Subquadratic space complexity multiplier for a class of binary fields using Toeplitz matrix approach

M. A. Hasan and C. Negre

1ECE Department and CACR, University of Waterloo, Ontario, Canada.
2Team DALI/ELIAUS, Université de Perpignan, France.

Abstract

In the recent past, subquadratic space complexity multipliers have been proposed for binary fields defined by irreducible trinomials and some specific pentanomials. For such multipliers, alternative irreducible polynomials can also be used, in particular, nearly all one polynomials (NAOPs) seem to be better than pentanomials (see [7]). For improved efficiency, multiplication modulo an NAOP is performed via modulo a quadrinomial whose degree is one more than that of the original NAOP. In this paper, we present a Toeplitz matrix-vector product based approach for multiplication modulo a quadrinomial. We obtain a fully parallel (non-sequential) multiplier with a subquadratic space complexity, which has the same order of space complexity as that of Fan and Hasan [4].

The Toeplitz matrix-vector product based approach is also interesting in the design of sequential multipliers. In this paper, we present two such multipliers: one with bit serial output and the other bit parallel output.

Index Terms

Subquadratic complexity, binary field, multiplication, double basis.

1. Introduction

For hardware implementation of certain cryptosystems [2], [9], [8], a finite field multiplier can be one of the most circuit or space demanding blocks. In order to make such a multiplier circuit-efficient, low weight irreducible polynomials are used for defining the representation of the field elements. For an irreducible polynomial, with coefficients being 0 and 1 only, the least weight is three. Such irreducible binary trinomials however do not exist for all degrees. The second least weight for an irreducible binary polynomial is five and such pentanomials appear to exist for all practical purpose.

A slightly different approach is to use a low weight composite, instead of irreducible, polynomial to define the representation of the field elements [7]. This leads to redundancy in the representation, but can reduce the circuit requirement of the multiplier. To this end, in the past composite binomials of the form \( X^n + 1 \) have been suggested. For reduced redundancy, such a binomial is chosen to be the product of \( X + 1 \) and an irreducible all-one polynomial (AOP) [6]. The latter is however not so abundant and the use of nearly AOP (NAOP) has been suggested. Irreducible NAOPs appear to be abundant. The multiplication of \( X + 1 \) and an NAOP results in a polynomial of weight four. Such a composite quadrinomial may be a better choice than an irreducible pentanomial when an irreducible trinomial is not available.

In this paper, we use quadrinomials. In addition, for multiplication we represent the two inputs with respect to two different bases. The main motivation of using two bases is to be able to formulate the field multiplication as a Toeplitz matrix-vector product (TMVP). It is well known that such a product can be obtained in a bit parallel fashion with subquadratic circuit/space complexity [4]. In the recent past there has been a considerable amount of interest to design sub-quadratic space complexity field multipliers, especially for cryptographic applications where the dimension of the field is of intermediate sizes, i.e., in the range of hundreds.

The main contributions of this work are as follows. For the modified polynomial basis \( B \) introduced in [10], we have formulated the problem of binary field multiplication as a TMVP. The TMVP approach has been known for other bases, but to the best of our knowledge this is the first time that a direct TMVP approach has been derived for the modified polynomial basis. The TMVP formulation for the modified polynomial basis has been achieved by expressing one of the inputs with respect to another basis \( B' \). We are not aware of the prior use of \( B' \) in finite field arithmetic and in our work, we have given explicit formulas for basis conversions— from \( B \) to \( B' \) and vice versa. We have also proposed bit serial multiplier structures involving bases \( B \) and \( B' \). Although bit serial multipliers are available for various bases, to the best of our knowledge the proposed structures are the first for these \( B \) and \( B' \) bases.
2. Binary field multiplication

A binary field $\mathbb{F}_{2^{n-1}}$ is the set of binary polynomials modulo an irreducible polynomial $P$ of degree $n - 1$

$\mathbb{F}_{2^{n-1}} = \mathbb{F}_2[X]/(P)$.

Each element can be seen as a polynomial of degree at most $n - 2$ and they are often expressed in the polynomial basis $\{1, X, X^2, \ldots, X^{n-2}\}$. In $\mathbb{F}_{2^{n-1}}$, for such representation, field operations like multiplication and inversion are done modulo $P$. Multiplication is widely used in practice, in this paper we focus on this operation. Let $A = \sum_{i=0}^{n-2} a_i X^i$ and $B = \sum_{i=0}^{n-2} b_i X^i$ be two elements expressed in $B$. We can compute $C = A \times B \mod P$ as

$$C = A \times B = \sum_{i=0}^{n-2} (AX^i)b_i \mod P = \sum_{i=0}^{n-2} A^{(i)}b_i,$$

after expanding the expression of $B$ in $B$ and noting that $A^{(i)} = AX^i \mod P$. This can be written through a matrix vector product

$$C = \begin{bmatrix} A^{(0)} & A^{(1)} & \cdots & A^{(n-2)} \end{bmatrix} \cdot B.$$

The two most used strategies to design an efficient hardware multiplier via the above matrix vector product are the following:

1) The choice of the polynomial $P$ must provide an efficient computation of the column $A^{(i)}$. Until now all the one polynomials (AOP) and trinomials seems to be the best possible choice. However, neither of them exists for all degree. Consequently other type of irreducible polynomials have been considered. A class of pentanomials of the form

$$P = X^{n-1} + X^{k+2} + X^{k+1} + X^k + 1$$

seems to be very interesting for the computation of the columns $A^{(i)}$ [4], [11]. Almost irreducible trinomials have also been proposed [7], [1], [3]; these trinomials $X^{n-1+\delta} + X^{k+1}$ have an irreducible factor of degree $n - 1$. For a large set of field $\delta$ has really small value.

