On Complexity of Polynomial Basis Squaring in $\mathbb{F}_{2^m}$

Huapeng Wu

The Centre for Applied Cryptographic Research,
Department of Combinatorics and Optimization, University of Waterloo, Waterloo, Canada
h3wu@cacr.math.uwaterloo.ca

Abstract. In this paper, the complexity of a squaring operation using polynomial basis (PB) in a class of finite fields $\mathbb{F}_{2^m}$ is evaluated. The main results are as follows:

1. When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$, where both m and k are odd, a PB squaring operation requires $\frac{m-1}{2}$ bit operations.
2. When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$, where $m + k$ is odd and $k \neq \frac{m}{2}$, a PB squaring operation requires $\frac{m+k-1}{2}$ bit operations.
3. When the field is generated with an irreducible trinomial $f(x) = x^m + x^{\frac{m}{2}} + 1$, a PB squaring operation requires $\frac{m+2}{4}$ bit operations.

1 Introduction

Finite field arithmetic has recently been paid much attention mainly because its use in elliptic curve cryptography. In implementing an elliptic curve cryptosystem, a normal basis is usually utilized, because squaring operation in normal basis is only a cyclic shift of the element's coefficients. A multiplication operation can also be performed efficiently with an optimal normal basis (ONB) [5]. It has been shown that a bit-parallel multiplication in $\mathbb{F}_{2^m}$ can be done in about $2m^2$ ground field operations if a type-I ONB is chosen [2]. However, type-I ONB exists only in a small class of fields $\mathbb{F}_{2^m}$ where m is an even number. Moreover, it is more likely to have a comparatively efficient discrete elliptic curve logarithm when m is composite [4]. On the other hand, it has been shown that a bit-parallel multiplier using trinomial-based polynomial basis (TPB) has about the same complexity as that using a type-I ONB [3], while irreducible trinomial over $\mathbb{F}_{2^m}$ exists much more prevailingly than type-I ONB. A squaring operation in TPB, however, is not free.

In this short article, we derive the complexity of a bit-parallel squaring operation using a TPB in $\mathbb{F}_{2^m}$. It is shown to be of order $O(m)$ ground field operations (comparing to $2m^2$ ground field operation needed for a bit-parallel multiplication operation). If we try to solve an inverse in $\mathbb{F}_{2^m}$ using the method from Fermat theorem, then the complexity of $m-1$ bit-parallel squaring operations

D.R. Stinson and S. Tavares (Eds.): SAC 2000, LNCS 2012, pp. 118–129, 2001.

required is not greater than that of half bit-parallel multiplication operation. The time propagation of the hardware architecture of a bit-parallel squarer is also addressed.

The main results include: When the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$, then a PB squaring operation requires at most

1. $\frac{m-1}{2}$ bit addition, if both m and k are odd;
2. $\frac{m+k-1}{2}$ bit operations, if $m+k$ is odd and $k \neq \frac{m}{2}$.
3. $\frac{k+1}{2}$ bit operations, if $k = \frac{m}{2}$.

The organization of this paper is as follows: An brief introduction to PB squaring operation is given in Section 2. In Section 3, we present new complexity upper bound for PB squaring operation in a class of finite fields. Hardware bit-parallel implementation is addressed in Section 4. Finally, a few concluding remarks are given in Section 5.

2 Polynomial Basis Squaring Operation

Let $f(x)$ be the irreducible polynomial over $\mathbb{F}_2$ generating the field $\mathbb{F}_{2^m}$. Let $A(x) = \sum_{i=0}^{m-1} a_i x^i$ be the polynomial representation of an arbitrary element of $\mathbb{F}_{2^m}$. The squaring operation of $A(x)$ is

$$\begin{aligned} C(x) \triangleq \sum_{i=0}^{m-1} c_i x^i &= A^2(x) \bmod f(x) \\ &= a_0 + a_1 x^2 + a_2 x^4 + \ldots + a_{m-1} x^{2m-2} \bmod f(x). \end{aligned}$$

It can be seen that squaring in $\mathbb{F}_{2^m}$ is actually a case of polynomial modular reduction. Then the following corollary is obvious from the results on complexity of polynomial modular reduction [6].

