Efficient Linear Array for Multiplication in $GF(2^m)$ Using a Normal Basis for Elliptic Curve Cryptography

Soonhak Kwon[1], Kris Gaj[2], Chang Hoon Kim[3], and Chun Pyo Hong[3]

[1] Inst. of Basic Science and Dept. of Mathematics, Sungkyunkwan University, Suwon 440-746, Korea
shkwon@math.skku.ac.kr
[2] Dept. of Electrical and Computer Engineering, George Mason University, University Drive, Fairfax, VA 22030, USA
kgaj@gmu.edu
[3] Dept. of Computer and Information Engineering, Daegu University, Kyungsan 712-714, Korea
chkim@dsp.taegu.ac.kr,cphong@daegu.ac.kr

Abstract. We present a new sequential normal basis multiplier over $GF(2^m)$. The gate complexity of our multiplier is significantly reduced from that of Agnew et al. and is comparable to that of Reyhani-Masoleh and Hasan, which is the lowest complexity normal basis multiplier of the same kinds. On the other hand, the critical path delay of our multiplier is same to that of Agnew et al. Therefore it is supposed to have a shorter or the same critical path delay to that of Reyhani-Masoleh and Hasan. Moreover our method of using a Gaussian normal basis makes it easy to find a basic multiplication table of normal elements. So one can easily construct a circuit array for large finite fields, $GF(2^m)$ where $m = 163, 233, 283, 409, 571$, i.e. the five recommended fields by NIST for elliptic curve cryptography.

Keywords: Massey-Omura multiplier, Gaussian normal basis, finite field, elliptic curve cryptography, critical path delay.

1 Introduction

Finite field multiplication finds various applications in many cryptographic areas such as ECC and AES. Though one may design a finite field multiplier in a software implementation, a hardware arrangement has a strong advantage when one wants a high speed multiplier. Moreover arithmetic of $GF(2^m)$ is easily realized in a circuit design using a few logical gates. A good multiplication algorithm depends on the choice of a basis for a given finite field. Especially a normal basis is widely used [5,10,11] because it has some good properties such as simple squaring. A multiplication in $GF(2^m)$ can be classified into two types, a parallel (two dimensional) [4,5,8,10] and a sequential (linear) [1,3,9,11] architectures.

Though a parallel multiplier is well suited for high speed applications, ECC requires large m for $GF(2^m)$ (at least $m = 163$) to support a sufficient security. In other words, since the parallel architecture has an area complexity of $O(m^2)$, it is not suited for this application. On the other hand, a sequential multiplier has an area complexity of $O(m)$ and therefore is applicable for ECC. Since it takes m clock cycles to produce one multiplication result using a sequential multiplier, it is slower than a parallel multiplier. Consequently reducing the total delay time of a sequential multiplier is very important.

A normal basis multiplier of Massey and Omura [7] has a parallel-in, serial-out structure and has a quite long critical path delay proportional to $\log_2 m$. Agnew et al. [1] proposed a sequential multiplier which has a parallel-in, parallel-out structure. It is based on the multiplication algorithm of Massey and Omura, however the critical path delay of the multiplier of Agnew et al. is significantly reduced from that of Massey and Omura. Recently, Reyhani-Masoleh and Hasan [3] presented two sequential multipliers using a symmetric property of multiplication of normal elements. Both multipliers in [3] have roughly the same area complexity and critical path delay. These multipliers have the reduced area complexity from that of Agnew et al. with a slightly increased critical path delay. In fact, the exact critical path delay of the multipliers of Reyhani-Masoleh and Hasan is difficult to estimate in terms of m and is generally believed to be slightly longer or equal to that of Agnew et al. For example, for the case of a type II ONB, the critical path delay of Reyhani-Masoleh and Hasan [3] is $T_A + 3T_X$ while that of Agnew et al. [1] is $T_A + 2T_X$, where T_A, T_X are the delay time of a two input AND gate and a two input XOR gate, respectively. However since we are dealing with a sequential (linear) multiplier, even a small increment of critical path delay such as T_X results in a total delay of mT_X where m is the size of a field.

Our aim in this paper is to present a sequential multiplier using a Gaussian normal basis in $GF(2^m)$ for odd m. Since choosing an odd m is a necessary condition for cryptographic purposes and since a low complexity normal basis is frequently a Gaussian normal basis of type (m, k) for low k, our restriction in this paper does not cause any serious problem for practical purposes. In fact all the five recommended fields $GF(2^m)$ by NIST [16] for ECC where $m = 163, 233, 283, 409, 571$ can be dealt using our Gaussian normal basis, and the corresponding circuits are easy to construct if one follows a simple arithmetic rule of a Gaussian normal basis. We will show that the area complexity of our sequential multiplier is reduced from that of the multiplier of Agnew et al. [1] and thus comparable to that of the multiplier of Reyhani-Masoleh and Hasan [3]. Moreover the critical path delay of our multiplier is same to that of Agnew et al. and therefore is believed to be shorter or equal to that of Reyhani-Masoleh and Hasan.

2 Review of the Multipliers of Agnew et al. and Reyhani-Masoleh and Hasan

Let $GF(2^m)$ be a finite field with characteristic two. $GF(2^m)$ is a vector space of dimension m over $GF(2)$. A basis of the form $\{\alpha, \alpha^2, \alpha^{2^2}, \cdots, \alpha^{2^{m-1}}\}$ is called a normal basis for $GF(2^m)$. It is well known [6] that a normal basis exists for all $m \geq 1$. Let $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ be a normal basis in $GF(2^m)$ with $\alpha_i = \alpha^{2^i}$. Let

$$\alpha_i \alpha_j = \sum_{s=0}^{m-1} \lambda_{ij}^{(s)} \alpha_s, \tag{1}$$

where $\lambda_{ij}^{(s)}$ is in $GF(2)$. Then for any integer t, we have

$$\alpha_i \alpha_j = (\alpha_{i-t}\alpha_{j-t})^{2^t} = \sum_{s=0}^{m-1} \lambda_{i-t,j-t}^{(s)} \alpha_{s+t} = \sum_{s=0}^{m-1} \lambda_{i-t,j-t}^{(s-t)} \alpha_s, \tag{2}$$

where the subscripts and superscripts of λ are reduced $\pmod{m}$. Therefore comparing the coefficients of α_s, we find

$$\lambda_{ij}^{(s)} = \lambda_{i-t,j-t}^{(s-t)}. \tag{3}$$

In particular, we have

$$\lambda_{ij}^{(s)} = \lambda_{i-s,j-s}^{(0)}. \tag{4}$$

Letting $A = \sum_{i=0}^{m-1} a_i \alpha_i$ and $B = \sum_{j=0}^{m-1} b_j \alpha_j$ in $GF(2^m)$, we have the multiplication $C = AB = \sum_{s=0}^{m-1} c_s \alpha_s$ where

$$C = \sum_{i,j} a_i b_j \alpha_i \alpha_j = \sum_{i,j} a_i b_j \sum_{s=0}^{m-1} \lambda_{ij}^{(s)} \alpha_s = \sum_{s=0}^{m-1} (\sum_{i,j} a_i b_j \lambda_{ij}^{(s)}) \alpha_s. \tag{5}$$