2) The second strategy consists of expressing the matrix

$$\begin{bmatrix} A^{(0)} & A^{(1)} & \cdots & A^{(n-2)} \end{bmatrix}$$

to a Toeplitz form. We will see in Subsection 2.2 that a Toeplitz matrix vector product can be done efficiently through a subquadratic complexity circuit. Generally we can obtain the Toeplitz form of the matrix by performing some row operations or column operations, or in other words, by using different bases of representation.

2.1. Nearly all one polynomials

Here we will design a multiplier modulo the irreducible polynomial introduced by Katti and Brennan in [7]. We call such polynomials nearly all one polynomials (NAOPs). They have the following form $P = \sum_{i=0}^{k_2-1} X^i + \sum_{i=k_1}^{n-1} X^i$ with $k_2 < k_1$. In other words, for a NAOP all the coefficients are equal to 1 unless they are in an interval $[k_2, k_1 - 1]$. If $P$ is irreducible, we can define $\mathbb{F}_{2^{n-1}}$ as $\mathbb{F}_2[X]/(P)$.

As noticed in [7], multiplication in these fields can be done efficiently modulo $Q = (X + 1) \times P$ since $Q$ is a quadrinomial

$$Q = 1 + X^{k_2} + X^{k_1} + X^n.$$

In Table 1 we give several irreducible NAOPs with degrees suitable for cryptographic applications. Irreducible NAOPs are abundant, so we do not list all of them for each degree. However it appears to be an open question that there exists a NAOP for each degree. As we will see later, irreducible NAOPs which satisfy $k_1 = k_2 + 1$ give a more efficient multiplier. As much as possible we give such NAOPs for each degree $(n - 1)$ in Table 1 (they are marked by $\dagger$). When there are no irreducible NAOPs with $k_1 = k_2 + 1$ for a certain degree, Table 1 lists the NAOP with minimum $k_1$ for the smallest possible $k_2$.

<table>
<thead>
<tr>
<th>$n-1$</th>
<th>$k_1, k_2$</th>
<th>$n-1$</th>
<th>$k_1, k_2$</th>
<th>$n-1$</th>
<th>$k_1, k_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>161</td>
<td>66, 65</td>
<td>193</td>
<td>116, 115</td>
<td>251</td>
<td>247</td>
</tr>
<tr>
<td>163</td>
<td>33, 22</td>
<td>194</td>
<td>117, 116</td>
<td>252</td>
<td>248</td>
</tr>
<tr>
<td>164</td>
<td>7, 5</td>
<td>195</td>
<td>118, 117</td>
<td>253</td>
<td>249</td>
</tr>
<tr>
<td>165</td>
<td>15, 8</td>
<td>196</td>
<td>119, 118</td>
<td>254</td>
<td>250</td>
</tr>
<tr>
<td>166</td>
<td>22, 12</td>
<td>197</td>
<td>120, 119</td>
<td>255†</td>
<td>234, 233</td>
</tr>
<tr>
<td>167†</td>
<td>7, 6</td>
<td>189†</td>
<td>35, 34</td>
<td>256†</td>
<td>254</td>
</tr>
<tr>
<td></td>
<td></td>
<td>200</td>
<td>10, 2</td>
<td>257†</td>
<td>53, 52</td>
</tr>
<tr>
<td></td>
<td></td>
<td>219†</td>
<td>24, 23</td>
<td>258†</td>
<td>69, 68</td>
</tr>
<tr>
<td></td>
<td></td>
<td>226†</td>
<td>259</td>
<td>260†</td>
<td>9, 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>247†</td>
<td>34, 33</td>
<td>251†</td>
<td>35, 34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256†</td>
<td>35, 34</td>
</tr>
</tbody>
</table>

Table 1. Irreducible NAOP of degree $n - 1$

2.2. Asymptotic Complexities of Toeplitz Matrix Vector Product

As previously mentioned, multiplication in the binary field is often expressed as a Toeplitz matrix vector product. In this section we recall the method used to build a subquadratic circuit which computes a Toeplitz-matrix-vector multiplication. Recall that a Toeplitz matrix is defined as
Definition 1: An \( n \times n \) Toeplitz matrix is a matrix \([t_{i,j}]_{0 \leq i, j \leq n-1}\) such that \( t_{i,j} = t_{i-1,j-1} \) for \( 1 \leq i, j \).

If \( 2|n \) we can use a two way approach presented in Table 2 to compute a matrix vector product \( T \cdot V \), where \( T \) is an \( n \times n \) Toeplitz matrix. If \( 3|n \) we can use the three way approach which is also presented in Table 2.

