Example 3. Here, the multiplication of $A = (01110)$ and $B = (10101)$ of Example 2 is shown using Algorithm 2. Table 3 shows the contents of various variables of the algorithm as they are updated. The row with j being "-" is for the initialization step (i.e., line 1) of the algorithm.

In order to obtain the overall computation time for a $GF(2^m)$ multiplication using Algorithm 2, the coordinates of the field elements can be divided into $\lceil \frac{m}{\omega} \rceil$ units where ω corresponds to the data path size of the processor. We assume that the processor can perform bit-wise XOR and AND of two ω-bit operands using one single XOR and one single AND instruction, respectively. Also, when a programming language, such as C, is used, we assume that an i-fold, $1 \leq i < \omega$, left/right shift is emulated using a total of p instructions. The value of p can be 4 or so when simple logical instructions, such as AND, SHIFT, and OR, are used.

Theorem 3. *The dynamic instruction count for Algorithm 2 is given by*

2. For an algorithmic description, the reader may refer to [14].

TABLE 3
Contents of Variables in Algorithm 2 for Multiplication of $A = (01110)$ and $B = (10101)$

j	S_A	S_B	L_A	L_B	k	$\Delta w_{j,k}$	R	C
-	01110	10101	-	-	-	-	-	00100
1	11100	01011	10010	11110			10010	
					1	1	01001	01101
					2	1	10100	11001
					3	1	01010	10011
2	11001	10110	10111	00011			00011	
					1	0	00011	10000
					2	1	10001	00001
					3	1	11000	11001
					4	2	00110	11111

$$\#Instructions \approx \left((p+1)\frac{C_N - 1}{2} + (2p+3)v + 1\right)\left\lceil\frac{m}{\omega}\right\rceil,$$

where C_N, v, p, and w are as defined earlier.

Proof. Initialization of C in line 1 needs $\lceil\frac{m}{w}\rceil$ instructions. Lines 3, 4, and 5 are repeated $v-1$ times and require $(v-1)(2p+3)\lceil\frac{m}{w}\rceil$ instructions. Lines 7 and 8 are inside a two-level nested loop and require $(\sum_{j=1}^{v-1} h_j)(p+1)\lceil\frac{m}{w}\rceil$ instructions. For lines 11 to 13, one needs $(2p+3)\lceil\frac{m}{w}\rceil$ instructions, whereas, for lines 17 and 18, $\epsilon h_v(p+1)\lceil\frac{m}{w}\rceil$ instructions are needed. Adding up all these instructions and assuming that overhead costs for the loops are small (alternatively, assuming unrolled loops), one completes the proof. □

Algorithm 2 and the NB multiplication algorithm of [14] have been implemented on an AMD Athlon XP 1500+ running at 1.33GHz with 480MB RAM. This implementation uses Visual C++ 6.0 and speed-optimized release build to obtain the timing data. For comparison purposes, we have used GF(2^{233}), which is one of the fields recommended by NIST and has a type-II ONB. Our results show that, for a GF(2^{233}) multiplication, Algorithm 2 and the multiplication algorithm of [14] require 22.53 μs and 607.8 μs, respectively.

Additional normal basis multiplication algorithms suitable for general purpose processors are the subject of discussions in another article by the authors [19].

4 Type-I Optimal Normal Basis Multiplication

An ONB-I is generated by the roots of an irreducible all-one polynomial (AOP), i.e.,

$$P(z) = z^m + z^{m-1} + \cdots + z + 1. \tag{33}$$

The AOP is irreducible if $m+1$ is prime and 2 is primitive modulo $m+1$ [24]. Thus, the roots of (33), i.e., β^{2^j}, $j = 0, 1, \cdots m-1$, form an ONB-I if and only if $m+1$ is prime and 2 is primitive in modulo $m+1$.