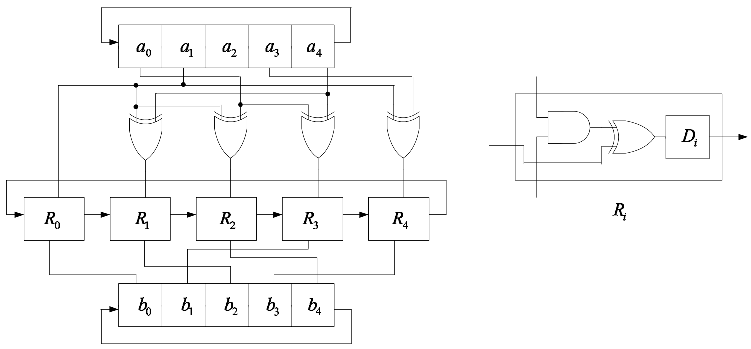

Fig. 1. A circuit of Agnew et al. in $GF(2^5)$

Therefore, using (4), we have the coefficients c_s of $C = AB$ as

$$c_s = \sum_{i,j} a_i b_j \lambda_{ij}^{(s)} = \sum_{i,j} a_i b_j \lambda_{i-s,j-s}^{(0)} = \sum_{i,j} a_{i+s} b_{j+s} \lambda_{ij}^{(0)}, \tag{6}$$

where the subscripts of a, b and λ are reduced $\pmod{m}$. The circuit of Agnew et al. [1] is a straightforward realization of the above equation with the information of the m by m matrix $(\lambda_{ij}^{(0)})$. When there is a type II ONB (optimal normal basis), it is easy to find $\lambda_{ij}^{(0)}$ as is explained in [1]. That is,

$$\lambda_{ij}^{(0)} = 1 \quad \text{iff} \quad 2^i \pm 2^j \equiv \pm 1 \pmod{2m+1}. \tag{7}$$

Figure 1 shows the circuit of Agnew et al. for the case $m = 5$ where a type II ONB is used. For arbitrary finite field, finding $\lambda_{ij}^{(0)}$ may not be so easy. However if we have a Gaussian normal basis, one can easily find $\lambda_{ij}^{(0)}$ by following a simple arithmetic rule. A Gaussian normal basis and a type II ONB will be discussed briefly in the following sections.

Recently, Reyhani-Masoleh and Hasan [3] suggested a new normal basis multiplication algorithm which significantly reduces the area complexity compared with the multiplier of Agnew et al. They used $\alpha\alpha_i$ instead of $\alpha_i\alpha_j$ and wisely utilized the symmetric property between $\alpha\alpha_i$ and $\alpha\alpha_{m-i}$. In fact they proposed two sequential multiplication architectures, so called XESMPO and AESMPO [3]. Since the hardware complexity of AESMPO is higher than that of XESMPO and both architectures have the same critical path delay, we will sketch the idea in [3,4] for the case of XESMPO. In [3,4], the multiplication $C = AB$ is expressed as

$$\begin{aligned} \sum_{i,j} a_i b_j \alpha_i \alpha_j &= \sum_{i=0}^{m-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{m-1} \sum_{j \neq i} a_i b_j (\alpha\alpha_{j-i})^{2^i} \\ &= \sum_{i=0}^{m-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{m-1} \sum_{j \neq 0} a_i b_{j+i} (\alpha\alpha_j)^{2^i}. \end{aligned} \tag{8}$$

When m is odd, the second term of the right side of the above equation is written as

$$\sum_{i=0}^{m-1} \sum_{j=1}^{\nu} a_i b_{j+i} (\alpha\alpha_j)^{2^i} + \sum_{i=0}^{m-1} \sum_{j=m-\nu}^{m-1} a_i b_{j+i} (\alpha\alpha_j)^{2^i}, \tag{9}$$

and when m is even, it is written as

$$\sum_{i=0}^{m-1} \sum_{j=1}^{\nu} a_i b_{j+i} (\alpha\alpha_j)^{2^i} + \sum_{i=0}^{m-1} \sum_{j=m-\nu}^{m-1} a_i b_{j+i} (\alpha\alpha_j)^{2^i} + \sum_{i=0}^{m-1} a_i b_{\nu+1+i} (\alpha\alpha_{\nu+1})^{2^i}, \tag{10}$$

where $\nu = \lfloor \frac{m-1}{2} \rfloor$, i.e. $m = 2\nu + 1$ or $m = 2\nu + 2$. Also the second term of (9) and (10) is written as

$$\begin{aligned} \sum_{i=0}^{m-1} \sum_{j=m-\nu}^{m-1} a_i b_{j+i} (\alpha\alpha_j)^{2^i} &= \sum_{i=0}^{m-1} \sum_{j=1}^{\nu} a_i b_{m-j+i} (\alpha\alpha_{m-j})^{2^i} \\ &= \sum_{i=0}^{m-1} \sum_{j=1}^{\nu} a_{i+j} b_i (\alpha\alpha_{m-j})^{2^{i+j}} \\ &= \sum_{i=0}^{m-1} \sum_{j=1}^{\nu} a_{i+j} b_i (\alpha\alpha_j)^{2^i}, \end{aligned} \tag{11}$$

where the first (resp. second) equality comes from the rearrangement of the summation with respect to j (resp. i) and all the subscripts are reduced to $\pmod m$. Therefore we have the basic multiplication formula of Reyhani-Masoleh and Hasan depending on whether m is *odd* or m is *even* as

$$AB = \sum_{i=0}^{m-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{m-1} \sum_{j=1}^{\nu} (a_i b_{j+i} + a_{j+i} b_i)(\alpha\alpha_j)^{2^i}, \tag{12}$$

or

$$AB = \sum_{i=0}^{m-1} a_i b_i \alpha_{i+1} + \sum_{i=0}^{m-1} \sum_{j=1}^{\nu} (a_i b_{j+i} + a_{j+i} b_i)(\alpha\alpha_j)^{2^i} + \sum_{i=0}^{m-1} a_i b_{\nu+1+i} (\alpha\alpha_{\nu+1})^{2^i}. \tag{13}$$

Using these formulas, they derived a sequential multiplier where the gate complexity is significantly reduced from that of [1]. The circuit of the multiplier is shown in Figure 2 for $m = 5$ where a type II ONB is used.

