SPA Resistant Scalar Multiplication using Golden Ratio Addition Chain Method

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

Abstract—**In this paper we propose an efficient and secure (SPA resistant) elliptic curve scalar multiplication algorithm over odd prime fields. For this purpose, we propose an explicit algorithm for short addition-subtraction chain method which utilizes a golden ratio. We term it as golden ratio addition chain method or GRAC method in short. Our proposed scalar multiplication algorithm based over GRAC method has preceded by 3% to 18% over the previous ones known in the literature. Hence scalar multiplication utilizing GRAC method shows significance in application to elliptic curve cryptosystems.**

Keywords: elliptic curve cryptosystems, scalar multiplication, addition chain, Euclidean addition chain (EAC), Fibonacci sequence.

1 Introduction

Elliptic curve cryptography was proposed independently in 1985 by Neal Koblitz [11] and Victor Miller [14]. Since then it is widely accepted due to its fascinating feature of having smaller key length of 160-bit in elliptic curve cryptosystems (ECC) to provide same level of security as for RSA with a 1024-bit of key length. An extensive amount of research has been dedicated to securing and accelerating its implementations. The overall efficiency of most ECC are dominated by computations of the form kP which is known as a scalar multiplication, where P is an elliptic curve point, and k is an arbitrary integer, which plays a role of a secret scalar. Hence, an efficient and secure scalar multiplication are essential in ECC. Most of the scalar multiplication methods utilizes multiple operations such as double-and-add, triple-and-add etc. These operations when dependable on secret scalar, are susceptible to leak secret information through simple power analysis (SPA), which is one type of side-channel attack discovered by Kocher et al. [12]. The SPA monitors the power consumption of a single execution during scalar multiplication. The fact that different operations has different power consumption, helps attackers to retrieve secret data especially when the operations are dependable on secret scalars. Thus, in order to resist SPA attack, scalar multiplication should be implemented using fixed sequence of operations [4]. One of the solution could be the use of doubling-free addition chain for scalar multiplication. Note that the appearance of numeral 2 in the chain is ineffective for SPA attack, since the computation of one doubling, that is $2P$, is a necessary computation in almost all scalar multiplication.

In this paper, we propose an explicit algorithm for the previous [7] golden ratio addition-subtraction chain method (GRASC method) to make it compatible in presenting scalar multiplication algorithm. We term it as the golden ratio addition chain method or GRAC method in short. Note that the subtraction involved in the GRASC method has been excluded in our proposed algorithm (GRAC method), thus facilitating us in presenting an elegant scalar multiplication algorithm. Instead of using subtraction, we consider taking inversion of a point which is cost negligible in elliptic curve cryptography. This results in a series of addition operation during scalar multiplication, hence resistant to SPA attack.

The rest of this paper is organized as follows. In section 2, we give a brief overview on elliptic curve cryptography, with some classic definitions on addition chains and Fibonacci sequence. In section 3, we review the GRASC method and propose an explicit algorithm for it, called GRAC method. In section 4, we exploit GRAC method by proposing a scalar multiplication algorithm. In section 5, we discuss and compare our results with the previous ones in the literature.

2 Background

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

2.1 Elliptic Curve Cryptography

We start with a practical definition of the concept of an elliptic curve. More details could be cited from [1, 2, 8].

Definition 2.1 *An elliptic curve E over a finite field K is defined by an equation*

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \qquad (1)$$

*R.R.Goundar is with Computing and Mathematics Department, 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

(Revised online publication: 21 June 2008)

where $a_1, a_2, a_3, a_4, a_6 \in K$, and $\Delta \neq 0$, where Δ is the discriminant of E.