Lemma 6 [20].

$$\delta_j = \begin{cases} \beta^{2^{k_j}} & j = 1, 2, \cdots, \frac{m}{2} - 1 \\ 1 = \sum_{i=0}^{m-1} \beta^{2^i} & j = \frac{m}{2}, \end{cases} \tag{34}$$

where k_j is obtained from

$$2^j + 1 \equiv 2^{k_j} \mod (m+1). \tag{35}$$

Substituting (34) into (18), the product C can be written as

$$C = \left(\sum_{i=0}^{m-1} a_i b_i \beta^{2^i}\right) + \sum_{j=1}^{v-1}\left(\sum_{i=0}^{m-1} y_{i,j}\beta^{2^i}\right)^{2^{k_j}} + \left(\sum_{i=0}^{v-1} y_{i,v}\right), \tag{36}$$

where the right most summation results in 0 or 1 and, in the normal basis representation, 0 and 1 correspond to $(0, 0, \cdots, 0)$ and $(1, 1, \cdots, 1)$, respectively. Based on (36), now we can state an algorithm for ONB-I multiplication as follows:

Algorithm 3. (Low Complexity ONB-I Multiplication over $GF(2^m)$)
Input: $A, B \in GF(2^m)$, k_j, $1 \le j < v$, $v = \frac{m}{2}$
Output: $C = AB$
1. Generate $y_{i,j} = (a_i + a_{((i+j))})(b_i + b_{((i+j))})$, $1 \le j < v$, $0 \le i \le m-1$,
2. Generate $y_{i,v} = (a_i + a_{((v+i))})(b_i + b_{((v+i))})$, $0 \le i \le v-1$,
3. Initialize $c_i := a_i b_i$, $0 \le i \le m-1$, $f := y_{0,v}$, $f \in GF(2)$
4. For $j = 1$ to $v-1$ {
5. $r_i := y_{i,j}, 0 \le i \le m-1$, $R = (r_0, r_1, \cdots, r_{m-1})$
6. $R := R^{2^{k_j}}$
7. $C := C + R$
8. $f := f + y_{j,v}$
9. }
10. If f is 1, $C := C + (1, 1, \cdots, 1, 1)$
11. }

The above algorithm is hereafter referred to as LCONB-I.

Remark 4. *In line 6 of the LCONB-I algorithm, the operation $R^{2^{k_j}}$ can be accomplished by a k_j-fold cyclic shift. The number of bit level operations of lines 1, 2, and 8 are $2m(v-1)$, $2v$, and $v-1$, respectively. Also, lines 7 and 10 need $m(v-1)$ and m additions. Thus, the total number of additions is*

$$\#Add_{LCONB-I} = 1.5m^2 - 0.5m - 1 \tag{37}$$

and the number of multiplications is the same as that of the LCNB algorithm given in (25).

For comparison, we consider four other ONB-I multipliers as shown in Table 4. This table shows the number of bit operations of these multipliers and the time complexity of multipliers in bit-parallel implementation. The multiplier of [25] is considered to be the first such work published in the open literature and those of [6], [7], [20] are more recent work and have the best results among the known existing ones. As can be seen in this table, although the total number of operations of the proposed LCONB-I algorithm is the same as those of the three best multiplication schemes, the LCONB-I algorithm requires the least number of bit level

TABLE 4
Comparison of Bit Level Operations of ONB-I-Based Multiplication Schemes

Multipliers	#Mult	#Add	Total operations	Time complexity
MO [25]	m^2	$2m^2-2m$	$3m^2-2m$	$T_A+(1+\lceil\log_2 m\rceil)T_X$
Hasan *et al.* [6]	m^2	m^2-1	$2m^2-1$	$T_A+(1+\lceil\log_2 m\rceil)T_X$
Koc and Sunar [7]	m^2	m^2-1	$2m^2-1$	$T_A+(2+\lceil\log_2 m\rceil)T_X$
RR_MO [20]	m^2	m^2-1	$2m^2-1$	$T_A+(1+\lceil\log_2 m\rceil)T_X$
LCONB-I	$\frac{m(m+1)}{2}$	$1.5m^2-0.5m-1$	$2m^2-1$	$T_A+(1+\lceil\log_2 m\rceil)T_X$

multiplications, which can be advantageous in composite finite fields, as discussed in the next section.