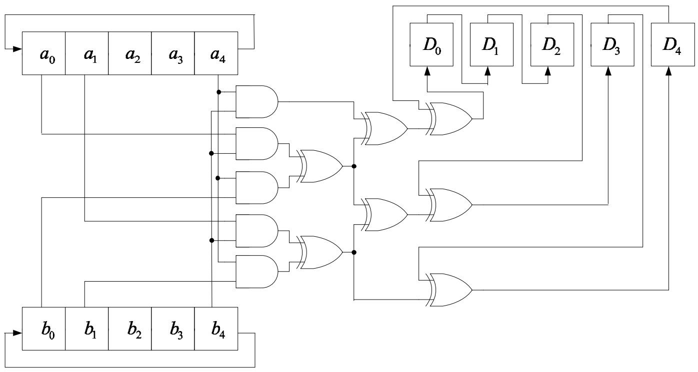

Fig. 2. A circuit of Reyhani-Masoleh and Hasan in $GF(2^5)$

3 Gaussian Normal Basis of Type k in $GF(2^m)$

We will briefly explain basic multiplication principle in $GF(2^m)$ with a Gaussian normal basis of type k over $GF(2)$ (See [6,12].). Let m, k be positive integers such that $p = mk+1$ is a prime $\neq 2$. Let $K = \langle \tau \rangle$ be a unique subgroup of order k in $GF(p)^\times$. Let β be a primitive pth root of unity in $GF(2^{mk})$. The following element

$$\alpha = \sum_{j=0}^{k-1} \beta^{\tau^j} \tag{14}$$

is called a Gauss period of type (m,k) over $GF(2)$. Let $ord_p 2$ be the order of $2 \pmod p$ and assume $gcd(mk/ord_p 2, m) = 1$. Then it is well known [6] that α is a normal element in $GF(2^m)$. That is, letting $\alpha_i = \alpha^{2^i}$ for $0 \le i \le m-1$, $\{\alpha_0, \alpha_1, \alpha_2, \cdots, \alpha_{m-1}\}$ is a basis for $GF(2^m)$ over $GF(2)$. It is called a Gaussian normal basis of type k or (m,k) in $GF(2^m)$. Since $K = \langle\tau\rangle$ is a subgroup of order k in the cyclic group $GF(p)^\times$, the quotient group $GF(p)^\times/K$ is also a cyclic group of order m and the generator of the group is $2K$. Therefore we have a coset decomposition of $GF(p)^\times$ as a disjoint union,

$$GF(p)^\times = K_0 \cup K_1 \cup K_2 \cdots \cup K_{m-1}, \tag{15}$$

where $K_i = 2^i K, 0 \le i \le m-1$, and an element in $GF(p)^\times$ is uniquely written as $\tau^s 2^t$ for some $0 \le s \le k-1$ and $0 \le t \le m-1$. For each $0 \le i \le m-1$, we have

$$\alpha\alpha_i = \sum_{s=0}^{k-1} \beta^{\tau^s} \sum_{t=0}^{k-1} \beta^{\tau^t 2^i} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^{t-s}2^i)} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t 2^i)}. \tag{16}$$

From (15), there are unique $0 \le u \le k-1$ and $0 \le v \le m-1$ such that $1+\tau^u 2^v = 0 \in GF(p)$. If $t \ne u$ or $i \ne v$, then we have $1+\tau^t 2^i \in K_{\sigma(t,i)}$ for some $0 \le \sigma(t,i) \le m-1$ depending on t and i. Thus we may write $1+\tau^t 2^i = \tau^{t'} 2^{\sigma(t,i)}$ for some t'. Now when $i \ne v$,

$$\begin{aligned}\alpha\alpha_i &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t 2^i)} = \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(\tau^{t'} 2^{\sigma(t,i)})} \\ &= \sum_{t=0}^{k-1}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} 2^{\sigma(t,i)}} = \sum_{t=0}^{k-1} \alpha^{2^{\sigma(t,i)}} = \sum_{t=0}^{k-1} \alpha_{\sigma(t,i)}.\end{aligned} \tag{17}$$

Also when $i = v$,

$$\begin{aligned}\alpha\alpha_v &= \sum_{s=0}^{k-1}\sum_{t=0}^{k-1} \beta^{\tau^s(1+\tau^t 2^v)} = \sum_{t\ne u}\sum_{s=0}^{k-1} \beta^{\tau^s(\tau^{t'} 2^{\sigma(t,v)})} + \sum_{s=0}^{k-1} \beta^{\tau^s(1+\tau^u 2^v)} \\ &= \sum_{t\ne u}\sum_{s=0}^{k-1} \beta^{\tau^{s+t'} 2^{\sigma(t,v)}} + \sum_{s=0}^{k-1} 1 = \sum_{t\ne u} \alpha^{2^{\sigma(t,v)}} + k = \sum_{t\ne u} \alpha_{\sigma(t,v)} + k.\end{aligned} \tag{18}$$

Therefore $\alpha\alpha_i$ is computed by the sum of at most k basis elements in $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ for $i \ne v$ and $\alpha\alpha_v$ is computed by the sum of at most $k-1$ basis elements and the constant term $k \equiv 0, 1 \in GF(2)$.

4 New Multiplication Algorithm Using a Gaussian Normal Basis in $GF(2^m)$ for m Odd

4.1 Symmetry of $(\lambda_{ij}^{(s)})$ and (λ_{ij})

Efficient implementation of ECC over a binary field $GF(2^m)$ requires that m is *odd*, or more strongly m is *prime*. These conditions are necessary to avoid Pohlig-Helllman type attacks. For example, all the five binary fields $GF(2^m)$, $m =$

$163, 233, 283, 409, 571$ suggested by NIST [16] for ECDSA have the property that $m = prime$. Therefore it is not so serious restriction to assume that m is odd for a fast multiplication algorithm if one is interested in this kind of applications. For odd values of m, it is well known [15] that a Gaussian normal basis of type k or (m, k) always exists for some $k \geq 1$. Since $mk + 1$ is a prime with $m = odd$, it follows that k is even. Thus it is enough to study the multiplication in $GF(2^m)$ for odd m with a Gaussian normal basis of type k for even k. To derive a low complexity architecture, in view of the multiplication formulas (17) and (18), one should choose a small k, i.e. low complexity Gaussian normal basis. The least possible even $k \geq 1$ is $k = 2$. This is so called a type II ONB (optimal normal basis) or more specifically a type 2 Gaussian normal basis. Among the five binary fields recommended by NIST, $m = 233$ is the only case where a type II ONB exists. On the other hand, the lowest complexity Gaussian normal basis for the rest of the fields are type 4 Gaussian normal basis when $m = 163, 409$, type 6 Gaussian normal basis when $m = 283$, and type 10 Gaussian normal basis when $m = 571$ (See [12]).