Corollary 1. Let the field $\mathbb{F}_{2^m}$ be generated with the irreducible r-term polynomial $f(x)$ of degree m. Then squaring a field element in parallel can be performed with at most $(r-1)(m-1)$ addition operations in $\mathbb{F}_2$.

When $f(x)$ is chosen as an irreducible trinomial, however, the complexity can be further reduced.

3 Complexity Upper Bound for PB Squaring

In this section, we assume that the field is generated with an irreducible trinomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$. Based on the the parity of m and k, the derivation is divided into the following three cases:

1. Both m and $1 \leqslant k < \frac{m}{2}$ are odd;
2. m is odd and $1 < k < \frac{m}{2}$ is even;
3. m is even and $1 \leqslant k \leqslant \frac{m}{2}$ is odd.

3.1 Both m and $1 \leqslant k < \frac{m}{2}$ Are Odd

Let

$$A^2(x) = \sum_{i=0}^{m-1} a_i x^{2i} = \sum_{i=0}^{2m-2} a'_i x^i,$$

where $a'_i \triangleq a_{\frac{i}{2}}$ if i even, and 0 if i odd. Define

$$\sum_{i=0}^{m+2l+1} a'_i x^i \bmod f(x) \triangleq \sum_{i=0}^{m-1} t_i^{(l)} x^i,$$

for $l = -1, 0, 1, \ldots, \frac{m-1}{2} - 1$. Then we have

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x). \tag{1}$$

The coefficient $t_i^{(l)}$'s have their initial values $t_i^{(-1)} = a'_i$, and we try to solve the final values $t_i^{(\frac{m-1}{2}-1)} = c_i, i = 0, 1, \ldots, m-1$. Note that $t_i^{(-1)} = 0$ if i is an odd number.

When $l = 0$,

$$\begin{aligned}\sum_{i=0}^{m-1} t_i^{(0)} x^i &= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} x^{m+1} \bmod f(x)\\ &= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1}(x + x^{k+1}) \bmod f(x).\end{aligned}$$

Then we have

$$t_i^{(0)} = \begin{cases} a'_i + a'_{m+1}, & i = k+1;\\ a'_i, & i \text{ even, and } i \neq k+1;\\ a'_{m+1}, & i = 1;\\ 0, & \text{otherwise}.\end{cases}$$

Clearly, one bit addition is needed to compute $t_i^{(0)}$ from $t_i^{(-1)}$, $i = 0, 1, \ldots, m-1$.

In the following we will repeatedly use (1) for $l = 1, 2, \ldots, \frac{m}{2} - 1$. It will be seen that there are a few newly generated terms at each step. For example, when $l = 0$ we have two newly generated terms $a'_m x$ and $a'_m x^{k+1}$. Note that $k+1$ is an even number and one bit operation is needed to take care of this even power term. In fact, one bit addition is *always* required if an *even* power term is generated, while one bit operation is *probably* needed if an *odd* power term is generated. This is because for some l, $t_i^{(l-1)}$ could be zero for some odd i.

For $l > 0$ and $l \leqslant \frac{m-k}{2} - 1$ (in order to keep $k + 2l + 1 < m$), we have

$$\begin{aligned}
\sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x)\\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1}(1+x^k) \bmod f(x)\\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{k+2l+1} \bmod f(x).
\end{aligned}$$

Obviously in this step (from $l-1$ to l), one odd power term (x^{2l+1}) and one even power term (x^{k+2l+1}) are generated at the right side of the above equation. When l runs through from 0 to $\frac{m-k}{2}-1$, the value of $2l+1$ runs through the odd numbers from 1 to $m-k-1$, and the value of $k+2l+1$ runs through the even numbers from $k+1$ to $m-1$.

