International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:9, 2009

A Novel Method for Elliptic Curve Multi-Scalar Multiplication

Raveen R. Goundar, Ken-ichi Shiota, and Masahiko Toyonaga

***Abstract*—The major building block of most elliptic curve cryptosystems are computation of multi-scalar multiplication. This paper proposes a novel algorithm for simultaneous multi-scalar multiplication, that is by employing addition chains. The previously known methods utilizes double-and-add algorithm with binary representations. In order to accomplish our purpose, an efficient empirical method for finding addition chains for multi-exponents has been proposed.**

***Keywords*—elliptic curve cryptosystems, multi-scalar multiplication, addition chains, Fibonacci sequence.**

I. Introduction

Multi-scalar multiplication is required in many elliptic curve cryptosystems (ECC) such as provable-secure digital signatures [11], [12], multi-party protocols [2] and protocols of Brands [3]. It is given by the formula $\sum_{i=1}^{t} k_i G_i$ where k_i is a scalar variable (exponent), G_i shows a rational point (base) on an elliptic curve and i is an integer in $[1, t]$ where $t \geq 2$.

In most cases where multi-scalar multiplication is applied, the process in dominant in determining the overall efficiency. Hence, efficiency of multi-scalar multiplication are essential in elliptic curve cryptosystems. Conventional methods for computation of multi-scalar multiplication can be classified into two types. In methods of one type includes independent computation of the scalar multiples $k_i G_i$, followed by their addition. Such method could be very expansive but in cases where some of the scalars are fixed then a comb method [8] combined with a window method could enhance the overall efficiency of the process. In the methods of the other type, the multi-scalar multiplication is computed in one stage, without separate computation of $k_i G_i$. This includes simultaneous methods such as Shamir [4] and Interleave [9] method which utilizes binary representations for double-and-add algorithm.

In this paper we propose a novel algorithm for simultaneous multi-scalar multiplication that is, by utilizing addition chains. Hence, to accomplish our purpose, we propose an efficient empirical method for finding short addition chains for multi-exponents.

II. Background

In this section, we give a brief overview on elliptic curve cryptography, addition chains and Fibonacci sequence.

R.R.Goundar is with Department of Computing and Mathematics, Fiji Institute of Technology, Suva, Fiji, email: goundar_rr@fit.ac.fj

K.Shiota and M.Toyonaga is with Graduate School of Mathematics and Information Science, Kochi University, Japan, email: {shiota, toyonaga}@is.kochi-u.ac.jp

A. Elliptic Curve Cryptography

Let $\mathbb{F}_p$ be a finite field, where $p > 3$ is prime. Let E be an elliptic curve over $\mathbb{F}_p$. The elliptic curve can be used to construct an abelian group $E(\mathbb{F}_p)$ with identity element $\mathcal{O}$ called the point of infinity. A point $P \in E(\mathbb{F}_p)$ in affine coordinates is represented as $P = (x, y)$ where its inverse $-P = (x, -y)$ can be computed virtually for free. The elliptic curve addition operation $P+Q$ and doubling operation $2P$ are denoted by ADD and DBL, respectively, where $P, Q \in E(\mathbb{F}_p)$. More details could be cited from [5], [10].

B. Review on Addition Chains and Fibonacci Sequence

The use of moderately short addition chains can result in an efficient multi-scalar multiplication algorithm. However finding the shortest addition chain is known to be an NP-complete problem [5]. Conventionally, utilization of addition chains are considered to be cheaper for the cases of fixed exponent and variable bases [10], [7], since it is exponent dependent. However, if efficient algorithms for generating short addition chains are available then one may also consider for the cases of variable exponents and fixed bases.

Different types of addition chains and efficient methods for finding short addition chains are discussed in [1], [10]. The following defines an addition chain.

Definition 1. *An addition chain computing an integer k is given by two sequences $c = (c_0, \ldots, c_\ell)$ and $d = (d_1, \ldots, d_\ell)$ such that $c_0 = 1$, $c_\ell = k$, $c_i = c_r + c_s$, for all $1 \leq i \leq \ell$ with respect to $d_i = (r, s)$ and $0 \leq r, s \leq i-1$. The length of the addition chain is ℓ.*

Note that if the construction of addition chain involves fixed pattern then representations could be used during exponentiation instead of the index d_i.