Let $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ be any normal basis in $GF(2^m)$ with $\alpha_i = \alpha^{2^i}$ and let

$$\alpha\alpha_i = \sum_{j=0}^{m-1} \lambda_{ij}\alpha_j, \tag{19}$$

where λ_{ij} is in $GF(2)$. Taking repeated powers of 2 for both sides of the above equation, one finds

$$\lambda_{ij}^{(s)} = \lambda_{i-j,s-j}, \tag{20}$$

where $\lambda_{ij}^{(s)}$ is defined in (1). An explicit table of $\lambda_{ij}^{(s)}$ is necessary for construction of the multipliers of Agnew et al. and also of Reyhani-Masoleh and Hasan. Finding $\lambda_{ij}^{(s)}$ may not be so easy unless one has a sufficient information on the given normal basis. Also note that $(\lambda_{ij}^{(s)})$ is a symmetric matrix but (λ_{ij}) is not in general. However, it turns out that (λ_{ij}) is a symmetric matrix if a Gaussian normal basis of type k with k even is used. More precisely, we have the following.

Lemma 1. *If $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ is a Gaussian normal basis of type k where k is even, then we have*

$$\lambda_{ij}^{(0)} = \lambda_{ij}.$$

Proof. From (20), it is enough to show that $\lambda_{ij} = \lambda_{i-j,-j}$. From the formulas (17) and (18), it is clear that $\lambda_{ij} = 1$ if and only there exist odd pairs of (s, s') (mod k) such that

$$1 + \tau^s 2^i = \tau^{s'} 2^j, \tag{21}$$

where $\langle\tau\rangle$ is a unique multiplicative subgroup of order k in $GF(p)^{\times}$ with $p = mk + 1$. Let S be the set of all pairs (s, s') (mod k) satisfying (21) and same way define T as the set of all pairs (t, t') (mod k) satisfying $1 + \tau^t 2^{i-j} = \tau^{t'} 2^{-j}$. To prove $\lambda_{ij} = \lambda_{i-j,-j}$, it suffices to show that the sets S and T have the same cardinality. Dividing both sides of the equation (21) by $\tau^{s'}2^j$, we get

$$\tau^{-s'}2^{-j}+\tau^{s-s'}2^{i-j}=1. \tag{22}$$

Since the order of τ is k where k is even, we have $-1=\tau^{\frac{k}{2}}$ and therefore

$$\tau^{-s'}2^{-j}=1+\tau^{\frac{k}{2}+s-s'}2^{i-j}. \tag{23}$$

Since the map $f_S: S\to T$ defined by $f_S(s,s')=(\frac{k}{2}+s-s',-s')$ and the map $f_T: T\to S$ defined by $f_T(t,t')=(\frac{k}{2}+t-t',-t')$ give one to one correspondence, i.e. $f_S\circ f_T=id=f_T\circ f_S$, we are done. $\square$

4.2 Construction of a sequential multiplier and complexity analysis

Now from (6) and also from Lemma 1, we have c_s of $C=\sum_{i=0}^{m-1}c_s\alpha_s=AB$ as

$$c_s=\sum_{i,j}a_{i+s}b_{j+s}\lambda_{ij}^{(0)}=\sum_{i,j}a_{i+s}b_{j+s}\lambda_{ij}=\sum_{j=0}^{m-1}(\sum_{i=0}^{m-1}a_{i+s}\lambda_{ij})b_{j+s}. \tag{24}$$

Let us define an element $x_{st},\ 0\le s,t\le m-1$, in $GF(2)$ as

$$x_{st}=(\sum_{i=0}^{m-1}a_{i+s}\lambda_{it})b_{t+s}, \tag{25}$$

with corresponding matrix $X=(x_{st})$. Then the tth column vector X_t of X is

$$X_t=(x_{0t},x_{1t},\cdots,x_{m-1,t})^T, \tag{26}$$

where $(x_{0t},x_{1t},\cdots,x_{m-1,t})^T$ is the transposition of the row vector $(x_{0t},x_{1t},\cdots,x_{m-1,t})$. Also the sum of all column vectors $X_t,\ t=0,1,\cdots,m-1$, is exactly

$$(c_0,c_1,\cdots,c_{m-1})^T, \tag{27}$$

because $\sum_{t=0}^{m-1}x_{st}=c_s$. Our purpose is to reduce the gate complexity of our multiplier by rearranging the column vectors X_t and reusing partial sums in the computation. Let $m-1=2\nu$ and define m by m matrix $Y=(y_{st})$ by the following permutation of the column vectors of X as follows; When ν is odd, define Y as

$$(X_\nu,\cdots,X_3,X_1,X_{m-1},X_{m-3},\cdots,X_{m-\nu},X_{\nu-1},\cdots,X_2,X_0,X_{m-2},\cdots,X_{m-\nu+1}), \tag{28}$$

and when ν is even, Y is defined as

$$(X_\nu,\cdots,X_2,X_0,X_{m-2},\cdots,X_{m-\nu},X_{\nu-1},\cdots,X_3,X_1,X_{m-1},X_{m-3},\cdots,X_{m-\nu+1}). \tag{29}$$

Then the sum of all column vectors $Y_t,\ 0\le t\le m-1$, of Y with $Y_t=(y_{0t},y_{1t},\cdots,y_{m-1,t})^T$ is same to the sum of all column vectors $X_t,\ 0\le t\le m-1$, of X which is $(c_0,c_1,\cdots,c_{m-1})^T$.

To derive a parallel-in, parallel-out multiplication architecture, we will compute the sum of shifted diagonal vectors of Y, instead of computing the sum of column vectors of Y. This can be done from the following observations. In the expression of the matrix Y, there are exactly $t-1$ columns between the vectors X_t and X_{m-t}. Also, sth entry of X_t and $s+t$th entry of X_{m-t} have the same terms of a_is in their summands. In other words, from (25), we have

$$x_{s+t,m-t} = (\sum_{i=0}^{m-1} a_{i+s+t}\lambda_{i,-t})b_s = (\sum_{i=0}^{m-1} a_{i+s}\lambda_{i-t,-t})b_s = (\sum_{i=0}^{m-1} a_{i+s}\lambda_{it})b_s, \quad (30)$$

where the third expression comes from the rearrangement of the summation on the subscript i and the last expression comes from Lemma 1 saying $\lambda_{ij} = \lambda_{i-j,-j}$. Thus x_{st} and $x_{s+t,m-t}$ have the same term $\sum_{i=0}^{m-1} a_{i+s}\lambda_{it}$ in their expression and this will save the number of XOR gates during the computation of AB.