Table 2. Subquadratic complexity Toeplitz matrix product

<table>
<thead>
<tr>
<th>Two-way</th>
<th>Three-way</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T = \begin{bmatrix} T_1 &amp; 0 &amp; \cdots &amp; 0 \ T_2 &amp; T_1 &amp; \cdots &amp; 0 \ \vdots &amp; \vdots &amp; \ddots &amp; \vdots \ T_n &amp; T_{n-1} &amp; \cdots &amp; T_1 \end{bmatrix} )</td>
<td>( T = \begin{bmatrix} T_2 &amp; T_1 &amp; 0 &amp; \cdots &amp; 0 \ T_3 &amp; T_2 &amp; T_1 &amp; \cdots &amp; 0 \ \vdots &amp; \vdots &amp; \ddots &amp; \ddots &amp; \vdots \ T_{n+1} &amp; T_n &amp; T_{n-1} &amp; \cdots &amp; T_2 \end{bmatrix} )</td>
</tr>
</tbody>
</table>

Recursive formulas

\[
T \cdot V = \begin{bmatrix} P_0 + P_2 \\ P_1 + P_3 \\ P_2 \\ P_3 \end{bmatrix} \quad T \cdot V = \begin{bmatrix} P_0 + P_3 + P_4 \\ P_1 + P_4 + P_5 \\ P_2 + P_4 + P_6 \\ P_3 \end{bmatrix}
\]

where

\[
P_0 = (T_0 + T_1) \cdot V_0, \quad P_0 = (T_0 + T_1 + T_2) \cdot V_2, \\
P_1 = (T_1 + T_2) \cdot V_0, \quad P_1 = (T_0 + T_1 + T_2) \cdot V_1, \\
P_2 = T_1 \cdot (V_0 + V_1), \quad P_2 = (T_2 + T_3) \cdot V_0, \\
P_3 = T_2 \cdot (V_0 + V_2), \quad P_3 = T_1 \cdot (V_1 + V_2), \\
P_4 = T_3 \cdot (V_0 + V_3), \quad P_5 = T_4 \cdot (V_0 + V_4).
\]

If \( n \) is a power of 2 or a power of 3 the formulas of Table 2 can be used recursively to perform \( T \cdot V \). Using these recursive processes through parallel computation, the resulting multipliers [4] have the complexity given in Table 3. In this table \( A_T \) represents the delay of an AND gate and \( D_X \) the delay of an XOR gate. It is also possible to design subquadratic TMVP multipliers for size \( n = 3^2 2^j \) by combining two-way and three-way splits in the recursive computations.

Table 3. Asymptotic complexity

<table>
<thead>
<tr>
<th># AND</th>
<th>Two-way split method</th>
<th>Three-way split method</th>
</tr>
</thead>
<tbody>
<tr>
<td># XOR</td>
<td>( \frac{n \log_2(3)}{8} ) + ( 0.5 n )</td>
<td>( \frac{24 n \log_2(6)}{D_A + 3 \log_3(n)} )</td>
</tr>
<tr>
<td>Delay</td>
<td>( \frac{D_A + 2 \log_2(n) D_X}{D_A + 2 \log_2(n) D_X} )</td>
<td>( \frac{24 n \log_2(6)}{D_A + 3 \log_3(n) D_X} )</td>
</tr>
</tbody>
</table>

3. Double basis representation

Let \( Q = 1 + X^{k_2} + X^{k_1} + X^n \) be a quadrinomial with \( k_2 < k_1 < n \) in \( \mathbb{F}_2[X] \). Let \( l_1 = n - k_1 \) and \( l_2 = n - k_2 \). We present now our contribution on multiplication modulo \( Q \). Our main goal is to get a subquadratic complexity multiplier modulo such a quadrinomial. To reach this goal, we attempt to express the product of two elements modulo \( Q \) as a Toeplitz matrix vector product.

First, we represent the elements of \( \mathbb{F}_2[X] \) using the basis \( B = \{ e_0, e_1, \ldots, e_{n-1} \} \) given in equation (1). This basis was introduced in [10]. In this basis the matrix involved in the multiplication is easier to compute than in the polynomial basis. However the matrix of [10] does not have the Toeplitz form. Here we obtain the Toeplitz form by performing some column operations on this matrix. These column operations should also be performed on the entries of \( B \) which is thus expressed in different or second basis \( B' \) and is given below.

\[
B = \begin{bmatrix} b_1 \mid b_2 \mid \cdots \mid b_{n-1} \end{bmatrix} \quad \text{and} \quad B' = \begin{bmatrix} e_1 \mid e_2 \mid \cdots \mid e_{n-1} \end{bmatrix}
\]

Let us specify the general construction of such a double basis multiplier. Let \( A = \sum_{i=0}^{n-1} a_i e_i \) be expressed relative to \( B \) and \( B = \sum_{j=0}^{n-1} b_j e_j \) be expressed relative to \( B' \). The product \( C = AB \) can be written as

\[
C = A \left( \sum_{j=0}^{n-1} b_j e_j' \right) = \sum_{j=0}^{n-1} b_j' (Ae_j')
\]

(2)

So if we denote \( A(j) = Ae_j' \) we obtain \( C = \sum_{j=0}^{n-1} b_j A(j) \).

Then if \( A(j) \) is expressed with respect to \( B \), using vector notations we obtain the \( B \) representation of \( C \) as

\[
C = \left[ A(0), A(1), \ldots, A(n-1) \right] B = M_A B
\]

(3)

We will show that a simple permutation of the columns of \( M_A \) results in a Toeplitz matrix, and thus the previous product can be performed using subquadratic approach given in Subsection 2.2.

To have a complete multiplier, one would expect to use the same basis to represent the elements \( A, B, C \). Here we use the basis \( B \). Thus, there is a preliminary step which performs a conversion of \( B \) from \( B \) to \( B' \). Below we present formulas to perform this conversion.