In practice, the Weierstrass equation (1) can be greatly simplified by applying admissible changes of variables. If the characteristic of K is not equal to 2 and 3, then (1) rewrites

$$y^2 = x^3 + ax + b \tag{2}$$

where $a, b \in K$, and $\Delta = -16(4a^3 + 27b^2) \neq 0$. When the characteristic of K is equal to 2, we use the non-supersingular form of an elliptic curve, given for $a \neq 0$ by

$$y^2 + xy = x^3 + ax^2 + b \tag{3}$$

where $a, b \in K$ and $\Delta = b \neq 0$. The set $E(K)$ of rational points on an elliptic curve E defined over a finite field K is an abelian group, where the operation (generally denoted additively) is defined by the well-known law of chord and tangent, and the identity element is the special point $\mathcal{O}$, called the point at infinity. If the points on the curve are represented using affine coordinates, as $P = (x, y)$, both the point addition and point doubling involve an expensive field inversion (to compute the slope of the chord of the tangent). To avoid these inversion, several projective systems of coordinates have been proposed in literature [1]. The major ones include the, affine coordinate system ($\mathcal{A}$), projective coordinates system ($\mathcal{P}$), Jacobian coordinates system ($\mathcal{J}$), Chudnovsky Jacobian coordinates system ($\mathcal{J}^C$) and modified Jacobian coordinates system ($\mathcal{J}^m$).

In this paper, we will consider mixed coordinates system since it has lower computational cost compared to other coordinate systems as proposed by Cohen et.al [5]. We will select the best operation for calculating the cost of GRAC based scalar multiplication algorithm. Note that we will use $[i], [s]$ and $[m]$ to denote the cost of one inversion, one squaring and one multiplication respectively. We shall always leave out the cost of field additions. Generally, it is assumed $[s] = 0.8[m]$ for curves over odd prime field [6].

2.2 Review on Addition Chains and Fibonacci Sequence

Here, we briefly state some classic definitions used in the study of addition chains. More details could be cited from [1, 16].

Definition 2.2 *An addition chain computing an integer k is given by two sequences $v = (v_0, \ldots, v_\ell)$ and $w = (w_1, \ldots, w_\ell)$ such that $v_0 = 1, v_\ell = k, v_i = v_r + v_s$, for all $1 \leq i \leq \ell$ with respect to $w_i = (r, s)$ and $0 \leq r, s \leq i - 1$. The length of the addition chain is ℓ.*

Definition 2.3 *An Euclidean addition chain (EAC) is an addition chain which satisfies $v_1 = 1, v_2 = 2, v_3 = v_2 + v_1$ and for all $3 \leq i \leq \ell - 1$, if $v_i = v_{i-1} + v_j$ for some $j < i - 1$, then $v_{i+1} = v_i + v_{i-1}$ or $v_{i+1} = v_i + v_j$.*

Definition 2.4 *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 [9, 17] but we recall only one here, by stating the following Binet's Formula.

Theorem 2.1 *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.

$$\lim_{n \to \infty} \frac{F_n}{F_{n-1}} = \phi, \tag{4}$$

where ϕ is a golden ratio, also known as a golden section.

3 GRASC Method and Proposed Explicit Algorithm

In this section, we review the GRASC method [7] for finding doubling-free short addition-subtraction chain for an arbitrary positive integer and later, we propose an explicit algorithm for it.

3.1 Review on GRASC Method [7]

Here we review the GRASC method for finding doubling-free short addition-subtraction chain by utilizing a precise golden ratio.

The GRASC method considers making chain starting from the last term, which is the input k and aims to follow a Fibonacci pattern using the fact from equation (4). Hence, it tries to maintain a near golden ratio value between two succeeding terms. It begins by letting

$$\begin{aligned} u_0 &= k, \\ u_1 &= [u_0 \times \phi^{-1}], \\ u_i &= u_{i-2} - u_{i-1} \text{for } i = 2, 3, \ldots \end{aligned} \tag{5}$$

Here u_i denotes the reverse of v_i that is, $u_i = v_{\ell - i}$. If continued with the procedure (5), u_i will exponentially deviate from $(u_{i-1} \times \phi^{-1})$ as i increases. In order to overcome this problem, a parameter MAXIMALGAP is introduced, such that the above procedure (5) terminates whenever

$$|u_i - (u_{i-1} \times \phi^{-1})| > \text{ MAXIMALGAP or } u_i \leqslant \frac{u_{i-1}}{2}.$$