Table 1. New multiplication algorithm

1. $A = \sum_{i=0}^{m-1} a_i\alpha_i$ and $B = \sum_{i=0}^{m-1} b_i\alpha_i$ are loaded in m-bit registers respectively. Also intermediate values $D_0, D_1, \cdots, D_{m-1}$ of the multiplication are all set to zero.
2. For $t = 0$ to $m-1$, do the following;

$$y_{s,s+t} + D_{s+t} \longrightarrow D_{s+t+1}, \quad (31)$$

   where the above computation is done in parallel for all $0 \le s \le m-1$.
3. After mth iteration, we have $D_i = c_i$ for all $0 \le i \le m-1$, where $AB = \sum_{i=0}^{m-1} c_i\alpha_i$.

Let us explain the above algorithm in detail. At the first cycle ($t = 0$), $D_{s+1} = D_s + y_{ss}$ are simultaneously computed for all $0 \le s \le m-1$, i.e. $D_1 = y_{00}, D_2 = y_{11}, \cdots, D_0 = y_{m-1,m-1}$. When $t = 1$, $D_{s+2} = D_{s+1} + y_{s,s+1}$ are simultaneously computed for all $0 \le s \le m-1$, i.e. $D_2 = D_1 + y_{01} = y_{00} + y_{01}, D_3 = D_2 + y_{12} = y_{11} + y_{12}, \cdots, D_1 = D_0 + y_{m-1,0} = y_{m-1,m-1} + y_{m-1,0}$. Finally, at mth ($t = m-1$) cycle, $D_s = D_{s-1} + y_{s,s-1}$ are simultaneously computed. That is,

$$\begin{aligned}
D_0 &= D_{m-1} + y_{0,m-1} = y_{00} + y_{01} + \cdots + y_{0,m-1} = c_0,\\
D_1 &= D_0 + y_{10} = y_{11} + y_{12} + \cdots + y_{10} = c_1,\\
&\cdots\cdots\\
&\cdots\cdots\\
D_{m-1} &= D_{m-2} + y_{m-1,m-2} = y_{m-1,m-1} + y_{m-1,0} + \cdots + y_{m-1,m-2} = c_{m-1}.
\end{aligned} \quad (32)$$

In other words, for a fixed s, the final value D_s is sequentially computed in the following order

$$D_s = \underbrace{\overbrace{y_{ss}}^{D_{s+1}} + y_{s,s+1}}_{D_{s+2}} + y_{s,s+2} + \cdots + y_{s,s-1} = \sum_{i=0}^{m-1} y_{s,s+i} = c_s. \quad (33)$$

Note that $y_{s-1,s}$ and y_{ss}, $0 \le s \le m-1$, in the equation (32), are from the same column Y_s of the matrix Y. Since Y is obtained by a column permutation of a matrix X, we conclude that $y_{s-1,s} = x_{s-1,s'}$ and $y_{ss} = x_{ss'}$ for some s' depending on s. Moreover from the equation (25), we get

$$x_{ss'} = (\sum_{i=0}^{m-1} a_{i+s}\lambda_{is'})b_{s'+s}, \quad and \quad x_{s-1,s'} = (\sum_{i=0}^{m-1} a_{i+s-1}\lambda_{is'})b_{s'+s-1}, \qquad (34)$$

which implies that $x_{s-1,s'}$ $(= y_{s-1,s})$ is obtained by right cyclic shifting by one position of the vectors a_is and b_is from the expression $x_{s,s'}$ $(= y_{s,s})$. Since this can be done without any extra cost, all the necessary gates to construct a circuit from the algorithm in Table 1 are the gates needed to compute the first (i.e. $t = 0$) clock cycle of the step 2 of the algorithm,

$$D_{s+1} = D_s + y_{ss}, \quad 0 \le s \le m-1. \qquad (35)$$

Recall that, for each s, there is a corresponding (because of a permutation) s' such that

$$y_{ss} = x_{ss'} = (\sum_{i=0}^{m-1} a_{i+s}\lambda_{is'})b_{s'+s}. \qquad (36)$$

If $s' \ne 0$, i.e. if $x_{ss'}$ is not in the 0th column of X, then from the equations (25) and (30), we find that the necessary XOR gates to compute $x_{ss'}$ and $x_{s+s',m-s'}$ (which are the diagonal entries of the matrix Y) can be shared. Note that $x_{ss'} = (\sum_{i=0}^{m-1} a_{i+s}\lambda_{is'})b_{s'+s}$ can be computed by one AND gate and at most $k-1$ XOR gates since the multiplication matrix (λ_{ij}) of a Gaussian normal basis of type k has at most k nonzero entries for each column (row) in view of the equation (17). Thus the total number of necessary gates to compute all $y_{ss} = x_{ss'}$ with $s' \ne 0$ is $m-1$ AND gates plus $\frac{m-1}{2}(k-1)$ XOR gates.

Table 2. Comparison with previously proposed architectures

	Critical path delay (Type II ONB case)	AND	XOR (Type II ONB case)	flip-flop
Massey and Omura [7]	$\le T_A + \lceil \log_2 (mk) \rceil T_X$ $(T_A + \lceil \log_2 (2m) \rceil T_X)$	C_N	$\le C_N - 1$ $(2m-2)$	$2m$
Agnew et al. [1]	$\le T_A + (1 + \lceil \log_2 k \rceil) T_X$ $(T_A + 2T_X)$	m	$\le C_N$ $(2m-1)$	$3m$
Reyhani-Masoleh and Hasan [3]	$\le T_A + (1 + \lceil \log_2 (k+2) \rceil) T_X$ $(T_A + 3T_X)$	m	$\le \frac{1}{2}(C_N + 1) + \lfloor \frac{m}{2} \rfloor$ $(\frac{3m-1}{2})$	$3m$
This paper	$\le T_A + (1 + \lceil \log_2 k \rceil) T_X$ $(T_A + 2T_X)$	m	$\le m + \frac{m-1}{2}(k-1)$ $(\frac{3m-1}{2})$	$3m$