Definition 2. *The Fibonacci sequence is defined as $F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$ where $F_0 = 0$ and $F_1 = 1$.*

The Fibonacci sequence has many properties [6], [13] but we recall only one here, by stating the following Binet's Formula.

Theorem 1. *Binet's Formula:*

$$F_n = \frac{\phi^n - (1-\phi)^n}{\sqrt{5}}, \quad \forall n \in \mathbb{N},$$

where $\phi = \frac{1+\sqrt{5}}{2}$ is the positive root of the real polynomial $X^2 - X - 1$.

From the above theorem, it is easy to deduce the following classical result.

$$\lim_{n\to\infty} \frac{F_n}{F_{n-1}} = \phi \,, \tag{1}$$

where ϕ is a golden ratio, also called a golden section.

III. Simultaneous Addition Chain for Multi-Exponents

In this section, we propose an algorithm for finding simultaneous addition chain for multi-exponents.

A. Strategy for Simultaneous Addition Chain for Multi-Exponents

Here, we discuss an efficient empirical method for the construction of simultaneous addition chain for multi-exponents, considering the case of dimension 2. We term it as simultaneous golden ratio addition chain method or SGRAC method in short.

The SGRAC method constructs chain starting from the last term, that is the input exponents u and v. We pair the two exponents with variables x and y to distinguish it from each other. Hence, we let $w_i = u_i x + v_i y$ in general. Our aim is to follow a Fibonacci pattern using the fact from equation (1). Hence, we try to maintain a near golden ratio value between succeeding terms. We begin by letting

$$\begin{aligned} w_0 &= ux + vy \,, \\ w_1 &= [w_0 \times \phi^{-1}] \,, \\ w_i &= w_{i-2} - w_{i-1} \quad \text{for} \quad i = 2, 3, \ldots \end{aligned} \tag{2}$$

Here w_i denotes the reverse of c_i that is, $w_i = c_{\ell - i}$. If continued with the procedure (2), w_i will exponentially deviate from $(w_{i-1} \times \phi^{-1})$ as i increases. In order to overcome this problem, a parameter MAXIMALGAP is introduced, where MAXIMALGAP $= u_{MG}x + v_{MG}y$. Hence, the above procedure (2) terminates whenever

$$|w_i - (w_{i-1} \times \phi^{-1})| > u_{MG}x + v_{MG}y \ \text{ or } w_i \leqslant \frac{w_{i-1}}{2} \,.$$

Note that the above inequalities holds for the corresponding x and y terms. Hence, a new w_i is defined to be the nearest integer of $(w_{i-1} \times \phi^{-1})$. Then procedure (2) is resumed with w_{i-1} and new w_i as the initial terms. The old w_i is included in the chain between w_{i-1} and new w_i, as a consequence there is a gap g_j =(old w_i−new w_i), which is included in the storage. Note that, subtraction is involved whenever old $w_i <$ new w_i.

We introduce another parameter LOWERBOUND as $u_{LB}x + v_{LB}y$. The above procedure (2) stops in either of the following three cases; (i) $(u_i < u_{LB})$ and $(v_i < v_{LB})$, (ii) $u_i < u_{LB}$ and $v_i > v_{LB}$, (iii) $u_i > u_{LB}$ and $v_i < v_{LB}$. The details of these three cases are included in the SGRAC algorithm.

B. Proposed SGRAC Algorithm

Algorithm 1 SGRAC Method (Dimension 2)

Input: An integer u, v, MAXIMALGAP and LOWERBOUND.
Output: $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}$, SAC_x, SAC_y and S.