Therefore, when $0 \leqslant l \leqslant \frac{m-k}{2}-1$, we have

$$\begin{aligned}
t_i^{(l)} &= \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+k+1;\\ a'_{m+2l+1}, & i = 2l+1;\\ t_i^{(l-1)}, & i \text{ even and } i \neq 2l+k+1; \text{or } i = 1,3,\ldots,2l-1;\\ 0, & \text{otherwise.} \end{cases}\\
&= \begin{cases} t_i^{(l-1)} + a'_{m-k+i}, & i = 2l+k+1;\\ a'_{m+i}, & i = 2l+1;\\ t_i^{(l-1)}, & i \text{ even and } i \neq 2l+k+1; \text{or } i = 1,3,\ldots,2l-1;\\ 0, & \text{otherwise.} \end{cases}\\
&= \begin{cases} a'_i + a'_{m-k+i} & i = k+1, k+3, \ldots, k+2l+1;\\ a'_{m+i} & i = 1,3,\ldots,2l+1;\\ a'_i & i \text{ even and } i \neq k+1, k+3, \ldots, k+2l+1;\\ 0 & \text{Otherwise.} \end{cases}
\end{aligned}$$

Thus for $l = \frac{m-k}{2}-1$, we can solve $t_i^{(l)}$ as follows

$$t_i^{(\frac{m-k}{2}-1)} = \begin{cases} a'_i + a'_{m-k+i} & i = k+1, k+3, \ldots, m-1;\\ a'_{m+i} & i = 1,3,\ldots,m-k-1;\\ a'_i & i = 0,2,\ldots,k-1;\\ 0 & i = m-k+1, m-k+3, \ldots, m-2. \end{cases}$$

In the following, we consider two cases:

1. If $k=1$.
   When $k=1$, we have $\frac{m-k}{2}-1 = \frac{m-1}{2}-1$. Therefore,

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_i + a'_{m-1+i} & i = 2,4,\ldots,m-1;\\ a'_{m+i} & i = 1,3,\ldots,m-2;\\ a'_i & i = 0. \end{cases} \tag{2}$$

   It can be seen from the (2) that $\frac{m-1}{2}$ bit additions are required for obtaining $c_i,\ i = 0,1,\ldots,m-1$.

2. If $1 < k < \frac{m}{2}$.
When $\frac{m-k}{2} \leqslant l \leqslant \frac{m-1}{2} - 1$, we have

$$\begin{aligned}
\sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} [1 + x^k] \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} \\
&\quad + a'_{m+2l+1} x^{2l+k+1-m} [1 + x^k] \bmod f(x) \\
&= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m} \\
&\quad + a'_{m+2l+1} x^{2l+2k+1-m} \bmod f(x)
\end{aligned}$$

Since $k \leqslant \frac{m}{2}$ and $l \leqslant \frac{m-1}{2} - 1$, we have $2l + 2k + 1 - m \leqslant m - 2$. It can be seen that there are two newly generated odd power terms (x^{2l+1} and $x^{2l+k+1-m}$) and one even power term ($x^{2l+2k+1-m}$) in this step. When l runs through from $\frac{m-k}{2}$ to $\frac{m-1}{2} - 1$, the value of $2l + 1$ runs through the odd numbers from $m - k + 1$ to $m - 2$, the value of $2l + k + 1 - m$ runs through the odd numbers from 1 to $k - 2$, and the value of $2l + 2k + 1 - m$ runs through the even numbers from $k + 1$ to $2k - 2$.
Therefore, $t_i^{(l)}$ can be given as follows

$$t_i^{(l)} = \begin{cases} a'_{m+2l+1}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+2k+1-m, 2l+k+1-m; \\ t_i^{(l-1)}, & i \text{ even and } i \neq k+1, k+3, \ldots, 2l+2k-1-m; \\ & \text{or } i \text{ odd and } i = 1, 3, \ldots, 2l+k-1-m, \\ & 2l+k+3-m, 2l+k+5-m, \ldots, 2l-1; \\ 0, & \text{otherwise.} \end{cases}$$