When $s' = 0$, then the number of nonzero entries of λ_{i0}, $0 \le i \le m-1$, is one because $\alpha\alpha_0 = \alpha^2 = \alpha_1$. Therefore we need one AND gate and no XOR gate to compute $x_{ss'}$ with $s' = 0$. Since the addition $D_s + y_{ss}$, $0 \le s \le m-1$, in (35) needs one XOR gate for each $0 \le s \le m-1$, the total gate complexity of our multiplier is m AND gates plus at most $m + \frac{m-1}{2}(k-1)$ XOR gates. The critical path delay can also be evaluated easily. It is clear from (35) and (36) that the critical path delay is at most $T_A + (1 + \lceil \log_2 k \rceil)T_X$. We compare our sequential multiplier with other multipliers of the same kinds in Table 2. In the table, C_N denotes the number of nonzero entries in the matrix $(\lambda_{ij}^{(0)})$. It is well known [6] that $C_N \le mk + m - k$ if k is odd and $C_N \le mk - 1$ if k is even. In our case of $GF(2^m)$ with $m = odd$, it is easy, from (17) and (18), to see that C_N has a more strong bound $C_N \le mk - k + 1$. Thus the bounds $\le \frac{C_N+1}{2} + \lfloor \frac{m}{2} \rfloor$ in [3] is same to $\le \frac{mk-k+2}{2} + \frac{m-1}{2} = \frac{2m+mk-m-k+1}{2} = m + \frac{m-1}{2}(k-1)$. Consequently the circuit in [3] and our multiplier have more or less the same hardware complexity.

4.3 Gaussian normal basis of type 2 and 4 for ECC

Let $p = 2m + 1$ be a prime such that $gcd(2m/ord_p 2, m) = 1$, i.e. either 2 is a primitive root $\pmod p$ or $ord_p 2 = m$ and $m = odd$. Then the element $\alpha = \beta + \beta^{-1}$ where β is a primitive pth root of unity in $GF(2^{2m})$ forms a normal basis $\{\alpha_0, \alpha_1, \cdots, \alpha_{m-1}\}$ in $GF(2^m)$, which we call a Gaussian normal basis of type 2 (or a type II ONB). A multiplication matrix (λ_{ij}) of $\alpha\alpha_i$ has the following property; $\lambda_{ij} = 1$ if and only if $1 \pm 2^i \equiv \pm 2^j \pmod p$ for any choice of $\pm$ sign. This is obvious from the basic properties of Gaussian normal basis in Section 3. Since m divides $ord_p 2$, $i = 0$ is a unique value $\pmod m$ satisfying $1 \pm 2^i \equiv 0 \pmod p$. That is, $\alpha\alpha_0 = \alpha_1$ and the 0th row of (λ_{ij}) is $(0, 1, 0, \cdots, 0)$. If $i \ne 0$, then $1 \pm 2^i \not\equiv 0 \pmod p$ and thus ith $(i \ne 0)$ row of (λ_{ij}) contains exactly two nonzero entries. Therefore for the case of a type II optimal normal basis, we need m AND gates and $m + \frac{m-1}{2} = \frac{3m-1}{2}$ XOR gates. Also the critical path delay is $T_A + 2T_X$, while that of [3] is $T_A + 3T_X$. Let us give a more explicit example for the case $m = 5$.

Example 1. Let β be a primitive 11th root of of unity in $GF(2^{10})$ and let $\alpha = \beta + \beta^{-1}$ be a type II optimal normal element in $GF(2^5)$. The computations of $\alpha\alpha_i$, $0 \le i \le 4$, are easily done from the following table. For each block regarding K and K', (s, t) entry with $0 \le s \le 1$ and $0 \le t \le 4$ has the value $\tau^s 2^t$ and $1 + \tau^s 2^t$ respectively, where $\langle \tau \rangle = \langle -1 \rangle$ is a unique multiplicative subgroup of order 2 in $GF(11)^\times$.

Table 3. Computation of K_i and K_i' using a type II ONB in $GF(2^m)$ for $m = 5$

K_0	K_1	K_2	K_3	K_4	K_0'	K_1'	K_2'	K_3'	K_4'
1	2	4	8	5	2	3	5	9	6
−1	−2	−4	−8	−5	0	−1	−3	−7	−4

From the above table, it can be found that $\alpha\alpha_0 = \alpha_1$ and

$$\alpha\alpha_1 = \alpha_0 + \alpha_3,\ \alpha\alpha_2 = \alpha_3 + \alpha_4,\ \alpha\alpha_3 = \alpha_1 + \alpha_2,\ \alpha\alpha_4 = \alpha_2 + \alpha_4. \tag{37}$$

For example, the computation of $\alpha\alpha_3$ can be done as follows. See the block K_3' and find $9 \equiv -2 \pmod{11}$ is in K_1 and $-7 \equiv 4$ is in K_2. Thus we have $\alpha\alpha_3 = \alpha_1 + \alpha_2$. In fact, for the case of type II ONB, there is a more regular expression called a palindromic representation which enables us to find the multiplication table more easily. However for the general treatments of all Gaussian normal bases of type k for arbitrary k, we are following this rule. Note that for all other type II ONB where $m \neq 5$, the multiplication table can be derived exactly the same manner. From (37), the corresponding matrix (λ_{ij}) for $m = 5$ is

$$(\lambda_{ij}) = \begin{pmatrix} 0\,1\,0\,0\,0 \\ 1\,0\,0\,1\,0 \\ 0\,0\,0\,1\,1 \\ 0\,1\,1\,0\,0 \\ 0\,0\,1\,0\,1 \end{pmatrix}, \tag{38}$$

and using (24),(25),(28),(29), we find that the multiplication $C = \sum_{i=0}^{4} c_i\alpha_i$ of $A = \sum_{i=0}^{4} a_i\alpha_i$ and $B = \sum_{i=0}^{4} b_i\alpha_i$ is written as follows.

$$\begin{aligned}
c_0 &= \underline{(a_3 + a_4)b_2} + a_1b_0 + (a_1 + a_2)b_3 + (a_0 + a_3)b_1 + (a_2 + a_4)b_4 \\
c_1 &= (a_4 + a_0)b_3 + \underline{a_2b_1} + (a_2 + a_3)b_4 + (a_1 + a_4)b_2 + (a_3 + a_0)b_0 \\
c_2 &= (a_0 + a_1)b_4 + a_3b_2 + \underline{(a_3 + a_4)b_0} + (a_2 + a_0)b_3 + (a_4 + a_1)b_1 \\
c_3 &= (a_1 + a_2)b_0 + a_4b_3 + (a_4 + a_0)b_1 + \underline{(a_3 + a_1)b_4} + (a_0 + a_2)b_2 \\
c_4 &= (a_2 + a_3)b_1 + a_0b_4 + (a_0 + a_1)b_2 + (a_4 + a_2)b_0 + \underline{(a_1 + a_3)b_3}
\end{aligned} \tag{39}$$

From this, one has the shift register arrangement of $C = AB$ using a type II ONB in $GF(2^m)$ for $m = 5$ and it is shown in Figure 3. Note that the underlined entries are the first terms to be computed. Also note that the (shifted) diagonal entries have the common terms of a_is.