1. $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2. $w_i \leftarrow u_i x + v_i y$
3. $w_0 \leftarrow ux + vy$
4. $w_1 \leftarrow [w_0 \times \phi^{-1}]$
5. $w_2 \leftarrow w_0 - w_1$
6. $m = \{0, 0\}$
7. $S = \{1x, 1y, 2x, 2y, 3x, 3y\}$
8. $G \leftarrow \emptyset$
9. $i \leftarrow 2$
10. $j \leftarrow 1$
11. $(u_{MG}x + v_{MG}y) \leftarrow$ MG
12. $(u_{LB}x + v_{LB}y) \leftarrow$ LB
13. **while** $(u_i x > u_{LB}x)$ or $(v_i y > v_{LB}y)$ **do**
14. $\quad m_i \leftarrow 0$
15. $\quad$ **if** $|w_i - (w_{i-1} \times \phi^{-1})| >$ MG or $w_i \leqslant \frac{w_{i-1}}{2}$ **then**
16. $\quad\quad w_{i+1} \leftarrow [w_{i-1} \times \phi^{-1}]$
17. $\quad\quad g_j \leftarrow (w_i - w_{i+1})$
18. $\quad\quad S \leftarrow S \cup \{g_j\}$
19. $\quad\quad j \leftarrow j + 1$
20. $\quad\quad m_{i-1} \leftarrow 2$, $m_i \leftarrow 1$, $m_{i+1} \leftarrow 0$
21. $\quad\quad m \leftarrow m \cup \{m_{i-1}, m_i, m_{i+1}\}$
22. $\quad\quad w_{i+2} \leftarrow (w_{i-1} - w_{i+1})$
23. $\quad\quad i \leftarrow i + 2$
24. $\quad$ **else**
25. $\quad\quad m \leftarrow m \cup \{m_i\}$
26. $\quad\quad i \leftarrow i + 1$
27. $\quad\quad w_i \leftarrow (w_{i-2} - w_{i-1})$
28. **if** $(u_i < u_{LB})$ and $(v_i < v_{LB})$ **then**
29. $\quad m_{i-1} \leftarrow 3$, $m_i \leftarrow 3$
30. $\quad m \leftarrow m \cup \{m_{i-1}, m_i\}$
31. $\quad T_1 \leftarrow w_{i-1}$, $T_0 \leftarrow w_i$
32. $\quad S \leftarrow S \cup \{T_1, T_0\}$
33. $\quad SAC_x$
34. $\quad SAC_y$
35. **else if** $u_i < u_{LB}$ and $v_i > v_{LB}$ **then**
36. $\quad g_j \leftarrow u_{i-1}x$, $g_{j+1} \leftarrow u_i x$
37. $\quad S \leftarrow S \cup \{g_j, g_{j+1}\}$
38. $\quad j \leftarrow j + 2$
39. $\quad SAC_x$
40. $\quad v_{i+1}y \leftarrow v_{i-1}y$
41. $\quad v_{i+2}y \leftarrow v_i y$
42. $\quad m_{i+1} \leftarrow 0$, $m_{i+2} \leftarrow 0$
43. $\quad m \leftarrow m \cup \{m_{i+1}, m_{i+2}\}$
44. $\quad v_{i+3}y \leftarrow (v_{i+1}y - v_{i+2}y)$
45. $\quad i \leftarrow i + 3$
46. $\quad$ Repeat step 13 to step 27 only for y terms
47. $\quad\quad m_{i-1} \leftarrow 3$, $m_i \leftarrow 3$
48. $\quad\quad m \leftarrow m \cup \{m_{i-1}, m_i\}$
49. $\quad\quad T_1 \leftarrow v_{i-1}y$, $T_0 \leftarrow v_i y$
50. $\quad\quad S \leftarrow S \cup \{T_1, T_0\}$
51. $\quad\quad SAC_y$
52. **else**
53. $\quad g_j \leftarrow v_{i-1}y$, $g_{j+1} \leftarrow v_i y$
54. $\quad S \leftarrow S \cup \{g_j, g_{j+1}\}$
55. $\quad j \leftarrow j + 2$
56. $\quad SAC_y$
57. $\quad u_{i+1}x \leftarrow u_{i-1}x$
58. $\quad u_{i+2}x \leftarrow u_i x$
59. $\quad m_{i+1} \leftarrow 0$, $m_{i+2} \leftarrow 0$
60. $\quad m \leftarrow m \cup \{m_{i+1}, m_{i+2}\}$
61. $\quad u_{i+3}x \leftarrow (u_{i+1}x - u_{i+2}x)$
62. $\quad i \leftarrow i + 3$
63. $\quad$ Repeat step 13 to step 27 only for x terms
64. $\quad\quad m_{i-1} \leftarrow 3$, $m_i \leftarrow 3$
65. $\quad\quad m \leftarrow m \cup \{m_{i-1}, m_i\}$
66. $\quad\quad T_1 \leftarrow u_{i-1}x$, $T_0 \leftarrow u_i x$
67. $\quad\quad S \leftarrow S \cup \{T_0, T_1\}$
68. $\quad\quad SAC_x$
69. $max \leftarrow j - 1$
70. $n \leftarrow i - 1$
71. $m \leftarrow$ reverse the arrangements in m and rename the elements in increasing order starting with numeral 1 to $n + 1$
72. **return** $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}$, SAC_x, SAC_y and S