In such case, a new u_i is defined to be the nearest integer of $(u_{i-1} \times \phi^{-1})$. Then procedure (5) is resumed with u_{i-1} and new u_i as the initial terms. The old u_i is included in the chain between u_{i-1} and new u_i, as a consequence there is a gap g_j =(old u_i−new u_i), which is included in the storage. Note that, subtraction is involved whenever old $u_i <$ new u_i. Another parameter LOWERBOUND is introduced to cease the procedure (5) when $u_i \leq$ LOWERBOUND. The storage initially consists of $1, 2,$ and 3. Later g_j's are included in the storage. Once the execution of procedure (5) is ceased, the last two u_i's of the chain is included in the storage. Thus, using the storage, a short addition chain is found randomly without using doubling, except for numeral 2. Finally, this chain is joined to the third last u_i of the previous chain resulting in a moderately short addition-subtraction chain for the given input k. Note that the storage capacity is dependent on the experimentally selected values of the two parameters.

3.2 Our Proposed Explicit Algorithm

Here, we propose an explicit algorithm for the GRASC method and rename it as golden ratio addition chain method or GRAC method in short. This explicit algorithm will facilitate us in proposing a scalar multiplication algorithm in the next section.

Algorithm 1 Golden Ratio Addition Chain Method

```
Input: A positive integer $k$, MG and LB.
Output: $m = \{e_1, \dots, e_{n+1}\}_{GRAC}$,
$S = \{1, 2, 3, g_1, \dots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$, $SAC$.
1.  $\phi^{-1} \leftarrow \frac{-1+\sqrt{5}}{2}$
2.  $u_0 \leftarrow k$
3.  $u_1 \leftarrow [u_0 \times \phi^{-1}]$
4.  $u_2 \leftarrow u_0 - u_1$
5.  $m = \{0, 0\}$
6.  $S = \{1, 2, 3\}$
7.  $i \leftarrow 2$
8.  $j \leftarrow 1$
9.  while $u_i >$ LB do
10.     $e_i \leftarrow 0$
11.     if $|u_i - (u_{i-1} \times \phi^{-1})| >$ MG or $u_i \leqslant \frac{u_{i-1}}{2}$ then
12.         $u_{i+1} \leftarrow [u_{i-1} \times \phi^{-1}]$
13.         $e_i \leftarrow 1$, $e_{i-1} \leftarrow 2$, $e_{i+1} \leftarrow 0$
14.         $m \leftarrow m \cup \{e_i, e_{i-1}, e_{i+1}\}$
15.         $g_j \leftarrow u_i - u_{i+1}$
16.         $S \leftarrow S \cup \{g_j\}$
17.         $j \leftarrow j + 1$
18.         $u_{i+2} \leftarrow u_{i-1} - u_{i+1}$
19.         $i \leftarrow i + 2$
20.     else
21.         $m \leftarrow m \cup \{e_i\}$
22.         $i \leftarrow i + 1$
23.         $u_i \leftarrow u_{i-2} - u_{i-1}$
24. $S \leftarrow S \cup \{u_i, u_{i-1}, u_{i-2}\}$
25. $max \leftarrow j$
26. $n \leftarrow i - 1$
27. $m \leftarrow$ reverse the arrangements in $m$ and rename the
    elements in increasing orderstarting with numeral 1 to $n + 1$
28. $SAC \leftarrow$a short addition chain using absolute values
    of $g_j$'s and other terms in $S$
29. return  $m = \{e_1, \dots, e_{n+1}\}_{GRAC}$,
    $S = \{1, 2, 3, g_1, \dots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$, $SAC$
```

In steps 14, the old e_{i-1} has been replaced with new e_{i-1} in m. Also note that the symbols MG and LB represents MAXIMALGAP and LOWERBOUND, respectively.

Example 1. Evaluate Algorithm 1 for input $k = 207062$, LOWERBOUND= 5 and MAXIMALGAP= 6.

First, we will find the GRAC representation m, during which we will obtain the elements for the storage S. Later, we will use all the storage elements to search for a short addition chain.