$$= \begin{cases} a'_{m+i}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l+k+1-m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l+2k+1-m; \\ t_i^{(l-1)}, & i \text{ even and } i \neq k+1, k+3, \ldots, 2l+2k-1-m; \\ & \text{or } i \text{ odd and } i = 1, 3, \ldots, 2l+k-1-m, \\ & 2l+k+3-m, 2l+k+5-m, \ldots, 2l-1; \\ 0, & \text{otherwise.} \end{cases}$$

$$= \begin{cases} a'_{m+i} & i = 2l+k+3-m, 2l+k+5 \\ & \quad -m, \dots, 2l+1; \\ a'_{m+i} + a'_{2m-k+i} & i = 1, 3, \dots, 2l+k+1-m; \\ a'_i + a'_{m-k+i} + a'_{2m-2k+i} & i = k+1, k+3, \dots, 2l+2k+1-m; \\ a'_i + a'_{m-k+i} & i = 2l+2k+3-m, 2l+2k+5 \\ & \quad -m, \dots, m-1; \\ a'_i & i = 0, 2, \dots, k-1; \\ 0 & i = 2l+3, 2l+5, \dots, m-2. \end{cases}$$

When $l = \frac{m-1}{2} - 1$, it follows from the above equations

$$c_i = t_i^{(\frac{m-3}{2})} = \begin{cases} a'_{m+i} & i = k, k+2, \dots, m-2; \\ a'_{m+i} + a'_{2m-k+i} & i = 1, 3, \dots, k-2; \\ a'_i + a'_{m-k+i} + a'_{2m-2k+i} & i = k+1, k+3, \dots, 2k-2; \\ a'_i + a'_{m-k+i} & i = 2k, 2k+2, \dots, m-1; \\ a'_i & i = 0, 2, \dots, k-1. \end{cases}$$

Rewrite the above equation as the following

$$\begin{aligned} c_i &= a'_i & i &= 0, 2, \dots, k-1; & \text{(3a)} \\ c_i &= (a'_{m+i} + a'_{2m-k+i}) & i &= 1, 3, \dots, k-2; & \text{(3b)} \\ c_{k+i} &= a'_{m+k+i} & i &= 0, 2, \dots, m-k-2; & \text{(3c)} \\ c_{k+i} &= a'_{k+i} + (a'_{m+i} + a'_{2m-k+i}) & i &= 1, 3, \dots, k-2; & \text{(3d)} \\ c_{2k+i} &= a'_{2k+i} + a'_{m+k+i} & i &= 0, 2, \dots, m-2k-1; & \text{(3e)} \end{aligned}$$

Then it can be seen from (3b) and (3d) that some partial sums can be reused (indicated with the bracket). This will save $\frac{k-1}{2}$ bit operations. The total number of bit operations required for the squaring operation can be counted from (3a-3e) and it is $\frac{m-1}{2}$.

3.2 m Is Odd and $1 < k < \frac{m}{2}$ Is Even

The definitions of a'_i and $t_i^{(l)}$ are the same as these in the last subsection. We rewrite the equation (1) here for convenience.

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x).$$

The terms $t_i^{(l)}$'s have their initial values $t_i^{(-1)} = a'_i$, and we try to solve the final values $t_i^{(\frac{m-1}{2}-1)} = c_i, i = 0, 1, \dots, m-1$.

When $l = 0$,

$$\sum_{i=0}^{m-1} t_i^{(0)} x^i = \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1} x^{m+1} \bmod f(x)$$

$$= \sum_{i=0}^{m-1} a'_i x^i + a'_{m+1}(x + x^{k+1}) \bmod f(x).$$

It follows

$$t_i^{(0)} = \begin{cases} a'_{m+1}, & i = 1, k+1; \\ a'_i, & i \text{ even}; \\ 0, & i \text{ odd and } i \neq 1, k+1; \end{cases}$$

Since both the newly generated terms are odd power ones, no bit addition is needed to obtain $t_i^{(0)}$ from $t_i^{(-1)}$, $i = 0, 1, \ldots, m-1$.