Note that in step 15 we check either the inequalities are satisfied for the corresponding x terms or y terms. In steps $20, 29, 47$ and 64, the old m_{i-1} has been replaced with new m_{i-1} in m. Also note that SAC_x and SAC_y represents the construction of short addition chains using the absolute values of x terms and y terms in the storage, respectively. The symbols MG and LB represents MAXIMALGAP and LOWERBOUND, respectively.

Example 1. Evaluate Algorithm 1 for input $u = 10361$, $v = 103864$, LOWERBOUND $= 5x + 5y$ and MAXIMALGAP $= 10x + 10y$.

First, we will find the SGRAC representation m, during which we will obtain the elements for the storage S. Later, we will use all the storage elements to construct a short addition chain SAC_x and SAC_y.

We pair u and v with variables x and y to distinguish their computations. We begin by letting,

$$\begin{array}{rcll} w_0 &=& 10361x + 103864y\,, & m_0 = 0 \\ w_1 &=& [w_0 \times \phi^{-1}] = 6403x + 64191y\,, & m_1 = 0 \\ w_2 &=& w_0 - w_1 = 3958x + 39673y\,, & m_2 = 0 \\ w_3 &=& w_1 - w_2 = 2445x + 24518y\,, & m_3 = 0 \\ w_4 &=& w_2 - w_3 = 1513x + 15155y\,, & m_4 = 0 \\ w_5 &=& w_3 - w_4 = 932x + 9363y\,, & m_5 = 2 \\ w_6 &=& w_4 - w_5 = 581x + 5792y\,, & m_6 = 1 \end{array}$$

since $v_6 y$ exceeds the MAXIMALGAP, that is $|v_6 y - (v_5 y \times \phi^{-1})| > 5y$, we let

$$w_7 = [w_5 \times \phi^{-1}] = 576x + 5787y\,. \qquad m_7 = 0$$

There exist a gap, $g_1 = w_6 - w_7 = 5x+5y$, which we include in the storage. Let

$$\begin{array}{rcll} w_8 &=& w_5 - w_7 = 356x + 3576y\,, & m_8 = 0 \\ w_9 &=& w_7 - w_8 = 220x + 2211y\,, & m_9 = 0 \\ w_{10} &=& w_8 - w_9 = 136x + 1365y\,, & m_{10} = 0 \\ w_{11} &=& w_9 - w_{10} = 84x + 846y\,, & m_{11} = 0 \\ w_{12} &=& w_{10} - w_{11} = 52x + 519y\,, & m_{12} = 2 \\ w_{13} &=& w_{11} - w_{12} = 32x + 327y\,, & m_{13} = 1 \end{array}$$

since $v_{13} y$ exceeds MAXIMALGAP, that is $|v_{13} y - (v_{12} y \times \phi^{-1})| > 5y$, we let

$$w_{14} = [w_{12} \times \phi^{-1}] = 32x + 321y. \qquad m_{14} = 0$$

There exist a gap, $g_2 = 6y$, which we include in the storage. Let

$$\begin{array}{rcll} w_{15} &=& w_{12} - w_{14} = 20x + 198y\,, & m_{15} = 0 \\ w_{16} &=& w_{14} - w_{15} = 12x + 123y\,, & m_{16} = 3 \\ w_{17} &=& w_{15} - w_{16} = 8x + 75y\,, & m_{17} = 3 \end{array}$$