Remark 5. *ONB-I can be treated as a polynomial basis after some permutations and then various methods can be applied for field multiplication [7], [8]. One of the methods is the Karatsuba-Ofman algorithm. In the asymptotic sense, the Karatsuba-Ofman algorithm has fewer bit level operations compared to the previously reported algorithms. However, for this special case of ONB-I, the value of m is composite. Using the algorithms presented in Section 5 of this paper, one can obtain an implementation for certain values of m, which has fewer number of bit level operations than the Karatsuba-Ofman algorithm based multiplier [17].*

5 Composite Field Multiplication

In this section, we consider multiplications in the finite field GF(2^m), where m is a composite number. These fields are referred to as composite fields and have been used in the recent past to develop efficient multiplication schemes [16], [15]. If such a field is to be used for cryptographic applications, special care needs to be taken in choosing the composite value for m. In order to avoid the recent Weil descent attack on elliptic curve cryptosystems [4], [22], the reader is referred to references [11] and [2] for "good" and "bad" composite values of m.

5.1 Algorithm Formulation

Theorem 4 [21]. *Let $m_1>1$, $m_2>1$ be relatively prime. Let $N_1=\{\beta_1^{2^i}\mid 0\le i\le m_1-1\}$ and $N_2=\{\beta_2^{2^j}\mid 0\le j\le m_2-1\}$ be normal bases for $GF(2^{m_1})$ and $GF(2^{m_2})$, respectively. Then, $N=\{\beta_1^{2^i}\beta_2^{2^j}\mid 0\le i\le m_1-1,\ 0\le j\le m_2-1\}$, is a normal basis for $GF(2^{m_1m_2})$ over $GF(2)$. The complexity of N is $C_N=C_{N_1}C_{N_2}$, where C_{N_1} and C_{N_2} are the complexities of N_1 and N_2, respectively.*

Assume that $m=m_1\cdot m_2$, where m_1 and m_2 are as defined above. Let $A\in GF((2^{m_2})^{m_1})$, then A can be represented w.r.t. the basis

$$N=\{\beta^{2^j}\mid 0\le j\le m-1\},\ \beta=\beta_1\beta_2,$$

as follows:

$$A=\sum_{j=0}^{m-1}a_j\beta^{2^j}=\sum_{j=0}^{m_1m_2-1}a_j\beta_1^{2^{j \bmod m_1}}\beta_2^{2^{j \bmod m_2}}=\sum_{i=0}^{m_1-1}A_i\beta_1^{2^i}, \quad (38)$$

where a_js are coordinates of A w.r.t. basis N and

$$A_i=\sum_{l=0}^{m_2-1}a_{i+l\cdot m_1}\beta_2^{2^{i+l\cdot m_1 \bmod m_2}}. \quad (39)$$

We assume this kind of representation for any two elements: A and $B\in GF((2^{m_2})^{m_1})$, i.e., $A=\sum_{i=0}^{m_1-1}A_i\beta_1^{2^i}$, $B=\sum_{i=0}^{m_1-1}B_i\beta_1^{2^i}$, where $A_i,\ B_i\in GF(2^{m_2})$. Without loss of generality, then the product $C=AB$ can be obtained from Lemma 4 as:

$$C=\begin{cases}\sum_{i=0}^{m_1-1}A_iB_i\gamma_0^{2^{i-1}}+\sum_{i=0}^{m_1-1}\sum_{j=1}^{v_1}Y_{i,j}\gamma_j^{2^i}, & \text{for } m_1 \text{ odd}\\ \sum_{i=0}^{m_1-1}A_iB_i\gamma_0^{2^{i-1}}+\sum_{i=0}^{m_1-1}\sum_{j=1}^{v_1-1}Y_{i,j}\gamma_j^{2^i}+\sum_{i=0}^{v_1-1}Y_{i,v_1}\gamma_{v_1}^{2^i}, & \text{for } m_1 \text{ even,}\end{cases} \quad (40)$$

where $v_1=\lfloor\frac{m_1}{2}\rfloor$, $\gamma_j=\beta_1^{1+2^j}$, $0\le j\le v_1$, and

$$\begin{aligned}&Y_{i,j}\triangleq(A_i+A_{((i+j))})(B_i+B_{((i+j))}),\\ &1\le j\le v_1,\ 0\le i\le m_1-1.\end{aligned} \quad (41)$$

In (41), $((i+j))=i+j \mod m_1$ and the underlying field operations are performed over the subfield $GF(2^{m_2})$.

