Fast Squaring in TypeI All One Polynomial Field

Hidehiro Kato¹, Yasuyuki Nogami¹ and Yoshitaka Morikawa¹
¹Graduate School of Natural Science and Technology, Okayama University
Tsushima-naka 3-1-1, Okayama 700-8530, Japan
E-mail : {kato, nogami, morikawa}@trans.cne.okayama-u.ac.jp

Abstract: In this paper, the authors propose a fast squaring on all one polynomial field and show some simulation results with characteristic \( p = 256 \)-bit prime and extension degree \( k \in \{2, 4, 6, 10, 12\} \) implemented on a PentiumIV (2.6GHz) using the C programming language. The computer simulations showed that, on average, the proposed algorithm accelerated about 30% in \( \mathbb{F}_{p^2} \), 12% in \( \mathbb{F}_{p^4} \) and 6% in \( \mathbb{F}_{p^6} \) compared to the conventional squaring.

1. Introduction

In recent years, bilinear pairings have been used to construct new cryptographic schemes with many novel and exciting properties, such as group signature and identity-based encryption scheme. The pairing computation is generally heavy. Thus, implementing the pairing based protocols, there is a trend toward to reduce a number of pairings in recent cryptographic protocols. If pairing calculation becomes fast, it will give us a lot of options for implementation of cryptographic protocols. In order to efficiently compute pairings, various techniques are proposed such as Tate, Ate, twisted-Ate and Cross-twisted Ate pairings [1]-[4]. These pairings use extension fields as definition field. Pairing computation consist of two steps. One is Miller’s algorithm and the other is final exponentiation. These computations need a lot of squaring in the concerned extension field. Thus, a squaring in the extension field plays a key role for cryptographic applications.

Throughout this paper, \( p \) and \( m \) denote characteristic and extension degree, respectively, where \( p \) is a prime number. \( \mathbb{F}_p \) denotes a prime field with characteristic \( p \) and \( \mathbb{F}_{p^m} \) denotes an \( m \)-th extension field over \( \mathbb{F}_p \). \#SADD and \#SMUL denote the number of additions and the number of multiplications, respectively. A subtraction in \( \mathbb{F}_p \) is counted up as an addition in \( \mathbb{F}_p \). Without any additional explanation, lower and upper case letters show elements in prime field and extension field, respectively, and a Greek character shows a zero of modular polynomial.

2. Fundamental

In this section, let us briefly go over TypeI all one polynomial field (AOPF) and cyclic vector multiplication algorithm (CVMA)[5].

2.1 TypeI All One Polynomial Field

It is well-known that the following pseudo polynomial basis in \( \mathbb{F}_{p^m} \) is efficient for multiplication and Frobenius mapping[5].

\[ \{ \omega, \omega^2, \omega^3, \ldots, \omega^m \}, \]

where \( \omega \) is a zero of \( (x^{m+1} - 1)/(x - 1) \). \( \square \)

In this paper we deal with the extension field \( \mathbb{F}_{p^m} \) called all one polynomial field (AOPF) that use the normal basis of Eq.(1). According to the condition 1 of Theo.1, extension degree \( m \) is even.

2.2 Cyclic Vector Multiplication Algorithm

Cyclic vector multiplication algorithm (CVMA) efficiently calculates the product of two elements represented with the normal basis of Eq.(1).
Let us consider $A$ and $B$ in AOPF $\mathbb{F}_{p^m}$ as

$$A = \sum_{i=1}^{m} a_i \omega^i = (a_1, a_2, \ldots, a_m),$$

$$B = \sum_{i=1}^{m} b_i \omega^i = (b_1, b_2, \ldots, b_m).$$

Let $C$ be the product of $A$ and $B$ as

$$C = AB = (c_1, c_2, \ldots, c_m).$$

CVMA calculates the coefficient $c_i$ ($1 \leq i \leq m$) by

$$q_k = \sum_{i=1}^{m/2} \left( a_{\left\lfloor \frac{k}{2} + i \right\rfloor} - a_{\left\lfloor \frac{k}{2} - i \right\rfloor} \right) \left( b_{\left\lfloor \frac{k}{2} + i \right\rfloor} - b_{\left\lfloor \frac{k}{2} - i \right\rfloor} \right),$$

$$c_j = q_0 - q_j,$$

where $0 \leq k \leq m$ and the subscript $\lfloor \cdot \rfloor$ denotes $\cdot \mod m + 1$. The calculation cost of CVMA is given by

$$\#\text{SMUL} = \frac{m(m+1)}{2}, \quad \#\text{SADD} = \frac{3m^2}{2} - m - 1. \quad (5)$$

Squaring $C = A^2$ is calculated as

$$q_k = \sum_{i=1}^{m/2} \left( a_{\left\lfloor \frac{k}{2} + i \right\rfloor} - a_{\left\lfloor \frac{k}{2} - i \right\rfloor} \right)^2,$$

$$c_j = q_0 - q_j.$$

The calculation cost is given by

$$\#\text{SMUL} = \frac{m(m+1)}{2}, \quad \#\text{SADD} = m^2 - 1. \quad (7)$$

A simple approach for squaring cannot reduce the number of multiplication in $\mathbb{F}_p$. From Eqs.(6), squaring with CVMA finally carries out the subtraction of square elements $(a_{\left\lfloor \frac{k}{2} + i \right\rfloor} - a_{\left\lfloor \frac{k}{2} - i \right\rfloor})^2$. Using this property, this paper proposes the efficient squaring.