Henceforth, we terminate the x term of w_i, since it transcends the corresponding LOWERBOUND $= 10x$. Thus, we store $g_3 = 12x$ and $g_4 = 8x$. We continue the above process, considering the y terms only by letting

$$\begin{array}{rcll} w_{18} &=& 123y\,, & m_{18} = 0 \\ w_{19} &=& 75y\,, & m_{19} = 0 \end{array}$$

Hence, we have

$$\begin{array}{rcll} w_{20} &=& w_{18} - w_{19} = 48y\,, & m_{20} = 0 \\ w_{21} &=& w_{19} - w_{20} = 27y\,, & m_{21} = 0 \\ w_{22} &=& w_{20} - w_{21} = 21y\,, & m_{22} = 0 \end{array}$$

We stop the above continuous procedure at w_{22}, since the next term,

$$w_{23} = w_{21} - w_{22} = 6y\,,$$

transcends the given LOWERBOUND $= 10y$. Let $T_0 = 6y$, $T_1 = 21y$ and store in S. Hence, we obtained the following storage.

$$S = \{1x, 1y, 2x, 2y, 3x, 3y, g_1 = 5x + 5y, g_2 = 6y,$$

$$g_3 = 12x, g_4 = 8x, T_1 = 21y, T_0 = 6y\}.$$

We list $m_0, \ldots, m_{22}$ as elements of set m. Hence, we have

$$m = \{0, 0, 0, 0, 0, 2, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 3, 3, 0, 0, 0, 0, 0\}.$$

Then we reverse the arrangements of the elements in the set m and rename it in increasing order starting from m_1 to m_{23}. Thus, it results in the following SGRAC representation.

$$m = \{0, 0, 0, 0, 0, 3, 3, 0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0\}_{SGRAC}.$$

Next, we shall consider constructing doubling-free short addition chain including absolute values of all x terms in S. We will denote it as SAC_x. Hence, we have

$$\{1x, 2x, 3x, 5x, 12x, 8x\}.$$

Excluding the repeated terms and rearrangement results

$$\{1x, 2x, 3x, 5x, 8x, 12x\}.$$

It follows that $1x + 2x \to 3x$, $2x + 3x \to 5x$, $3x + 5x \to 8x$, $3x + 8x \to 11x$ and $1x + 11x \to 12x$. Hence, the following results the SAC_x.

$$1 \to 2 \to 3 \to 5 \to 8 \to 11 \to 12.$$

Now, we shall consider constructing doubling-free short addition chain including absolute values of all y terms in S. We will denote it as SAC_y. Hence, we have

$$\{1y, 2y, 3y, 5y, 6y, 21y, 6y\}.$$

Excluding the repeated terms and rearrangement results

$$\{1y, 2y, 3y, 5y, 6y, 21y\}.$$

It follows that $1y + 2y \to 3y$, $2y + 3y \to 5y$, $y + 5y \to 6y$, $5y + 6y \to 11y$, $6y + 11y \to 17y$, $3y + 17y \to 20y$ and $1y + 20y \to 21y$. Hence the following results the SAC_y.

$$1 \to 2 \to 3 \to 5 \to 6 \to 11 \to 17 \to 20 \to 21\,.$$

IV. Application to Elliptic Curve Cryptosystems

In this section, we propose a multi-scalar multiplication algorithm by utilizing the proposed SGRAC method.

Algorithm 2 Multi-scalar multiplication using SGRAC (Dimension 2)

Input: An integer u, v and $P, Q \in E(\mathbb{F}_p)$.
Output: $uP + vQ$.