Also, using (20), one can write γ_j w.r.t. N_1 as

$$\gamma_j=\sum_{k=1}^{h_j^{(1)}}\beta_1^{2^{w_{j,k}^{(1)}}},\ 1\le j\le v_1, \quad (42)$$

and, similarly to (22), the product C can also be obtained as

$$C=\begin{cases}\sum_{i=0}^{m_1-1}A_iB_i\beta_1^{2^i}+\sum_{j=1}^{v_1}\sum_{k=1}^{h_j^{(1)}}\left(\sum_{i=0}^{m_1-1}Y_{((i-w_{j,k}^{(1)})),j}\beta_1^{2^i}\right), & \text{for } m_1 \text{ odd}\\ \sum_{i=0}^{m_1-1}A_iB_i\beta_1^{2^i} & \\ +\sum_{j=1}^{v_1-1}\sum_{k=1}^{h_j^{(1)}}\left(\sum_{i=0}^{m_1-1}Y_{(i-w_{j,k}^{(1)})),j}\beta_1^{2^i}\right)+D, & \text{for } m_1 \text{ even,}\end{cases} \quad (43)$$

where

$$D=\sum_{k=1}^{\frac{h_{v_1}^{(1)}}{2}}\sum_{i=0}^{v_1-1}Y_{((i-w_{v_1,k}^{(1)})),v_1}(\beta_1^{2^i}+\beta_1^{2^{i+v_1}}),\ v_1=\frac{m_1}{2}.$$

Based on (43), we can state the following algorithm for multiplication in GF(2^m), where $m = m_1 \cdot m_2$.

Algorithm 4. (Composite Field Normal Basis Multiplication)
Input: $A,\ B \in GF((2^{m_2})^{m_1})$, $\gamma_j \in GF(2^{m_1})$, $1 \le j \le v_1$
Output: $C = AB$

1. $A_i[l'] := A[i + m_1 l]$, $B_i[l'] := B[i + m_1 l]$, $0 \le l \le m_2 - 1$, $0 \le i \le m_1 - 1$, where $l' = i + m_1 l \bmod m_1$
2. Generate $Y_{i,j} := (A_i + A_{((i+j))})(B_i + B_{((i+j))})$, $1 \le j < v_1$, $0 \le i \le m_1 - 1$, where $Y_{i,j},\ A_i,\ B_i \in GF(2^{m_2})$.
3. Initialize $C_i := A_i B_i$, $0 \le i \le m_1 - 1$, $\widetilde{C} := C_0 || C_1 || \cdots || C_{m_1 - 1}$
4. For $j = 1$ to $v_1 - 1$ {
5.   For $k = 1$ to $h_j^{(1)}$ {
6. $R_i := Y_{((i - w_{j,k}^{(1)})),j}$, $0 \le i \le m_1 - 1$, $\widetilde{R} := R_0 || R_1 || \cdots || R_{m_1 - 1}$
7. $\widetilde{C} := \widetilde{C} + \widetilde{R}$
8.   }
9. }
10. If m_1 is odd,
11. $s := h_{v_1}^{(1)}$, $t := m_1$
12. else $s := \frac{h_{v_1}^{(1)}}{2}$, $t := \frac{m_1}{2}$
13. Generate $Y_{i,v} = (A_i + A_{((v_1 + i))})(B_i + B_{((v_1 + i))})$, $0 \le i \le t - 1$,
14. If m_1 is even $Y_{i+v_1, v_1} = Y_{i,v_1}$, $0 \le i \le \frac{m_1}{2} - 1$
15. For $k = 1$ to s {
16. $R_i := Y_{((i - w_{v_1,k}^{(1)})),v_1}$, $0 \le i \le t - 1$
17.   If m_1 is even,
18. $R_{i + \frac{m_1}{2}} := R_i$, $0 \le i \le \frac{m_1}{2} - 1$, $\widetilde{R} := R_0 || \cdots || R_{\frac{m_1}{2} - 1} || R_0 || \cdots || R_{\frac{m_1}{2} - 1}$
19. $\widetilde{C} := \widetilde{C} + \widetilde{R}$
20. }
21. $C[i + m_1 l] := C_i[l']$, $0 \le l \le m_2 - 1$, $0 \le i \le m_1 - 1$.