We begin by letting,

$$\begin{array}{lcll}
u_0 &=& k = 207062\,, & e_0 = 0\\
u_1 &=& [u_0 \times \phi^{-1}] = 127971\,, & e_1 = 0\\
u_2 &=& u_0 - u_1 = 79091\,, & e_2 = 0\\
u_3 &=& u_1 - u_2 = 48880\,, & e_3 = 0\\
u_4 &=& u_2 - u_3 = 30211\,, & e_4 = 0\\
u_5 &=& u_3 - u_4 = 18669\,, & e_5 = 0\\
u_6 &=& u_4 - u_5 = 11542\,, & e_6 = 0\\
u_7 &=& u_5 - u_6 = 7127\,, & e_7 = 2\\
u_8 &=& u_6 - u_7 = 4415\,, & e_8 = 1
\end{array}$$

since u_8 exceeds the MAXIMALGAP, that is $|4415 - (7127 \times \phi^{-1})| > 6$, we let

$$u_9 = [u_7 \times \phi^{-1}] = 4405\,. \qquad e_9 = 0$$

There exist a gap, $g_1 = 4415 - 4405 = 10$, which we include in the storage. Let

$$\begin{array}{lcll}
u_{10} &=& u_7 - u_9 = 2722\,, & e_{10} = 0\\
u_{11} &=& u_9 - u_{10} = 1683\,, & e_{11} = 0\\
u_{12} &=& u_{10} - u_{11} = 1039\,, & e_{12} = 0\\
u_{13} &=& u_{11} - u_{12} = 644\,, & e_{13} = 0\\
u_{14} &=& u_{12} - u_{13} = 395\,, & e_{14} = 0\\
u_{15} &=& u_{13} - u_{14} = 249\,, & e_{15} = 2\\
u_{16} &=& u_{14} - u_{15} = 146\,, & e_{16} = 1
\end{array}$$

since u_{16} exceeds MAXIMALGAP, that is $|146 - (249 \times \phi^{-1})| > 6$, we let

$$u_{17} = [u_{15} \times \phi^{-1}] = 154. \qquad e_{17} = 0$$

There exist a gap, $g_2 = 146 - 154 = -8$, which we include in the storage. Let

$$\begin{array}{lcll}
u_{18} &=& u_{15} - u_{17} = 95\,, & e_{18} = 0\\
u_{19} &=& u_{17} - u_{18} = 59\,, & e_{19} = 0\\
u_{20} &=& u_{18} - u_{19} = 36\,, & e_{20} = 0\\
u_{21} &=& u_{19} - u_{20} = 23\,, & e_{21} = 0\\
u_{22} &=& u_{20} - u_{21} = 13\,, & e_{22} = 0\\
u_{23} &=& u_{21} - u_{22} = 10\,. & e_{23} = 0
\end{array}$$

We stop the above continuous procedure at u_{23}, since the next term, $u_{24} = u_{22} - u_{23} = 3$, transcends the given LOWERBOUND. We obtained the following storage.

$$S = \{1\,, 2\,, 3\,, g_1 = 10\,, g_2 = -8\,, u_{22} = 13\,, u_{23} = 10\,, u_{24} = 3\}\,.$$

We list $e_0, \dots, e_{23}$ as elements of set m.

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

Then we reverse the arrangements of the elements in the set m and rename it in increasing order starting from e_1. Thus, it results in the following GRAC representation.

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

Next, we randomly search for a doubling-free short addition chain (SAC) for absolute values of g_j's and other terms in storage S obtained before. That is

$$S = \{1\,,2\,,3\,,10\,,8\,,13\,,10\,,3\}$$

Excluding the repeated numbers and rearranging results

$$S = \{1\,,2\,,3\,,8\,,10\,,13\}$$

It follows that $1+2 \to 3$. We need to insert 5 so that $3+5 \to 8$. Next, we have $2+8 \to 10$ and $3+10 \to 13$. Hence, following is the doubling-free short addition chain.