For $l \geqslant 0$, we have

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1}(1 + x^k) \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{k+2l+1}. \end{aligned}$$

It can be seen that two odd power terms are generated at the right side of the above equation.

When l runs through from 0 to $\frac{k}{2} - 1$, the value of $2l + 1$ runs through the odd numbers from 1 to $k - 1$, and the value of $k + 2l + 1$ runs through the odd numbers from $k + 1$ to $2k - 1$. Note that $2k - 1 < m - 1$.

Then we have

$$\begin{aligned} t_i^{(l)} &= \begin{cases} a'_{m+2l+1}, & i = 2l+1, k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i \neq 1, 3, \ldots, 2l+1, k+1, k+3, \ldots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} a'_{m+i}, & i = 2l+1; \\ a'_{m-k+i}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i \neq 1, 3, \ldots, 2l+1, k+1, k+3, \ldots, k+2l-1; \\ 0, & \text{otherwise.} \end{cases} \\ &= \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, 2l+1; \\ a'_{m-k+i}, & i = k+1, k+3, \ldots, k+2l+1; \\ a'_i, & i = 0, 2, \ldots, m-1; \\ 0, & \text{otherwise.} \end{cases} \end{aligned} \tag{4}$$

When $l = \frac{k}{2} - 1$, from the equation (4) we have

$$t_i^{(\frac{k}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m-k+i}, & i = k+1, k+3, \ldots, 2k-1; \\ a'_i, & i = 0, 2, \ldots, m-1; \\ 0, & \text{otherwise}. \end{cases} \quad (5)$$

In the following we consider two cases:

1. If $2k < m - 1$.
   In this case we have $k \leqslant m-k-3$. When l runs through from $\frac{k}{2}$ to $\dfrac{m-k-3}{2}$ (in order to satisfy $k + 2l + 1 < m - 1$), the value of $2l + 1$ runs through the odd numbers from $k + 1$ to $m - k - 2$, and the value of $k + 2l + 1$ runs through the odd numbers from $2k + 1$ to $m - 2$.
   From (4) and since $2l + 1 \geqslant k + 1$, we have

$$t_i^{(l)} = \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1; \\ a'_{m+2l+1}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i = 1, 3, \ldots, 2l-1, 2l+3, \ldots, k+2l-1; \\ 0, & \text{otherwise}. \end{cases}$$

$$= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ a'_{m-k+i}, & i = k+2l+1; \\ t_i^{(l-1)}, & i \text{ even, or } i \text{ odd and} \\ & i = 1, 3, \ldots, 2l-1, 2l+3, \ldots, k+2l-1; \\ 0, & \text{otherwise}. \end{cases}$$

$$= \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \ldots, k+2l+1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \ldots, 2l+1; \\ a'_i, & i = 0, 2, \ldots, m-1; \\ 0, & \text{otherwise}. \end{cases}$$

   When $l = \dfrac{m-k-3}{2}$, it follows

$$t_i^{(\frac{m-k-3}{2})} = \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \ldots, m-k-2; \\ a'_{m-k+i}, & i = m-k, m-k+2, \ldots, m-2; \\ a'_i, & i = 0, 2, \ldots, m-1; \\ 0, & \text{otherwise}. \end{cases}$$

   When $\dfrac{m-k-1}{2} \leqslant l \leqslant \dfrac{m-1}{2} - 1$, we have

$$\begin{aligned} \sum_{i=0}^{m-1} t_i^{(l)} x^i &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a_{m+2l+1} x^{m+2l+1} \\ &= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \end{aligned}$$

$$= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m}$$
$$+ a'_{m+2l+1} x^{2l+2k+1-m}$$

When l runs through from $\frac{m-k-1}{2}$ to $\frac{m-1}{2}-1$, the value of $2l+1$ runs through from the odd numbers $m-k$ to $m-2$, the value of $2l+k+1-m$ runs through the even numbers from 0 to $k-2$, and the value of $2l+2k+1-m$ runs through the even numbers from k to $2k-2$.
Therefore, we have

$$t_i^{(l)} = \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1, 2l+k+1-m, 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases}$$
$$= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l+k+1-m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases}$$
$$= \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \ldots, 2l+1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \ldots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \ldots, 2l+k+1-m; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \ldots, 2l+2k+1-m; \\ a'_i, & \text{otherwise.} \end{cases}$$