3.1. Conversion from \( B \) to \( B' \)

We assume here that \( l_1 > l_2 - l_1 \). Let \( B = \sum_{i=0}^{n-1} b_i e_i \) be expressed relative to \( B \). To convert the coefficient of \( B \) from \( B \) to \( B' \) we use the relation

\[
e_i = e'_i \quad \text{if} \quad i \in [0, l_1] \cup [l_2, n],
\]

\[
e_i = X^{i-l_1} + e'_i - e'_{i-l_1} \quad \text{if} \quad i \in [l_1, l_2[.
\]

For \( B \), we can write

\[
B = \sum_{i=0}^{l_1-1} b_i e_i + \sum_{i=l_1}^{l_2-1} b_i e_i + \sum_{i=l_1}^{n-1} b_i e_i
\]

\[
= \sum_{i=0}^{l_1-1} b_i e'_i + \sum_{i=l_1}^{l_2-1} b_i (e'_i + e'_{i-l_1}) + \sum_{i=l_2}^{n-1} b_i e'_i
\]

\[
= \sum_{i=0}^{l_1-1} (b_i + b_{i+l_2}) e'_i + \sum_{i=l_2}^{n-1} b_i e'_i
\]

In other words \( b'_i = b_i + b_{i+l_2} \) if \( i < l_2 - l_1 \) and \( b'_i = b_i \) if \( i \geq l_2 - l_1 \).
3.2. Conversion from $B'$ to $B$

The reverse conversion is not required in our double basis multiplier, but for completeness we present it here. We need to express $b_i$ in terms of $b'_j$. From the previous conversion from $B$ to $B'$ we know that

\[
\begin{align*}
  b_i &= b'_i \text{ if } i \geq l_2 - l_1, \\
  b_i &= b'_i + b_{i+l_1} \text{ if } i < l_2 - l_1.
\end{align*}
\]

Thus we need to replace $b_{i+l}$ by its own expression in terms of $b'_i$.

\[
\begin{align*}
  b_i &= b'_i + b'_{i+l_1} \text{ if } i + l_1 \geq l_2 - l_1, \\
  b_i &= b'_i + b'_{i+l_1} + b_{i+2l_1} \text{ if } i + l_1 < l_2 - l_1.
\end{align*}
\]

For the last case, we have to repeat the process with $b_{i+2l_1}$. If we expand the recursion we obtain the following formula

\[
b_i = b'_i + b'_{i+l_1} + \ldots + b'_{i+al_1} \text{ with } \alpha = \left\lfloor \frac{l_2 - l_1 - i}{l_1} \right\rfloor.
\]

Consequently if such conversion is needed, it is better to take quadrinomial such that $l_1 > l_2 - l_1$ since in this case $\alpha$ is at most equal to 1 and conversion requires at most $l_1 - l_2$ XOR operations. Most of the NAOP given in Table 1 satisfy $l_1 > l_2 - l_1$ and thus conversions between $B$ and $B'$ can be done using the formulas of the current section.

4. Construction of the matrix $M_A$

Let $M_A = [A^{(0)}, A^{(1)}, \ldots, A^{(n-1)}]$ be the resulting matrix of the double basis multiplier associated to bases $B$ and $B'$ defined in (1). The $n \times n$ matrix $M_A$ can be generated column by column starting from the left end as follows.

4.1. Generating the first $l_2$ columns of $M_A$.

First, note that $A^{(0)} = A$ and for $1 \leq i \leq l_2 - 1$ one can write

\[
A^{(i)} = A e'_{i} = AX^{i} = AX^{i-1}X = A^{(i-1)}X \text{ mod } Q.
\]

If we call $a^{(i)}_j$ the $j$-th coefficient in $B$ of $A^{(i)}$, the previous equation becomes

\[
A^{(i)} = \sum_{j=0}^{n-1} a^{(i)}_j e_j X.
\]

We need to express $e_j X$ in $B$ and this is the goal of the following lemma.

**Lemma 1:** Let $B = \{e_0, \ldots, e_{n-1}\}$ as defined in (1). Then the following equation holds

\[
e_{i}X = \begin{cases} 
  e_{i+1} & \text{if } i \neq n - 1, l_1 - 1, l_2 - 1, n - 1, \\
  e_{i+1} + e_0 & \text{if } i = l_1 - 1, l_2 - 1, \\
  e_0 & \text{if } i = n - 1.
\end{cases}
\]

**Proof:** Suppose that $i \neq n - 1, l_1 - 1, l_2 - 1, n - 1$. We recall that

\[
e_{i} = \begin{cases} 
  X^i & \text{if } 0 \leq i < l_1 - 1, \\
  X^i + X^{i-l_1} & \text{if } l_1 \leq i < l_2 - 2, \\
  X^i + X^{i-l_1} + X^{i-2} & \text{if } i < n - 1.
\end{cases}
\]

Thus if we multiply these expressions with $X$, we get

\[
e_{i+1}X = \begin{cases} 
  X^{i+1} = e_{i+1} & \text{if } 0 \leq i < l_1 - 1, \\
  X^{i+1} + X^{i+1-l_1} = e_{i+1} & \text{if } l_1 \leq i < l_2 - 1, \\
  X^{i+1} + X^{i+1-l_1} + X^{i+1-l_2} = e_{i+1} & \text{if } l_2 \leq i < n - 1.
\end{cases}
\]

For the special cases $i \in \{l_1 - 1, l_2 - 1, n - 1\}$, we have

\[
e_{l_1-1}X = X^{l_1-1}X = (X^{l_1} + 1) = e_{l_1} + e_0, \\
e_{l_2-1}X = (X^{l_2-1} + X^{l_1-1})X = (X^{l_2} + X^{l_2-l_1} + 1) + 1 = e_{l_2} + e_0, \\
e_{n-1}X = X^n + X^{k_2} + X^{k_1} \equiv 1 \text{ mod } Q = e_0.
\]