3. Main Proposal

This section proposes a fast squaring method based on CVMA. Let $A$ and $B$ be non-zero elements in extension field $\mathbb{F}_{p^m}$. In order to calculate $A^2 - B^2$ with direct computation, it needs two squarings in $\mathbb{F}_{p^m}$. This formula can be factorized as

$$A^2 - B^2 = (A + B)(A - B). \quad (8)$$

The product needs only one multiplication in $\mathbb{F}_{p^m}$. This paper shows that squaring with CVMA can efficiently carried out by applying this technique. The well-known other multiplication methods such as Karatsuba multiplication [6], toom-cook multiplication [7] and Montgomery [8] multiplication not have such a property.

3.1 Fast Squaring with CVMA

As previously introduced, a squaring $C = A^2$ with CVMA is shown in Eq.(6). Let $X_{i,k}$ be a

$$X_{i,k} = a_{\left\lfloor \frac{k}{2} + i \right\rfloor} - a_{\left\lfloor \frac{k}{2} - i \right\rfloor},$$

CVMA calculates

$$c_k = \sum_{i=1}^{m/2} X_{i,0}^2 - \sum_{i=1}^{m/2} X_{i,k}^2,$$

where the first and second terms in Eq.(10) are $q_0$ and $q_j$, respectively. Based on the idea described in Sec.3, $c_j$ are written as

$$c_k = \sum_{i=1}^{m/2} \left( X_{i,0}^2 - X_{i,k}^2 \right),$$

$$= \sum_{i=1}^{m/2} \left( X_{i,0} + X_{i,k} \right) \left( X_{i,0} - X_{i,k} \right). \quad (12)$$

According to Eq.(12), the proposed squaring with CVMA needs the following calculation costs.

$$\#\text{SMUL} = \frac{m^2}{2}, \quad \#\text{SADD} = 2m^2 - \frac{3m}{2}. \quad (13)$$

Compared with the cost of the original CVMA in Eq.(7), the proposed squaring method reduces the total amount of multiplications in $\mathbb{F}_p$.

4. Comparison

In order to evaluate the calculation cost for conventional and proposed squaring with CVMA, the authors considered extension field $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^4}$, $\mathbb{F}_{p^6}$, $\mathbb{F}_{p^{10}}$, and $\mathbb{F}_{p^{12}}$. Table 1 shows the calculation cost for squaring with CVMA. The proposed squaring reduced the total amount of multiplications in $\mathbb{F}_p$.

Table 2 shows the computational timings for conventional and proposed squarings on PentiumIV (2.6GHz), C language and GNU MP library. $p$ and $m$ are 256-bit prime number and
{2, 4, 6, 10, 12}, respectively. From Table 2, the proposed squaring in $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^4}$ and $\mathbb{F}_{p^6}$ are faster than conventional method about $30\%$, $12\%$ and $6\%$, respectively. The proposed squaring decreases a total amount of multiplications in $\mathbb{F}_{p^n}$, but more increases a total amount of multiplications. As a result, our proposed squaring method is more efficient the case of small extension degree.

Table 1. Calculation cost for squaring in $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^4}$, $\mathbb{F}_{p^6}$, $\mathbb{F}_{p^{10}}$ and $\mathbb{F}_{p^{12}}$.

<table>
<thead>
<tr>
<th>Field</th>
<th>conv. squaring</th>
<th>prop. squaring</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_{p^2}$</td>
<td>(3,3)</td>
<td>(2,5)</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^4}$</td>
<td>(10,15)</td>
<td>(8,26)</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^6}$</td>
<td>(21,35)</td>
<td>(18,63)</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^{10}}$</td>
<td>(55,99)</td>
<td>(50,185)</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^{12}}$</td>
<td>(78,143)</td>
<td>(72,270)</td>
</tr>
</tbody>
</table>

*(†, ‡) means (#SMUL, #SADD).

Table 2. Timings for squaring in $\mathbb{F}_{p^2}$, $\mathbb{F}_{p^4}$, $\mathbb{F}_{p^6}$, $\mathbb{F}_{p^{10}}$ and $\mathbb{F}_{p^{12}}$ (μsec).

<table>
<thead>
<tr>
<th>Field</th>
<th>conv. squaring</th>
<th>prop. squaring</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbb{F}_{p^2}$</td>
<td>3.19</td>
<td>2.18</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^4}$</td>
<td>10.7</td>
<td>9.36</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^6}$</td>
<td>22.4</td>
<td>21.2</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^{10}}$</td>
<td>59.5</td>
<td>64.3</td>
</tr>
<tr>
<td>$\mathbb{F}_{p^{12}}$</td>
<td>85.4</td>
<td>95.1</td>
</tr>
</tbody>
</table>

5. conclusion

In this paper, the authors proposed a fast squaring method on all one polynomial field. The conventional and proposed squaring methods with $p = 256$-bit prime and $m \in \{2, 4, 6, 10, 12\}$ were implemented on a PentiumIV (2.6GHz) using the C programming language. The simulation showed that, on average, the proposed algorithm accelerated about $30\%$ in $\mathbb{F}_{p^2}$, $12\%$ in $\mathbb{F}_{p^4}$ and $6\%$ in $\mathbb{F}_{p^6}$ compared to the conventional squaring. As a result, our proposed squaring method is more efficient the case of small extension degree.

6. Acknowledgment

This work was supported by the Strategic Information and Communications R&D Promotion Pro-

---

References