Then we can solve the final values for this case:

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \ldots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \ldots, k-2; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \ldots, 2k-2; \\ a'_i, & i = 2k, 2k+2, \ldots, m-1. \end{cases} \quad (6)$$

From the above equation we conclude that the total cost for computing squaring operation for this case is $\frac{m+k-1}{2}$ bit addition. The longest time delay to compute a c_i is the time taking to finish one bit addition.

2. If $2k = m-1$.
   In this case we have $2k-1 = m-2$. Thus from (5) it follows

$$t_i^{(\frac{k}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \ldots, k-1; \\ a'_{m-k+i}, & i = k+1, k+3, \ldots, m-2; \\ a'_i, & i = 0, 2, \ldots, m-1. \end{cases}$$

   Then for $\frac{k}{2} \leqslant l \leqslant \frac{m-1}{2} - 1$, we have

$$\sum_{i=0}^{m-1} t_i^{(l)} x^i = \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{m+2l+1} \bmod f(x)$$

$$= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i + a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1} \bmod f(x)$$
$$= \sum_{i=0}^{m-1} t_i^{(l-1)} x^i a'_{m+2l+1} x^{2l+1} + a'_{m+2l+1} x^{2l+k+1-m}$$
$$+ a'_{m+2l+1} x^{2l+2k+1-m} \bmod f(x)$$

When l runs through from $\frac{k}{2}$ to $\frac{m-1}{2} - 1$, the value of $2l+1$ runs through from the odd numbers $k+1$ to $m-2$, the value of $2l+k+1-m$ runs through the even numbers from 0 to $k-2$, and the value of $2l+2k+1-m$ runs through the even numbers from k to $2k-2$.
Therefore, we have

$$t_i^{(l)} = \begin{cases} t_i^{(l-1)} + a'_{m+2l+1}, & i = 2l+1, 2l+k+1-m, 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases}$$
$$= \begin{cases} t_i^{(l-1)} + a'_{m+i}, & i = 2l+1; \\ t_i^{(l-1)} + a'_{2m-k+i}, & i = 2l+k+1-m; \\ t_i^{(l-1)} + a'_{2m-2k+i}, & i = 2l+2k+1-m; \\ t_i^{(l-1)}, & \text{otherwise.} \end{cases}$$
$$= \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \dots, 2l+1; \\ a'_{m-k+i}, & i = 2l+3, 2l+5, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, 2l+k+1-m; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2l+2k+1-m; \\ a'_i, & \text{otherwise.} \end{cases}$$

Then we can solve the final values for this case:

$$c_i = t_i^{(\frac{m-1}{2}-1)} = \begin{cases} a'_{m+i}, & i = 1, 3, \dots, k-1; \\ a'_{m+i} + a'_{m-k+i} & i = k+1, k+3, \dots, m-2; \\ a'_i + a'_{2m-k+i}, & i = 0, 2, \dots, k-2; \\ a'_i + a'_{2m-2k+i}, & i = k, k+2, \dots, 2k-2; \\ a'_i, & i = 2k, 2k+2, \dots, m-1. \end{cases} \quad (7)$$

From the above equation it is clear that the total cost for computing squaring operation for this case is also $\frac{m+k-1}{2}$ bit addition.

3.3 m Is Even and $1 \leqslant k \leqslant \frac{m}{2}$ Is Odd

When the field is generated with an irreducible trinomial of form $f(x) = x^m + x^k + 1$, where m is even and $k \leqslant \frac{m}{2}$ is odd, similar analysis can be applied. In this case the complexity for a PB squaring operation in $\mathbb{F}_{2^m}$ is $\frac{m+k-1}{2}$ bit additions if $k < \frac{m}{2}$, and $\frac{k+1}{2}$ bit additions if $k = \frac{m}{2}$ [6].