This completes the proof. □

Now we can write

\[
A^{(i)} = \sum_{j=0}^{n-1} a^{(i)}_j e_j X
\]

\[
= \left( \sum_{j=0}^{n-2} a^{(i-1)}_j e_{j+1} + a^{(i-1)}_{n-1} e_0 \right) + a^{(i-1)}_{l_1-1} e_0,
\]

This previous expression enables us to compute $A^{(i)}$ as illustrated below:

\[
\begin{array}{cccccccc}
  A^{(0)} & A^{(1)} & A^{(2)} & \cdots \\
  \downarrow & \downarrow & \downarrow & \cdots \\
  a_0 & a_{n-1} + a_{l_1-1} + a_{l_2-1} + a_{n-2} + a_{l_1-2} + a_{l_2-2} & \cdots \\
  a_1 & a_0 & a_{n-1} + a_{l_1-1} + a_{l_2-1} & \cdots \\
  a_2 & a_1 & a_0 & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
  a_{n-1} & a_{n-2} & a_{n-3} & \cdots
\end{array}
\]

This gives the column $A^{(i)}$ of $M_A$ for $i < l_2$. Now consider how to express $A^{(i)}$ for $l_2 \leq i \leq n - 1$.

4.2. Generating the last $k_2$ columns of $M_A$

We remark that $A^{(i)} = A(X^i + X^{i-l_1} + X^{i-l_2})$ for $i = l_2, l_2 + 1, \ldots, n - 1$. Thus, if we factorize $X$ out from the right side we obtain for $i = l_2, l_2 + 1, \ldots, n - 1$

\[
A^{(i)} = (X^{i-1} + X^{i-l_1} + X^{i-l_2})X = A^{(i-1)}X,
\]

which can be rewritten as

\[
A^{(i-1)} = A^{(i)}X^{-1}.
\]

Again if we replace $A^{(i)}$ by its expression in $B$ we get

\[
A^{(i-1)} = \sum_{j=0}^{n-1} a^{(i)}_j e_j X^{-1}.
\]

To proceed, we need to compute $e_j X^{-1}$ and this is done in the following lemma.
Lemma 2: Let $B = \{e_0, \ldots, e_{n-1}\}$ be the basis given in of $\mathbb{F}_2[X]/(Q(X))$. Then we have

$$e_i X^{-1} = \begin{cases} 
  e_{i-1} & \text{if } i \neq 0, l_1, l_2, \\
  e_{i-1} + e_{n-1} & \text{if } i = l_1, l_2, \\
  e_{n-1} & \text{if } i = 0.
\end{cases} \quad (4)$$

Proof: We first deal with the cases $i \neq 0, l_1, l_2$. We have

$$e_i = \begin{cases} 
  X^i & \text{if } 0 < i < l_1, \\
  X^i + X^{i-l_1} & \text{if } l_1 < i < l_2, \\
  X^i + X^{i-l_1} + X^{i-l_2} & \text{if } l_2 < i < n.
\end{cases}$$

If we multiply $e_i$ with $X^{-1}$ we get

$$e_i X^{-1} = \begin{cases} 
  X^{i-1} & \text{if } 0 < i < l_1, \\
  X^{i-1} + X^{i-l_1} = e_{i-1} & \text{if } l_1 < i < l_2, \\
  X^{i-1} + X^{i-l_1} + X^{i-l_2} = e_{i-1} & \text{if } l_2 < i < n,
\end{cases}$$

as required.

Now we assume that $i = 0, l_1$ or $l_2$. To prove these three cases we use $e_{n-1} = X^{-1}$ mod $Q$, indeed

$$e_{n-1}X = (X^{n-1} + X^{n-l_1} + X^{n-l_2})X = X^n + X^{k_1} + X^{k_2} = 1 \mod Q.$$

Thus $e_0 X^{-1} = X^{-1} = e_{n-1} \mod Q$. Similarly, we have

$$e_{l_1} X^{-1} = X^{l_1-1} + X^{-1} = e_{l_1-1} + e_{n-1}, \quad \text{and} \quad e_{l_2} X^{-1} = e_{l_2-1} + e_{n-1}.$$

For $A^{(i-1)}$,

$$A^{(i-1)} = \sum_{j=0}^{n-2} a_j^{(i)} e_j X^{-1} = \sum_{j=1}^{n-2} a_j^{(i)} e_{j-1} + a_0^{(i)} e_{n-1} + a_{l_2}^{(i)} e_{n-1} + a_{l_1}^{(i)} e_{n-1}.$$

We can thus compute $A^{(i)}$ for $i = n-1, \ldots, l_2$ recursively, beginning with $A^{(n)}$ and multiplying it by $X^{-1}$. For $A^{(n)}$ we have

$$A^{(n)} = A \times (X^n + X^{n-l_1} + X^{n-l_2}) = A \times (X^n + X^{k_1} + X^{k_2}) = A.$$

Since $X^n + X^{k_1} + X^{k_2} = 1 \mod (X^n + X^{k_1} + X^{k_2})$, we can then compute $A^{(n-1)} = A X^{-1}, A^{(n-2)} = A^{(n-1)} X^{-1}$, as shown below:

$$\vdots \quad A^{(n-2)} \quad A^{(n-1)} \quad A$$

$$\downarrow \quad \downarrow \quad \downarrow$$

$$a_2 \quad a_1 \quad a_0$$

$$\vdots \quad \vdots \quad \vdots$$

$$a_0 + a_1 + a_2 \quad a_{n-1} \quad a_{n-2}$$

$$\vdots \quad a_1 + a_{l_1+1} + a_{l_2+1} \quad a_0 + a_{l_1} + a_{l_2} \quad a_{n-1}$$