Example 4. Let $m = 33$, $m_1 = 3$, and $m_2 = 11$. As per Table 3 of [13], there are ONBs for $GF(2^3)$ and $GF(2^{11})$. Thus, $N_1 = \{\beta_1^{2^i} \mid 0 \le i \le 2\}$ and $N_2 = \{\beta_2^{2^l} \mid 0 \le l \le 10\}$ are type-II optimal normal bases of $GF(2^3)$ and $GF(2^{11})$, respectively. Using Theorem 4, $N = \{\beta^{2^j} \mid 0 \le j \le 32\}$, where $\beta = \beta_1 \beta_2$ is a normal basis of $GF(2^{33})$ over $GF(2)$. The complexity of N is $C_N = C_{N_1} C_{N_2} = (2 \cdot 3 - 1)(2 \cdot 11 - 1) = 105$. Any two field elements $A, B \in GF(2^{33})$ can be written w.r.t. N as

$$A = \sum_{j=0}^{32} a_j \beta^{2^j} = A_0 \beta_1 + A_1 \beta_1^2 + A_2 \beta_1^4$$

$$B = \sum_{j=0}^{32} b_j \beta^{2^j} = B_0 \beta_1 + B_1 \beta_1^2 + B_2 \beta_1^4,$$

where $A_i = \sum_{l=0}^{10} a_{i+3l} \beta_2^{2^{l'}}$, $B_i = \sum_{l=0}^{10} b_{i+3l} \beta_2^{2^{l'}}$, $0 \le j \le 2$, and $l' = i + 3l \mod 11$. Let $C = C_0 \beta_1 + C_1 \beta_1^2 + C_2 \beta_1^4$ be the product of A and B. Thus, using (40), we have

$$\begin{aligned} C \ = \ & A_0 B_0 \beta_1 & + (A_0 + A_1)(B_0 + B_1)\beta_1^3 \\ & + A_1 B_1 \beta_1^2 & + (A_1 + A_2)(B_1 + B_2)\beta_1^6 \\ & + A_2 B_2 \beta_1^4 & + (A_2 + A_0)(B_2 + B_0)\beta_1^{12}. \end{aligned}$$

Using Table 2 in [9], for the type-II ONB over $GF(2^3)$, we have $\beta_1^3 = \beta_1 + \beta_1^2$. Thus,

$$\begin{aligned} C = \ & ((A_0 B_0 + (A_0 + A_1)(B_0 + B_1) + (A_2 + A_0)(B_2 + B_0))\beta_1 \\ & + ((A_1 B_1 + (A_1 + A_2)(B_1 + B_2) + (A_0 + A_1)(B_0 + B_1))\beta_1^2 \\ & + ((A_2 B_2 + (A_2 + A_0)(B_2 + B_0) + (A_1 + A_2)(B_1 + B_2))\beta_1^4. \end{aligned} \tag{44}$$

From (44), we see that six multiplications and 12 additions over subfield $GF(2^{m_2})$ are needed to generate C_0, C_1, and C_2. Thus, the total numbers of bit level multiplications and additions are 396 and 1,452, respectively.

5.2 Complexity and Comparison

In Algorithm 4, $\widetilde{C}$ in line 3 is obtained by concatenating C_js. $\widetilde{R}$ in line 6 is obtained in a similar way. The total number of operations of the composite field NB (CFNB) multiplication algorithm consists of two parts: multiplications and additions over the subfield $GF(2^{m_2})$. Using Theorem 2, the numbers of multiplications and additions over $GF(2^{m_2})$ are $\frac{m_1(m_1+1)}{2}$ and $\frac{m_1}{2}(C_{N_1} + 2m_1 - 3)$,[3] respectively. Each $GF(2^{m_2})$ addition can be performed by m_2 bit level (i.e., GF(2)) additions. If we use Algorithm 1 for subfield operations, then, at the bit level, each $GF(2^{m_2})$ multiplication requires $\frac{m_2(m_2+1)}{2}$ multiplications and $\frac{m_2}{2}(C_{N_2} + 2m_2 - 3)$ additions. Thus, the total numbers of bit level operations are as follows:

$$\#\mathrm{Mult}_{\mathrm{CFNB}} = \frac{m(m_1 + 1)(m_2 + 1)}{4}, \tag{45}$$

and

$$\begin{aligned} \#\mathrm{Add}_{\mathrm{CFNB}} &= \frac{m_1}{2}(C_{N_1} + 2m_1 - 3) \cdot m_2 \\ &\quad + \frac{m_1(m_1 + 1)}{2} \cdot \frac{m_2}{2}(C_{N_2} + 2m_2 - 3) \\ &= \frac{m}{2}\left[C_{N_1} + 2m_1 - 3 + \frac{m_1 + 1}{2}(C_{N_2} + 2m_2 - 3)\right]. \end{aligned} \tag{46}$$