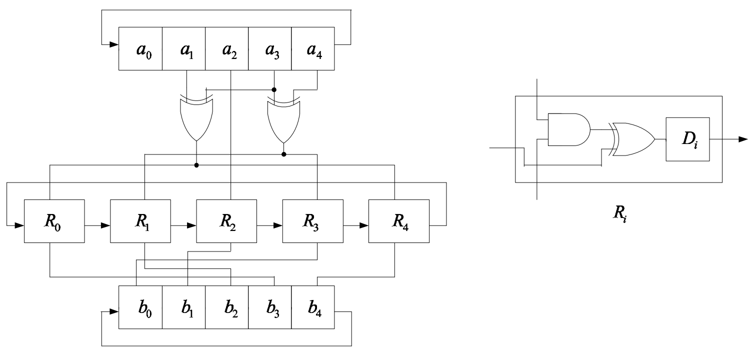

Fig. 3. A new multiplication circuit using a type II ONB in $GF(2^m)$ for $m = 5$

As is mentioned before, there exists only one field $GF(2^{233})$ for which a type II ONB exists in $GF(2^m)$ among the recommended five fields $GF(2^m)$, $m = 163, 233, 283, 409, 571$, by NIST. Though the circuits of multiplication using a type II ONB are presented in many places [1,3,10,11], the authors could not find an explicit example of a circuit design using a Gaussian normal basis of type $k > 2$. Since there are two fields $GF(2^{163}), GF(2^{409})$ for which a Gaussian normal basis of type 4 exists, it is worthwhile to study the multiplication and the corresponding circuit for this case. For the clarity of exposition, we will explain a Gaussian normal basis of type $k = 4$ in $GF(2^m)$ for $m = 7$. Note that the general case can be dealt in the same manner as in the following example.

Example 2. Let $p = 29 = mk + 1$ with $m = 7, k = 4$ where a Gauss period α of type $(7, 4)$ exists in $GF(2^7)$. In this case, the unique cyclic subgroup of order 4 in $GF(29)^\times$ is $K = \{1, 2^7, 2^{14}, 2^{21}\} = \{1, 12, 28, 17\}$. Let β be a primitive 29th root of unity in $GF(2^{28})$. Thus letting $\tau = 12$, a normal element α is written as $\alpha = \beta + \beta^{12} + \beta^{17} + \beta^{28}$ and $\{\alpha_0, \alpha_1, \cdots, \alpha_6\}$ is a normal basis in $GF(2^7)$. The computations of $\alpha\alpha_i$, $0 \le i \le 6$, are done from the following table. For each block regarding K and K', (s, t) entry with $0 \le s \le 3$ and $0 \le t \le 6$ has the value $\tau^s 2^t$ and $1 + \tau^s 2^t$ respectively.

Table 4. Computation of K_i and K_i' using a Gaussian normal basis of type $k = 4$ in $GF(2^m)$ for $m = 7$

K_0	K_1	K_2	K_3	K_4	K_5	K_6	K_0'	K_1'	K_2'	K_3'	K_4'	K_5'	K_6'
1	2	4	8	16	3	6	2	3	5	9	17	4	7
12	24	19	9	18	7	14	13	25	20	10	19	8	15
28	27	25	21	13	26	23	0	28	26	22	14	27	24
17	5	10	20	11	22	15	18	6	11	21	12	23	16

From the above table, we find $\alpha\alpha_0 = \alpha_1$ and

$$\begin{aligned} &\alpha\alpha_1 = \alpha_0 + \alpha_2 + \alpha_5 + \alpha_6,\ \alpha\alpha_2 = \alpha_1 + \alpha_3 + \alpha_4 + \alpha_5,\ \alpha\alpha_3 = \alpha_2 + \alpha_5, \\ &\alpha\alpha_4 = \alpha_2 + \alpha_6,\ \alpha\alpha_5 = \alpha_1 + \alpha_2 + \alpha_3 + \alpha_6,\ \alpha\alpha_6 = \alpha_1 + \alpha_4 + \alpha_5 + \alpha_6. \end{aligned} \tag{40}$$

For example, see the block K_2' for the expression of $\alpha\alpha_2$. The entries of K_2' are $5, 20, 26, 11$. Now see the blocks of $K_i s$ and find $5 \in K_1, 20 \in K_3, 26 \in K_5, 11 \in K_4$. Thus we get $\alpha\alpha_2 = \alpha_1 + \alpha_3 + \alpha_4 + \alpha_5$. From (40), the multiplication matrix (λ_{ij}) is written as

$$(\lambda_{ij}) = \begin{pmatrix} 0&1&0&0&0&0&0 \\ 1&0&1&0&0&1&1 \\ 0&1&0&1&1&1&0 \\ 0&0&1&0&0&1&0 \\ 0&0&1&0&0&0&1 \\ 0&1&1&1&0&0&1 \\ 0&1&0&0&1&1&1 \end{pmatrix}, \tag{41}$$

and again using the relations (24),(25),(28),(29), we get the following multiplication result $C = AB = \sum_{i=0}^{6} c_i\alpha_i$. In the following table, a_{ijkl} is defined as $a_{ijkl} = a_i + a_j + a_k + a_l$. For example, we have $c_0 = (a_2 + a_5)b_3 + (a_0 + a_2 + a_5 + a_6)b_1 + (a_1 + a_4 + a_5 + a_6)b_6 + (a_2 + a_6)b_4 + (a_1 + a_3 + a_4 + a_5)b_2 + a_1 b_0 + (a_1 + a_2 + a_3 + a_6)b_5$.