Precomputation (SGRAC method)
1. $m = \{m_1, \ldots, m_{n+1}\}_{SGRAC}$
2. S
3. SAC_x and SAC_y
4. $G \leftarrow \emptyset$
5. $x \leftarrow P$, $y \leftarrow Q$
6. **for** $j = 1$ **to** max
7. $\quad g_j \leftarrow$ compute using SAC_x and SAC_y
8. $\quad G \leftarrow G \cup \{g_j\}$
9. $G \leftarrow$ reverse the arrangements in G and rename the elements in increasing order starting with numeral 1 to max
10. $T_0 \leftarrow$compute using SAC_x or SAC_y
11. $T_1 \leftarrow$compute using SAC_x or SAC_y

Main loop
12. $j \leftarrow 1$
13. **for** $i = 1$ **to** n **do**
14. $\quad$ **if** $e_{i+1} = 0$ **then**
15. $\quad\quad T_{i+1} \leftarrow T_{i-1} + T_i$
16. $\quad$ **else if** $e_{i+1} = 1$ **then**
17. $\quad\quad T_{i+1} \leftarrow T_i + G_j$
18. $\quad\quad j \leftarrow j + 1$
19. $\quad$ **else if** $e_{i+1} = 2$ **then**
20. $\quad\quad T_{i+1} \leftarrow T_{i-2} + T_{i-1}$
21. $\quad$ **else** $e_{i+1} = 3$ **then**
22. $\quad\quad T_{i+1} \leftarrow T_{i-1} + G_j$
23. $\quad\quad j \leftarrow j + 1$
24. **return** T_{n+1}

Hence the required output $uP + vQ = T_{n+1}$. Note that Algorithm 2 involves storage of the preceding two points during scalar multiplication. Also, a temporary storage G containing max number of points g_j's, which are discarded during the scalar multiplication after being used, hence having less constraint on memory containing devices.

Example 2. Compute $10361P + 103864Q$ using Algorithm 2.

Precomputation.
Example 1 results the following.
$m = \{0, 0, 0, 0, 0, 3, 3, 0, 0, 1, 2, 0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 0\}_{SGRAC}$.
$S = \{1x, 1y, 2x, 2y, 3x, 3y, g_1 = 5x + 5y, g_2 = 6y, g_3 = 12x,$
$g_4 = 8x, T_1 = 21y, T_0 = 6y\}$.
$SAC_x : 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow 11 \rightarrow 12$.
$SAC_y : 1 \rightarrow 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 11 \rightarrow 17 \rightarrow 20 \rightarrow 21$.

Next, we replace x and y with P and Q in S, respectively. Then, we compute all g_j's using SAC_x and SAC_y for $j = 1$ to 4, which are then stored in G. Hence, we have $G = \{g_1 = 5P + 5Q, g_2 = 6Q, g_3 = 12P, g_4 = 8P\}$. Now we reverse the arrangements in G and rename the elements in increasing order starting with numeral 1 to 4. Hence, we have $G = \{g_1 = 8P, g_2 = 12P, g_3 = 6Q, g_4 = 5P + 5Q\}$. We compute $T_0 = 6Q$ and $T_1 = 21Q$ using SAC_y.

Evaluation Stage.
Henceforth, we use the SGRAC representation of m, to compute T_i for $i = 2$ to $i = 23$ as follows.