Thus, for a given m, we can use $m_1 < m_2$ to reduce the number of addition operations given in (46). Additionally, if $m_2 + 1$ is prime and 2 is primitive modulo $m_2 + 1$, then there exists an ONB-I over $GF(2^{m_2})$ and Algorithm 3 can be used for $GF(2^{m_2})$ multiplication. Thus, using (37), the number of additions as given in (46) can be reduced to $\frac{m_1}{2}(C_{N_1} + 2m_1 - 3)m_2 + \frac{m_1(m_1+1)}{2}(1.5m_2^2 - 0.5m_2 - 1)$.

In order to obtain the time complexity of the composite field NB multiplication of GF($(2^{m_2})^{m_1}$) over GF(2^{m_2}) in bit-parallel implementation, one can easily replace the time delay of AND gate with the time delay of subfield multiplication of GF(2^{m_2}) over GF(2) into (28). Thus, the time delay of the CFNB multiplier is $T_A + (\lceil \log_2 C_{N_1} \rceil + \lceil \log_2 C_{N_2} \rceil) T_X$.

Table 5 compares bit level operations for multiplication over GF(2^{33}) for a number of algorithms. Rows 2, 3, and 4,

3. For the sake of simplicity, we have not used the symmetrical property for m even.

TABLE 5
Comparison of Operations for Normal Basis Multipliers over $GF(2^{33})$

Multipliers	C_N	#Mult	#Add	Total bit operations
MO [25]	65	1089	2112	3201
RR_MO [20]	65	1089	1584	2673
LCNB	65	561	2112	2673
MO [25]	105	1089	3432	4521
RR_MO [20]	105	1089	2244	3333
LCNB	105	561	2772	3333
CFNB	105	396	1452	1848

where $C_N = 65$, use ONB-II which exists for $\mathrm{GF}(2^{33})$ over GF(2). On the other hand, rows 5, 6, and 7, where $C_N = C_{N_1} \cdot C_{N_2} = 105$, use the two ONB-IIs which exist for the subfields $\mathrm{GF}(2^3)$ and $\mathrm{GF}(2^{11})$ as discussed in the above example. This comparison shows that the proposed CFNB multiplier has the least number of bit level operations. More interestingly, for composite values of m, the well-known optimal normal bases $\mathrm{GF}(2^m)$ over GF(2) do not seem to be the best choice when one considers bit level operations, which in turn determines the space complexity for hardware implementation of a normal basis multiplier.

In [15], two normal basis multipliers in the composite field $\mathrm{GF}((2^{m_2})^{m_1})$ over $\mathrm{GF}(2^{m_2})$ are proposed. The structures are only applicable to special cases of $m = m_1 m_2$, $\gcd(m_1, m_2) = 1$, where there exists an ONB-I for the subfield and ONB-II for the extension field or vice versa. In both structures, the number of subfield multiplications required is m_1^2, which is about twice of what has been proposed here, i.e., $\frac{m_1(m_1+1)}{2}$.

We wind up this section by stating the following theorem which gives the bit level operations for normal basis multiplication over generalized composite fields.

Theorem 5. *Let* $m = \prod_{i=1}^{n} m_i$, $1 < m_1 < m_2 < \cdots < m_n$, *where* $\gcd(m_i, m_j) = 1$, $i \neq j$. *Then, for a normal basis multiplication over the composite field* $\mathrm{GF}(2^m)$, *the numbers of bit level multiplications and additions are*

$$\#Mult_{CFNB} = \frac{m}{2^n} \prod_{i=1}^{n} (m_i + 1) \tag{47}$$

and

$$\#Add_{CFNB} = \frac{m}{2}\left(C_{N_1} + 2m_1 - 3 + \sum_{j=1}^{n-1} \frac{C_{N_{j+1}} + 2m_{j+1} - 3}{2^j} \prod_{i=1}^{j} (m_i + 1)\right), \tag{48}$$

respectively.