$$
\begin{aligned}
c_0 &= \underline{(a_2+a_5)b_3} + a_{0256}b_1 + a_{1456}b_6 + (a_2+a_6)b_4 + a_{1345}b_2 + a_1b_0 + a_{1236}b_5 \\
c_1 &= (a_3+a_6)b_4 + \underline{a_{1360}b_2} + a_{2560}b_0 + (a_3+a_0)b_5 + a_{2456}b_3 + a_2b_1 + a_{2340}b_6 \\
c_2 &= (a_4+a_0)b_5 + a_{2401}b_3 + \underline{a_{3601}b_1} + (a_4+a_1)b_6 + a_{3560}b_4 + a_3b_2 + a_{3451}b_0 \\
c_3 &= (a_5+a_1)b_6 + a_{3512}b_4 + a_{4012}b_2 + \underline{(a_5+a_2)b_0} + a_{4601}b_5 + a_3b_3 + a_{4562}b_1 \qquad (42) \\
c_4 &= (a_6+a_2)b_0 + a_{4623}b_5 + a_{5123}b_3 + (a_6+a_3)b_1 + \underline{a_{5012}b_6} + a_4b_4 + a_{5603}b_2 \\
c_5 &= (a_0+a_3)b_1 + a_{5034}b_6 + a_{6234}b_4 + (a_0+a_4)b_2 + a_{6123}b_0 + \underline{a_6b_5} + a_{6014}b_3 \\
c_6 &= (a_1+a_4)b_2 + a_{6145}b_0 + a_{0345}b_5 + (a_1+a_5)b_3 + a_{0234}b_1 + a_0b_6 + \underline{a_{0125}b_4}
\end{aligned}
$$

The corresponding shift register arrangement of $C = AB$ using a Gaussian normal basis of type 4 in $GF(2^m)$ for $m = 7$ is shown in Figure 4. Also note that the underlined entries are the first terms to be computed and the (shifted) diagonal entries have the common terms of a_is. The critical path delay of the circuit using a type 4 Gaussian normal basis is only $T_A + 3T_X$ and can be effectively realized for the case $GF(2^{163})$ and $GF(2^{409})$ also.

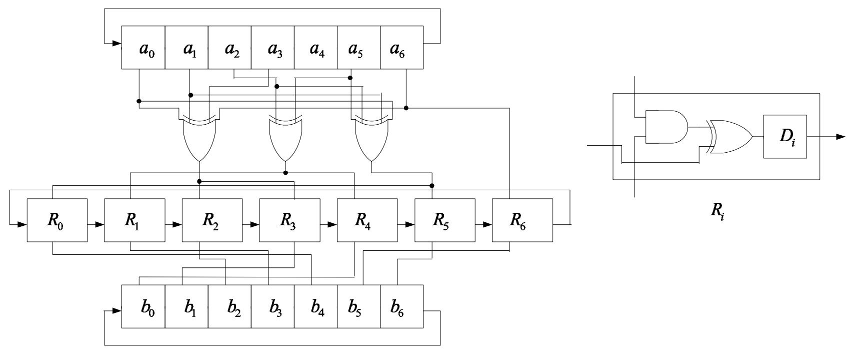

Fig. 4. A new multiplication circuit using a Gaussian normal basis of type 4 in $GF(2^m)$ for $m = 7$

5 Conclusions

In this paper, we proposed a low complexity sequential normal basis multiplier over $GF(2^m)$ for odd m using a Gaussian normal basis of type k. Since, in many cryptographic applications, m should be an odd integer or a prime, our assumption on m is not at all restrictive for a practical purpose. We presented a general method of constructing a circuit arrangement of the multiplier and showed explicit examples for the cases of type 2 and 4 Gaussian normal bases. Among the five binary fields, $GF(2^m)$ with $m = 163, 233, 283, 409, 571$, recommended by NIST [16] for ECC, our examples cover the cases $m = 163, 233, 409$ since

$GF(2^{233})$ has a type II ONB and $GF(2^{163}), GF(2^{409})$ have a Gaussian normal basis of type 4. Our general method can also be applied to other fields $GF(2^{283})$ and $GF(2^{571})$ since they have a Gaussian normal basis of type 6 and 10, respectively. Compared with previously proposed architectures of the same kinds, our multiplier has a superior or comparable area complexity and delay time. Thus it is well suited for many applications such as VLSI implementation of elliptic curve cryptographic protocols.

References

1. G.B. Agnew, R.C. Mullin, I. Onyszchuk, and S.A. Vanstone, "An implementation for a fast public key cryptosystem," *J. Cryptology*, vol. 3, pp. 63–79, 1991.
2. G.B. Agnew, R.C. Mullin, and S.A. Vanstone, "Fast exponentiation in $GF(2^n)$," *Eurocrypt 88, Lecture Notes in Computer Science*, vol. 330, pp. 251–255, 1988.
3. A. Reyhani-Masoleh and M.A. Hasan, "Low complexity sequential normal basis multipliers over $GF(2^m)$," *16th IEEE Symposium on Computer Arithmetic*, vol. 16, pp. 188–195, 2003.
4. A. Reyhani-Masoleh and M.A. Hasan, "A new construction of Massey-Omura parallel multiplier over $GF(2^m)$," *IEEE Trans. Computers*, vol. 51, pp. 511–520, 2002.
5. A. Reyhani-Masoleh and M.A. Hasan, "Efficient multiplication beyond optimal normal bases," *IEEE Trans. Computers*, vol. 52, pp. 428–439, 2003.
6. A.J. Menezes, I.F. Blake, S. Gao, R.C. Mullin, S.A. Vanstone, and T. Yaghoobian, *Applications of Finite Fields*, Kluwer Academic Publisher, 1993.
7. J.L. Massy and J.K. Omura, "Computational method and apparatus for finite field arithmetic," *US Patent No. 4587627*, 1986.
8. C. Paar, P. Fleischmann, and P. Roelse, "Efficient multiplier architectures for Galois fields $GF(2^{4n})$," *IEEE Trans. Computers*, vol. 47, pp. 162–170, 1998.
9. E.R. Berlekamp, "Bit-serial Reed-Solomon encoders," *IEEE Trans. Inform. Theory*, vol. 28, pp. 869–874, 1982.
10. B. Sunar and Ç.K. Koç, "An efficient optimal normal basis type II multiplier," *IEEE Trans. Computers*, vol. 50, pp. 83–87, 2001.
11. H. Wu, M.A. Hasan, I.F. Blake, and S. Gao, "Finite field multiplier using redundant representation," *IEEE Trans. Computers*, vol. 51, pp. 1306–1316, 2002.
12. S. Gao, J. von zur Gathen, and D. Panario, "Orders and cryptographical applications," *Math. Comp.*, vol. 67, pp. 343–352, 1998.
13. J. von zur Gathen and I. Shparlinski, "Orders of Gauss periods in finite fields," *ISAAC 95, Lecture Notes in Computer Science*, vol. 1004, pp. 208–215, 1995.
14. S. Gao and S. Vanstone, "On orders of optimal normal basis generators," *Math. Comp.*, vol. 64, pp. 1227–1233, 1995.
15. S. Feisel, J. von zur Gathen, M. Shokrollahi, "Normal bases via general Gauss periods," *Math. Comp.*, vol. 68, pp. 271–290, 1999.
16. NIST, "Digital Signature Standard," *FIPS Publication*, 186-2, February, 2000.