We summarize the results obtained from the three cases in this section in the following theorem:

Theorem 1. If there is an irreducible polynomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$ over $\mathbb{F}_2$, then a squaring operation in $\mathbb{F}_{2^m}$ can be performed in

(i) $\frac{m-1}{2}$ bit additions, if both m and k are odd.
(ii) $\frac{m+k-1}{2}$ bit additions, if $m+k$ is odd and $k \neq \frac{m}{2}$.
(iii) $\frac{k+1}{2}$ bit additions, if $k = \frac{m}{2}$.

4 Bit-Parallel Implementation

In hardware implementation, a bit addition in $\mathbb{F}_2$ can be realized using an XOR gate. If we denote the time propagation delay of an XOR gate by T_X, then the time delay of a hardware architecture can be measured in terms of gate delays.

For example, from (3a-3d) it can be seen that the most bit operations taken to compute a c_i are when $i = 1, 3, \ldots, k-2$, as it is shown in (3d). Thus in this case the longest time propagation delay in a bit-parallel architecture for squaring is $2T_X$. The time delay for the other cases can be obtained from (2), (6), and (7) in a similar way.

The results on the complexity for a bit-parallel implementation of squaring operation are summarized as follows:

Theorem 2. If there is an irreducible polynomial $f(x) = x^m + x^k + 1,\ 1 \leqslant k \leqslant \frac{m}{2}$ over $\mathbb{F}_2$, then a bit-parallel hardware implementation of squaring operation in $\mathbb{F}_{2^m}$ can be constructed with

(i) $\frac{m-1}{2}$ XOR gates and the incurred time delay is $2T_X$, if both m and $k > 1$ are odd;
(ii) $\frac{m-1}{2}$ XOR gates and the incurred time delay is T_X, if m is odd and $k = 1$;
(iii) $\frac{m+k-1}{2}$ XOR gates and the incurred time delay is T_X, if m is odd and k is even;
(iv) $\frac{m+k-1}{2}$ XOR gates and the incurred time delay is $2T_X$, if m is even and $1 < k < \frac{m}{2}$ is odd.
(v) $\frac{m}{2}$ XOR gates and the incurred time delay is T_X, if m is even and $k = 1$.
(vi) $\frac{k+1}{2}$ XOR gates and the incurred time delay is T_X, if m is even and $k = \frac{m}{2}$.

5 Concluding Remarks

Squaring operation is frequently required in elliptic curve cryptographic systems when an inversion or a point multiple operation is performed. Normal basis has been widely used because squaring operation using normal basis is only a cyclic shift of the coefficients. However, normal basis multiplication can be performed efficiently only when there is an optimal normal basis [5]. The results in this paper have shown that the complexity of a PB squaring operation is very low, comparing to that of a multiplication operation ($O(m^2)$). This fact suggests that polynomial basis might be a good replacement for normal basis in many cryptographic application, since the prevailing existence of irreducible trinomial [1], comparing to that of optimal normal basis.

References

1. Blake, I.F., Gao, S., Lambert, R.: Constructive Problems for Irreducible Polynomials over Finite Fields. Canadian Workshop on IT, Springer-Verlag, 1993
2. Hasan, M. A., Wang, M., Bhargava, V. K.: A modified Massey-Omura parallel multiplier for a class of finite fields. IEEE Trans. Comput. **42** (1993) 1278-1280
3. Sunar, B., Koc, C. K.:Mastrovito Multiplier for all trinomials. IEEE Trans. Comput. **48** (1999) 522-527
4. Menezes, A.: Private communication. March, 2000.
5. Mullin, R., Onyszchuk, I., Vanstone, S.A., Wilson, R.: Optimal normal bases in GF(p^n). Disc. Appl. Math. **22** (1988) 149-161
6. Wu, H.: Efficient Computations in Finite Fields with Cryptographic Significance. Ph.D Thesis, University of Waterloo, Waterloo, Canada, 1998