We easily remark that the matrix

$$T_A = [A^{(l_2)}, A^{(l_2+1)}, \ldots, A^{(n-1)}, A^{(0)}, A^{(1)}, \ldots, A^{(l_2-1)}]$$

is a Toeplitz matrix. In addition, the first $k_2$ columns (resp. the last $l_2$ columns) of $T_A$ are the last $k_2$ columns (resp. the first $n-k_2$ columns) of $M_A = [A^{(0)}, A^{(1)}, \ldots, A^{(n-1)}]$ from (3). Similarly we denote

$$\overline{B} = [b_{l_2}^1 \ b_{l_2+1}^1 \ \cdots \ b_{n-1}^1 \ b_0 \ b_1 \ \cdots \ b_{l_2-1}^1]$$

such that the first $k_2$ (resp. the last $n-k_2$) coefficients are equal to the last $k_2$ (resp. the first $n-k_2$) coefficients of $B$. In this situation, the coordinates in $B$ of $C = AB$ are given by

$$C = T_A \cdot \overline{B}. \quad (6)$$

5. Example

We use the irreducible NAOP $P = X^8 + X^7 + X^6 + X^3 + X^2 + X + 1$ to define the field $\mathbb{F}_{2^8}$. We construct the matrix $M_A$ of an element $A$ modulo $Q = P \times (X + 1) = X^9 + X^6 + X^3 + 1$. The basis $B$ of $\mathbb{F}_2[X]/(Q(X))$ is defined by the 9 elements

$$e_0 = 1, e_1 = X, e_2 = X^2, e_3 = X^3 + 1, e_4 = X^4 + X, e_5 = X^5 + X^2 + 1, e_6 = X^6 + X^3 + X, e_7 = X^7 + X^4 + X^2, e_8 = X^8 + X^5 + X^3.$$

In this situation, we have

$$l_1 = 3, \quad l_2 = 5, \quad k_1 = 6, \quad k_2 = 4.$$
6. Multiplier architecture

In this section, we present several multiplier architectures associated to the Toeplitz matrix expression in (6). Multiplier architectures can be classified into two types:
- **Parallel architecture.** Computations are done with no reuse of circuits and all the bits of the product $C = AB$ are output at each clock cycle.
- **Sequential architecture.** Computations are done with reuse of circuits and the result is obtained after $n$ clock cycles. There are mainly two types of sequential multipliers, one which output one bit per clock cycle, and the other which output all $n$ bits (in parallel) only at the end of $n$ clock cycles. In the paper the former will be referred to as sequential multiplier with bit serial output (SMOO) and the latter as sequential multiplier with bit parallel output (SMPO).

6.1. Parallel architecture

Here we present a parallel architecture associated to the double basis approach. This architecture is sketched in Figure 1. It consists of two preliminary parallel computations followed by a Toeplitz matrix vector product. The preliminary step computes the entries of the matrix $T_A$ from the coordinates of $A$ and in parallel performs the conversion of $B$ from $B$ to $B'$. For the former, we use the expression of the columns $A(i)$ given in Section 4. Let $a_j(i)$ denote the $j$-th entry of $A(i)$. Then we have

\[
\begin{align*}
    a_j(i) &= a_{n-i} + a_{2-i} + a_{1-i}, 1 \leq i \leq l_1, \\
    a_l(i) &= a_{i-1} + a_{n-i} + a_{2-i}, 1 < i < l_2, \\
    a_{n-i} &= a_i + a_{i-1} + a_{2-i} + a_{1-i}, 1 \leq i \leq k_2.
\end{align*}
\]

(7) (8) (9)

To compute the coordinates $b_j(i), i = 0, \ldots, n - 1$ we only need to apply the formula given in subsection 3.1. We evaluate the complexity of this architecture below:

- **Space complexity.** It includes the space (i.e., logic gates) needed for the precomputations as well as the matrix-vector product. Using (7), (8) and (9), we deduce that the precomputations require

\[
\left(\frac{2{l_2 - 1}}{2} + 2{k_2}\right) + \left(\frac{l_2 - l_1}{2}\right) \text{ XOR gates.}
\]

The computation of $T_A$ conversion of $B$

We already know the space complexity of the matrix vector product which is given in Table 3.

- **Time complexity.** The delay of the critical path is equal to the delay for the preliminary computations plus the delay of the Toeplitz matrix-vector product part. The critical path for the precomputations corresponds to the computation of Eq. (8) and the corresponding delay is equal to $3D_X$ to compute the coefficient of $T_A$.

**Remark 1:** The parallel architecture in Figure 1 has a small improvement in the complexity of the multiplier when $k_1 = k_2 + 1$. In this case, all the entries of $T_A$ are computed with (7) and (9). Consequently the space complexity required for the computation of the entries of $T_A$ is equal to $2(n - 1)$ XOR gates and the time delay is $2D_X$.

For a fixed field $\mathbb{F}_{2^n-1}$ we give in Table 4 the complexity results of the two-way and three-way approaches for the double basis multiplier. We also give the complexities of a two-way and three-way splitting multipliers based on [4] for almost irreducible trinomial $X^n - 1 + \delta + X^k + 1$, also called redundant trinomials in [3]. These redundant trinomials admit an irreducible factor of degree $(n-1)$ which defines the field $\mathbb{F}_{2^n-1}$. The authors of [1] proposed several algorithms to find almost irreducible trinomials with small $\delta$.