6 Concluding Remarks

In this article, efficient algorithms for normal basis multiplication over $\mathrm{GF}(2^m)$ have been proposed. These algorithms are suitable for implementation of cryptographic functions both in hardware and software. It has been shown that, when m is composite, the proposed CFNB algorithm requires significantly fewer numbers of bit level operations compared to other similar algorithms available in the open literature. More interestingly, it has been shown that, for composite values of m, the well-known optimal normal bases $\mathrm{GF}(2^m)$ over GF(2) do not seem to be the best choice when one considers bit level operations, which in turn determines the space complexity for hardware implementation of a normal basis multiplier.

There are a number of possibilities for construction of the composite field NB multipliers in hardware implementation. These depend on which architecture is chosen for subfield implementation. Investigation is being carried out to obtain the best composite field multiplier such that the complexities of the multiplier architecture is minimum for any given composite $m \in [160, 600]$.

Acknowledgments

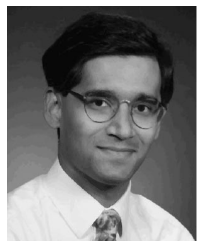
The authors would like to thank the anonymous reviewers for their comments. The authors would also like to thank Z. Zhang for his help with implementing the algorithms and getting their timing results. The work was supported in part by an NSERC Research grant to M.A. Hasan. A preliminary version of this article was presented at the First International Conference in Cryptology in India, Calcutta, India, December 2000 [18].

References

[1] G.B. Agnew, R.C. Mullin, I.M. Onyszchuk, and S.A. Vanstone, "An Implementation for a Fast Public-Key Cryptosystem," *J. Cryptology*, vol. 3, pp. 63-79, 1991.
[2] M. Ciet, J.-J. Quisquater, and F. Sica, "A Secure Family of Composite Finite Fields Suitable for Fast Implementation of Elliptic Curve Cryptography," *Proc. Indocrypt 2001*, pp. 108-116, Dec. 2001.
[3] M. Elia and M. Leone, "On the Inherent Space Complexity of Fast Parallel Multipliers for $GF(2^m)$," *IEEE Trans. Computers*, vol. 51, no. 3, pp. 346-351, Mar. 2002.
[4] S.D. Galbraith and N. Smart, "A Cryptographic Application of Weil Descent," *Proc. Seventh IMA Conf. Cryptography and Coding*, pp. 191-200, 1999.
[5] S. Gao and H.W. Lenstra Jr., "Optimal Normal Bases," *Designs, Codes, and Cryptography*, vol. 2, pp. 315-323, 1992.
[6] M.A. Hasan, M.Z. Wang, and V.K. Bhargava, "A Modified Massey-Omura Parallel Multiplier for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 42, no. 10, pp. 1278-1280, Oct. 1993.
[7] C.K. Koc and B. Sunar, "Low-Complexity Bit-Parallel Canonical and Normal Basis Multipliers for a Class of Finite Fields," *IEEE Trans. Computers*, vol. 47, no. 3, pp. 353-356, Mar. 1998.
[8] M. Leone, "A New Low Complexity Parallel Multiplier for a Class of Finite Fields," *Proc. Cryptographic Hardware and Embedded Systems (CHES 2001)*, pp. 160-170, 2001.
[9] C.-C. Lu, "A Search of Minimal Key Functions for Normal Basis Multipliers," *IEEE Trans. Computers*, vol. 46, no. 5, pp. 588-592, May 1997.
[10] J.L. Massey and J.K. Omura, "Computational Method and Apparatus for Finite Field Arithmetic," US Patent No. 4,587,627, 1986.
[11] M. Maurer, A. Menezes, and E. Teske, "Analysis of the GHS Weil Descent Attack on the ECDLP over Characteristic Two Finite Fields of Composite Degree," *Proc. Indocrypt 2001*, pp. 195-213, Dec. 2001.
[12] A.J. Menezes, I.F. Blake, X. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*. Kluwer Academic, 1993.
[13] R.C. Mullin, I.M. Onyszchuk, S.A. Vanstone, and R.M. Wilson, "Optimal Normal Bases in $GF(p^n)$," *Discrete Applied Math.*, vol. 22, pp. 149-161, 1988/89.