$$
\begin{array}{llll}
i = 1, & m_2 = 0, & T_2 = T_0 + T_1 & = 27Q, \\
i = 2, & m_3 = 0, & T_3 = T_1 + T_2 & = 48Q, \\
i = 3, & m_4 = 0, & T_4 = T_2 + T_3 & = 75Q, \\
i = 4, & m_5 = 0, & T_5 = T_3 + T_4 & = 123Q, \\
i = 5, & m_6 = 3, & T_6 = T_4 + g_1 & = 8P + 75Q, \\
i = 6,, & m_7 = 3, & T_7 = T_5 + g_2 & = 12P + 123Q, \\
i = 7, & m_8 = 0, & T_8 = T_6 + T_7 & = 20P + 198Q, \\
i = 8, & m_9 = 0, & T_9 = T_7 + T_8 & = 32P + 321Q, \\
i = 9, & m_{10} = 1, & T_{10} = T_9 + g_3 & = 32P + 327Q, \\
i = 10, & m_{11} = 2, & T_{11} = T_8 + T_9 & = 52P + 519Q, \\
i = 11, & m_{12} = 0, & T_{12} = T_{10} + T_{11} & = 84P + 846Q, \\
i = 12, & m_{13} = 0, & T_{13} = T_{11} + T_{12} & = 136P + 1365Q, \\
i = 13, & m_{14} = 0, & T_{14} = T_{12} + T_{13} & = 220P + 2211Q, \\
i = 14, & m_{15} = 0, & T_{15} = T_{13} + T_{14} & = 356P + 3576Q, \\
i = 15, & m_{16} = 0, & T_{16} = T_{14} + T_{15} & = 576P + 5787Q, \\
i = 16, & m_{17} = 1, & T_{17} = T_{16} + g_4 & = 581P + 5792Q, \\
i = 17, & m_{18} = 2, & T_{18} = T_{15} + T_{16} & = 932P + 9363Q, \\
i = 18, & m_{19} = 0, & T_{19} = T_{17} + T_{18} & = 1513P + 15155Q, \\
i = 19, & m_{20} = 0, & T_{20} = T_{18} + T_{19} & = 2445P + 24518Q, \\
i = 20, & m_{21} = 0, & T_{21} = T_{19} + T_{20} & = 3958P + 39673Q, \\
i = 21, & m_{22} = 0, & T_{22} = T_{20} + T_{21} & = 6403P + 64191Q, \\
i = 22, & m_{23} = 0, & T_{23} = T_{21} + T_{22} & = 10361P + 103864Q.
\end{array}
$$

TABLE I
THE DISTRIBUTION OF PRECOMPUTATION CHAIN LENGTHS FOR 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

length (ℓ)	12	13	14	15	16	17	18	19	20
# inputs	31	198	463	144	103	37	12	10	2

TABLE II
THE DISTRIBUTION OF STORAGE FOR 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

Storage capacity	14	15	16	17	18	19
# inputs	5	42	173	397	345	38

V. EXPERIMENTAL RESULTS

In this section, we show the experimental analysis for multi-scalar multiplication algorithm based on SGRAC method for the case of dimension 2. We consider the factors necessitated in obtaining the best results.

We carried out an experiment to analyze Algorithm 2 using a python programming language on 2.60 GHz intel celeron processor. We randomly selected 1000 integers u and v of 160 bits and set the searching range of the parameters, LOWERBOUND to be between $4x + 4y$ to $23x + 23y$ and MAXIMALGAP to be between $5x + 5y$ to $16x + 16y$. It took 209 trials to obtain chains of lengths between 295 to 316 as shown in Table IV. On average it took about 11.31 seconds to find each chain. The Table I, II, III, V and VI shows the distribution of precomputation, storage, main loop, MAXIMALGAP and LOWERBOUND, respectively. The average chain length is found to be 307 and the average storage capacity is found to be 17. Experiment result shows that SGRAC method achieves 63% of Fibonacci pattern, but overall we could not guarantee it to be optimal.

In a similar experiment as above, we give an experimental analysis of multi-scalar multiplication based on SGRAC method for dimensions $t = 2, \ldots, 6$ as shown in Table VII. We randomly selected 1000 scalars k_i's of 160 bits for the respective dimensions. It shows that there is a linear increase in the storage capacity and the addition chains with respect to dimension.

VI. CONCLUSION

In this paper we have proposed a novel algorithm for the simultaneous computation of multi-scalar multiplication, that

TABLE VII
ANALYSIS OF SIMULTANEOUS MULTI-SCALAR MULTIPLICATION ALGORITHMS FOR HIGHER DIMENSIONS.

Dimension (t)	Avg. storage	Average precomputation	Average Main loop	Average Cost	Fibonacci pattern	Average Run time
2	17	14 ECADD + 2 ECDBL	293 ECADD	307 ECADD + 2 ECDBL	63%	10.7 sec
3	26	21 ECADD + 3 ECDBL	333 ECADD	354 ECADD + 3 ECDBL	54%	13.9 sec
4	35	28 ECADD + 4 ECDBL	372 ECADD	400 ECADD + 4 ECDBL	47%	17.5 sec
5	44	34 ECADD + 5 ECDBL	411 ECADD	445 ECADD + 5 ECDBL	43%	21 sec
6	53	41 ECADD + 6 ECDBL	449 ECADD	490 ECADD + 6 ECDBL	39%	24.3 sec