In Table 4 we also give the complexity of the multiplier of [4] for specific pentanomials and the complexity of the multiplier of Sunar [12]. The complexity results given in Table 4 are valid if $n$ (resp. $n - 1$, $n - 1 + \delta$) is a power of 2 or 3. Some combination of two-way and three-way approaches could also be used, which extend the use of TMVP multiplication to size $n = 2^i3^j$.

In this situation, we can see that trinominal with double basis approach gives a parallel multiplier with the same space complexity as the multiplier of Fan and Hasan for pentanomials, but with an improvement by $D_X$ ($2D_X$ for special quadrinomials) in the delay of the multiplier. The redundant trinomial is better when $\delta$ is small, otherwise...
when $\delta$ is quite big the space complexity is higher.

For a general $n$, there are several strategies to obtain a subquadratic multiplier using the Toeplitz matrix. Using a small example, below we present two possible approaches: the first one enlarges the dimension of the Toeplitz matrix to the nearest $2^3 3^i$ bigger than $(n - 1)$, the second approach decomposes the matrix in a number Toeplitz blocks of size $2^3 3^j$.

**Example 1:** here we study here subquadratic space complexity multipliers for $n \neq 2^3 3^i$. We do it for the field $\mathbb{F}_{243}$, first assuming double basis with quadrinomial and then redundant trinomials.

- **Quadrinomial approach.** We represent the field as a subring of $\mathbb{F}_2[X]/(X^{249} + X^7 + 1)$. In this situation, a multiplication of two elements is expressed as $T_A \cdot B$ where $T_A$ is a $237 \times 237$ matrix. Since 237 cannot be decomposed as power of 2 and 3, we expand the matrix $T_A$ to a $243 \times 243$ Toeplitz matrix $T_A'$.

**Figure 2. Redundant quadrinomial architecture for $\mathbb{F}_{243}$**

We also append 6 zeros to $\overline{B}$ to obtain $\overline{B}'$. The product of $A$ and $B$ is now given by the first 237 coefficients of $T_A' \cdot B'$ and the three-way splitting approach can be applied since $243 = 3^5$. The resulting multiplier has a delay of $17D_X + D_A$ and a space complexity of 36586 XOR and 7776 AND gates.

- **Almost irreducible trinomial approach.** We represent the field $\mathbb{F}_{243}$ as a subring of $\mathbb{F}_2[X]/(X^{249} + X^7 + 1)$. The polynomial $X^{249} + X^7 + 1$ is the smallest degree trinomial which has a prime factor of a degree 235 polynomial. Following [4], the multiplication of two elements $A$ and $B$ can be expressed as a Toeplitz matrix vector product $T_A \cdot B$. In order to use the three-way splitting approach we propose to split the matrix $T_A$ into three blocks (cf. Fig. 3).

**Figure 3. Redundant trinomial architecture for $\mathbb{F}_{243}$**

The computation $T_A \cdot B$ is decomposed in $T_0 \cdot B_0$ and $T_1 \cdot B_1$ and $T_2 \cdot B$. We can perform these three TMVPs in parallel. This approach is depicted in Fig. 3. The computation of $T_0 \cdot B$ is done using three-way split subquadratic multiplication. For both $T_1 \cdot B_1$ and $T_2 \cdot B$ we use AND gates in parallel followed by XOR gates arranged in binary tree fashion. Other multiplication strategies could be applied for example for $T_2 \cdot B$, by splitting $T_2$ in several $6 \times 6$ Toeplitz matrices, computing each product in parallel and add the results with a binary tree of XORs. But this would not be advantageous considering both space and time complexities since the size of the Toeplitz matrices is really small. The resulting space complexity of this architecture is equal to 39061 XOR and 10728 AND gates and the delay is $17D_X + D_A$.

In conclusion of this example, we point out that the construction of the subquadratic multiplier for degree $n \neq 2^3 3^i$ involves some decomposition or expansion of the Toeplitz

<table>
<thead>
<tr>
<th>Method</th>
<th>Splitting</th>
<th>Space Complexity</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>This paper</td>
<td>Two-way</td>
<td>$5.5n \log_2(n) - 3n + 0.5$</td>
<td>$D_A + (2 \log_3(n) + 3)DX$</td>
</tr>
<tr>
<td>Pentanomial [4]</td>
<td>Two-way</td>
<td>$5.5(n - 1) \log_2(n) - 3n + 4$</td>
<td>$D_A + (2 \log_3(n - 1) + 4)DX$</td>
</tr>
<tr>
<td>Redundant-trinomial [4], [1], [3]</td>
<td>Two-way</td>
<td>$5.5(n - 1 + \delta) \log_2(n) - 6 + k + 5$</td>
<td>$D_A + (2 \log_2(n - 1 + \delta) + 1)DX$</td>
</tr>
<tr>
<td>This paper</td>
<td>Three-way</td>
<td>$\frac{243}{10}n \log_2(n) - 2n + \frac{1}{2}$</td>
<td>$D_A + (3 \log_3(n + 3) + 2)DX$</td>
</tr>
<tr>
<td>Pentanomial [4]</td>
<td>Three-way</td>
<td>$\frac{243}{10}n \log_2(n) - 2(n - n) + \frac{7}{10}$</td>
<td>$D_A + (3 \log_3(n - 1) + 3)DX$</td>
</tr>
<tr>
<td>Redundant-trinomial [4], [1], [3]</td>
<td>Three-way</td>
<td>$\frac{243}{10}n \log_2(n) - 2(n - 1 + \delta) - \frac{4}{5}$</td>
<td>$D_A + (3 \log_3(n - 1 + \delta) + 1)DX$</td>
</tr>
<tr>
<td>Sunar [12]</td>
<td>Winograd</td>
<td>$(n - 1) \log_2(n) - 8n + 10$</td>
<td>$D_A + (2 \log_4(n - 1) + 1)DX$</td>
</tr>
<tr>
<td>Sunar [12]</td>
<td>Winograd</td>
<td>$(n - 1) \log_2(n) - \frac{243}{10}(n - 1) + 2$</td>
<td>$D_A + (4 \log_4(n - 1) + 1)DX$</td>
</tr>
</tbody>
</table>