[14] Nat'l Inst. of Standards and Technology, "Digital Signature Standard," FIPS Publication 186-2, Feb. 2000.
[15] S. Oh, C.H. Kim, J. Lim, and D.H. Cheon, "Efficient Normal Basis Multipliers in Composite Fields," *IEEE Trans. Computers,* vol. 49, no. 10, pp. 1133-1138, Oct. 2000.
[16] C. Paar, P. Fleishmann, and P. Soria-Rodriguez, "Fast Arithmetic for Public-Key Algorithms in Galois Fields with Composite Exponents," *IEEE Trans. Computers,* vol. 48, no. 10, pp. 1025-1034, Oct. 1999.
[17] A. Reyhani-Masoleh, "Low Complexity and Fault Tolerant Arithmetic in Binary Extended Finite Fields," PhD thesis, Dept. of Electrical and Computer Eng., Univ. of Waterloo, Canada, May 2001.
[18] A. Reyhani-Masoleh and M.A. Hasan, "On Efficient Normal Basis Multiplication," *Proc. Indocrypt 2000,* pp. 213-224, Dec. 2000.
[19] A. Reyhani-Masoleh and M.A. Hasan, "Fast Normal Basis Multiplication Using General Purpose Processors," *Proc. Selected Areas in Cryptography (SAC 2001),* pp. 230-244, Aug. 2001.
[20] A. Reyhani-Masoleh and M.A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over $GF(2^m)$," *IEEE Trans. Computers,* vol. 51, no. 5, pp. 511-520, May 2002.
[21] J.E. Seguin, "Low Complexity Normal Bases," *Discrete Applied Math.,* vol. 28, pp. 309-312, 1990.
[22] N.P. Smart, "How Secure Are Elliptic Curves over Composite Extension Fields?" *Proc. Eurocrypt 2001,* pp. 30-39, 2001.
[23] B. Sunar and C.K. Koc, "An Efficient Optimal Normal Basis Type II Multiplier," *IEEE Trans. Computers,* vol. 50, no. 1, pp. 83-88, Jan. 2001.
[24] P.K.S. Wah and M.Z. Wang, "Realization and Application of the Massey-Omura Lock," *Proc. IEEE Int'l Zurich Seminar Digital Comm.,* pp. 175-182, 1984.
[25] C.C. Wang, T.K. Truong, H.M. Shao, L.J. Deutsch, J.K. Omura, and I.S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in $GF(2^m)$," *IEEE Trans. Computers,* vol. 34, no. 8, pp. 709-716, Aug. 1985.

Arash Reyhani-Masoleh received the BSc degree from Iran University of Science and Technology in 1989, the MSc degree from the University of Tehran in 1991, both with the first rank in electrical and electronic engineering, and the PhD degree in electrical and computer engineering from the University of Waterloo, Canada, in 2001. From 1991 to 1997, he was with the Department of Electrical Engineering, Iran University of Science and Technology. Since June 2001, he has been a postdoctoral fellow with the Centre for Applied Cryptographic Research, University of Waterloo. His current research interests include algorithms and VLSI architectures for computations in finite fields, fault-tolerant computing, and error-control coding. Dr. Reyhani-Masoleh is a recipient of two Ontario Graduate Scholarships and three Faculty of Engineering awards at the University of Waterloo. He was awarded an NSERC (Natural Sciences and Engineering Research Council of Canada) postdoctoral fellowship in 2002. He is a member of the IEEE and the IEEE Computer Society.



M. Anwar Hasan received the BSc degree in electrical and electronic engineering, the MSc degree in computer engineering, both from the Bangladesh University of Engineering and Technology, in 1986 and 1988, respectively, and the PhD degree in electrical engineering from the University of Victoria, Canada, in 1992. Since 1993, he has been with the Department of Electrical and Computer Engineering, University of Waterloo, Canada, where he is now a professor. At the University of Waterloo, he is also a member of the Centre for Applied Cryptographic Research and the Center for Wireless Communications. His current research interests include algorithms and architectures for computations in Galois fields, data security and reliability, and digital communication networks. From January to December of 1999, he was on sabbatical with Motorola Labs., Schaumburg, Illinois. Dr. Hasan is a recipient of the Raihan Memorial Gold Medal. At the University of Victoria, he was awarded the President's Research Scholarship four times. He served on the program and executive committees of several conferences and, currently, he is an associate editor of the *IEEE Transactions of Computers*. He is a senior member of the IEEE, a member of the IEEE Computer Society, and a licensed professional engineer of Ontario.