TABLE III
THE DISTRIBUTION MAIN LOOP CHAIN LENGTHS OF 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

length (ℓ)	280	281	282	283	284	285	286	287
# inputs	2	1	-	5	6	8	26	30

length (ℓ)	288	289	290	291	292	293	294	295
# inputs	37	60	80	80	90	110	109	118

length (ℓ)	296	297	298	299	300	301	302	303
# inputs	103	49	41	20	15	7	2	1

TABLE IV
THE DISTRIBUTION OF TOTAL CHAIN LENGTHS FOR 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

length (ℓ)	295	296	297	298	299	300	301	302
# inputs	1	2	1	3	3	13	25	34

length (ℓ)	303	304	305	306	307	308	309	310
# inputs	48	71	83	105	121	127	125	103

length (ℓ)	311	312	313	314	315	316
# inputs	60	33	22	11	7	2

TABLE V
THE DISTRIBUTION OF MAXIMALGAP FOR 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

MAXIMALGAP	5	6	7
# inputs	783	203	14

TABLE VI
THE DISTRIBUTION OF LOWERBOUND FOR 1000 RANDOMLY SELECTED INTEGERS u AND v OF 160 BIT.

LOWERBOUND	4	5	6	7	8	9	10	11
# inputs	248	154	197	137	75	54	34	28

LOWERBOUND	12	13	14	15	16	17	18	19
# inputs	14	12	8	15	8	4	3	4

LOWERBOUND	20	21	22
# inputs	3	1	1

is by employing addition chains. In order to accomplish our purpose, we have proposed an efficient empirical method to generate addition chains for multi-exponents simultaneously. The analysis from Table VII shows that there is a linear increase in the cost with respect to the dimension of the multi-scalar multiplication. Further work may include reducing storage capacity and chain length in order to enhance further efficiency.

REFERENCES

[1] R.M. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.
[2] M. Bellare, J.A. Garray, and T. Rabin. Fast Batch verification for modular exponentiation and digital signatures. *Advances in Cryptology-EUROCRYPTO'98*, volume 1403 of *Lecture Notes in Computing Science*, pages 236-250. Springer-Verlag, 1998.
[3] S. Brands. Rethinking Public Key Infrastructures and Digital Certificates-Building in Privacy. *MIT Press*, p.356, 2000.
[4] T. ElGamal. A Public key Cryptosystem and a Signature Scheme Based on Discrete Logarithms. *IEEE Transactions on Information Theory*, vol.31, pp.469-472, 1985.
[5] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography.* Springer-Verlag, 2004.
[6] D. Knuth. Fundamental Algorithms. *The Art of Computer Programming*, volume 1, Addision-Wesley, 1981.
[7] K. Kobayashi, H. Morita, and M. Hakuta. Multiple Scalar-Multiplication Algorithm over Elliptic Curve. *IEICE Transactions on Information and System*, E84-D, No.2, pp.271-276, Feb.2001.
[8] C.H. Lim and P.J. Lee. More Flexible Exponentiation with Precomputation. *Advances in Cryptology-CRYPTO'94*, volume 839 of *Lecture Notes in Computing Science*, pages 95-107. Springer-Verlag, 1994.
[9] B. Möller. Algorithms for Multi-exponentiation. *Selected Areas in Cryptography*, volume 2259 of *Lecture Notes in Computing Science*, pages 165-180. Springer-Verglag, 2001.
[10] A.J. Menezes, P.C. vanOorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press, 1997.
[11] T. Okamoto. Provably secure and practical identification schemes and corresponding signature schemes. *Advances in Cryptology-CRYPTO'92*, volume 740 of *Lecture Notes in Computing Science*, pages 31-53. Springer-Verlag, 1993.
[12] T. Okamoto. Practical identification schemes as secure as the DL and RSA problems. *http://grouper.ieee.org/groups/1363/addendum.html#Okamoto*, March 1999.
[13] N. Vorobiev. *Fibonacci Numbers.* Birkhuser Verlag, 2002.