Table 4. Complexity comparison for field $\mathbb{F}_{2^{243}}$
6.2. Sequential multiplier with bit serial output
(SMSO)

This multiplier is based on (6) where the i-th coordinate of C is obtained as the inner product of the i-th row of TA and the vector B. We remark that if we begin from the l2-th row of TA, we have

\[ R_{l_2} = [a_{n-1} \ a_{n-2} \ \ldots \ \ a_0], \]

and \( R_{l_2+1} \) is generated from \( R_{l_2} \) by performing a right shift and placing \( a_0 + a_{l_1} + a_{l_2} \) into the resulting empty position at the leftmost end. Then \( R_{l_2+2} \) is generated from \( R_{l_2+1} \) in the same way, and so on.

This process can be performed by applying right shifts to a bidirectional linear feedback shift register (bLFSR) as shown in the lower part of Figure 4. The bidirectional feature implies that the feedback shift register can be shifted to either left or right direction as desired.

Figure 5. Bi-directional storage cell and modulo two adder [5]

Hardware structure for such an bLFSR has been presented in [5]. Specifically, the implementation of the modulo two addition of the linear relation for both left and right shifts requires an XOR gate and some additional hardware which manages the change of direction of the output (see Fig 5 where \( L/L' \) determines the direction of data flow). The same is true for the storage cells. More details about bLFSR can be found in [5].

We remark also that if we begin with \( R_{l_2} \), we can shift the bLFSR contents to the left and place \( a_{n-1} + a_{l_1} + a_{l_2-1} \) into the right most cell to have \( R_{l_2-1} \). The row \( R_{l_2-2}, \ldots, R_0 \) can be computed in the same way. So using the left shift property of the bLFSR we can produce the first \( l_2 \) rows of \( TA \) and thus compute the corresponding coefficients of \( C \).

Consequently, we propose an original bit serial multiplier which uses these properties. This bit serial multiplier is depicted in Figure 4 and its operation has the following four steps.

1) Load the register with \( A \).
2) Apply right shifts to bLFSR to output the bits \( c_{l_2}, c_{l_2+1}, \ldots, c_{n-1} \).
3) Reload the register with \( A \).
4) Apply \( l_2 + 1 \) left shifts to bLFSR to output the bits \( c_{l_2}, c_{l_2-1}, \ldots, c_0 \).

We remark that the bit \( c_{l_2} \) is output twice, since the rows in the upper and the lower parts of \( TA \) are generated from the same row \( R_{l_2} \). One of these bits should be removed.

6.3. Sequential multiplier with bit parallel output
(SMPO)

We have seen in Section 4 that there are two recursive relations which allow us to generate the columns of \( TA \). By generating the columns \( A^{(j)} = A^{(j)}_{j} \) for \( j = 0, 1, \ldots, n - 1 \) with a weight of \( b_j \), we obtain \( C \) as in equation (6). Note that \( A^{(j)} \)’s are columns of \( TA \).

The expressions of \( A^{(j)} \) in terms of \( A^{(j-1)} \) or \( A^{(j+1)} \) given in Section 4 correspond to a left and right shifts of a bi-directional LFSR as described above to generate the rows of \( TA \). This leads to a multiplier with bit parallel output.
Figure 6. Sequential multiplier with bit parallel output

(Figure 6). The operation of the multiplier has the following steps:

1) Load the register with \( A \).
2) Apply \( k_2 \) right shifts to bLFSR.
3) Reload the register with \( A \) and apply a left shift with 0 as input in AND gates.
4) Apply \( l_2 \) left shifts to bLFSR to output the bits \( c_{n-1}, c_{n-2}, \ldots, c_0 \).

6.4. Comparison of SMSO and SMPO

Using Figures 4 and 6, we can easily evaluate the complexity of the two sequential multipliers. We can see that the two architectures have roughly the same number of XOR and AND gates. Compared to SMSO, SMPO requires twice more flip-flops but is faster by a factor of \( \log_2(n+1) \).

<table>
<thead>
<tr>
<th></th>
<th>SMSO</th>
<th>SMPO</th>
</tr>
</thead>
<tbody>
<tr>
<td>XOR</td>
<td>( n+1 )</td>
<td>( n+2 )</td>
</tr>
<tr>
<td>AND</td>
<td>( n )</td>
<td>( n )</td>
</tr>
<tr>
<td>Flip-Flop</td>
<td>( n ) cells</td>
<td>( 2n ) cells</td>
</tr>
<tr>
<td>Delay</td>
<td>((n+1)(D_A + \log_2(n+1)D_X))</td>
<td>((n+1)(D_A + D_X))</td>
</tr>
</tbody>
</table>

7. Conclusion

In this paper we have considered multipliers over binary fields defined by near all one polynomials. To this end, we have used multiplication modulo a quadrinomial. We have introduced a double basis approach which provides a Toeplitz matrix vector product expression for multiplication modulo the quadrinomial. This has resulted in a subquadratic space complexity parallel multiplier, which has lower delay than the Fan and Hasan multiplier modulo pentanomials [4]. We have also presented two sequential multipliers using a bidirectional LFSR.

References