$$1 \to 2 \to 3 \to 5 \to 8 \to 10 \to 13\,.$$

4 Application to Elliptic Curve Cryptosystems

In this section, we propose a SPA resistant scalar multiplication algorithm by utilizing the proposed GRAC method.

```
Algorithm 2 Scalar multiplication using GRAC method
Input: An integer $k$ and $P \in E(\mathbb{F}_{q^k})$ .
Output: $kP$.
Precomputation (GRAC method)
1.   $m = \{e_1, \dots, e_{n+1}\}_{GRAC}$
2.   $S = \{1, 2, 3, g_1, \dots, g_{max}, u_{i-2}, u_{i-1}, u_i\}$
3.   $SAC$
Main loop
4.   $G \leftarrow \emptyset$
5.   for j=1 to $max$
6.       $G_j \leftarrow g_j P$ (using $SAC$)
7.       $G \leftarrow G \cup \{G_j\}$
8.   $G \leftarrow$ reverse the arrangements in $G$ and rename the
      elements in increasing order starting with numeral
      1 to $max$
9.   $T_0 \leftarrow u_i P$ (using $SAC$)
10.  $T_1 \leftarrow u_{i-1} P$ (using $SAC$)
11.  $T_2 \leftarrow u_{i-2} P$ (using $SAC$)
12.  $j \leftarrow 1$
13.  for $i = 2$ to $n$ do
14.      if $e_{i+1} = 0$ then
15.          $T_{i+1} \leftarrow T_i + T_{i-1}$
16.      if $e_{i+1} = 1$ then
17.          $T_{i+1} \leftarrow T_i + G_j$
18.          $j \leftarrow j + 1$
19.      if $e_{i+1} = 2$ then
20.          $T_{i+1} \leftarrow T_{i-1} + T_{i-2}$
21.  return $T_{n+1}$
```

Hence the required output $kP = T_{n+1}$. 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. The proposed algorithm uses a fixed sequence of addition operation therefore it is SPA resistant.

Example 2. Compute $207062P$ using Algorithm 2.

Precomputation. (Example 1)

1. $m = \{\mathbf{0}\,,\mathbf{0}\,,0\,,0\,,0\,,0\,,0\,,1\,,2\,,0\,,0\,,0\,,0\,,0\,,0\,,1\,,2\,,0\,,0\,,0\,,0\,,0\,,0\,,0\}_{GRAC}\,.$

2. $S = \{1\,,2\,,3\,,g_1 = 10\,,g_2 = -8\,,u_{22} = 13\,, u_{23} = 10\,,u_{24} = 3\}$.

3. $SAC : 1 \to 2 \to 3 \to 5 \to 8 \to 10 \to 13$.

Evaluation Stage.
For an input $P \in E(\mathbb{F}_{q^k})$, we utilize the above short addition chain (SAC) to compute the following: $2P\,, P+2P = 3P\,, 2P+3P = 5P\,, 3P+5P = 8P$, taking it's inverse gives $-8P$, followed by $2P+8P = 10P$ and $3P+10P = 13P$. Hence, we have $G_1 = g_1P = 10P$ and $G_2 = g_2P = -8P$, which we store in G resulting as $G = \{10P\,, -8P\}$. Now, we reverse the arrangements in G and rename the terms in increasing order starting with numeral 1. Thus, $G = \{-8P\,, 10P\}$ where $G_1 = -8P$ and $G_2 = 10P$. Also, we have $T_0 = 3P$, $T_1 = 10P$, $T_2 = 13P$. Henceforth, we use the GRAC representation m, to compute T_i for $i = 2$ to $i = 23$ as follows.

$i = 2$,	$e_3 = 0$,	$T_3 = T_2 + T_1 = 23P$,
$i = 3$,	$e_4 = 0$,	$T_4 = T_3 + T_2 = 36P$,
$i = 4$,	$e_5 = 0$,	$T_5 = T_4 + T_3 = 59P$,
$i = 5$,	$e_6 = 0$,	$T_6 = T_5 + T_4 = 95P$,
$i = 6$,,	$e_7 = 0$,	$T_7 = T_6 + T_5 = 154P$,
$i = 7$,	$e_8 = 1$,	$T_8 = T_7 + G_1 = 146P$,
$i = 8$,	$e_9 = 2$,	$T_9 = T_7 + T_6 = 249P$,
$i = 9$,	$e_{10} = 0$,	$T_{10} = T_9 + T_8 = 395P$,
$i = 10$,	$e_{11} = 0$,	$T_{11} = T_{10} + T_9 = 644P$,
$i = 11$,	$e_{12} = 0$,	$T_{12} = T_{11} + T_{10} = 1039P$,
$i = 12$,	$e_{13} = 0$,	$T_{13} = T_{12} + T_{11} = 1683P$,
$i = 13$,	$e_{14} = 0$,	$T_{14} = T_{13} + T_{12} = 2722P$,
$i = 14$,	$e_{15} = 0$,	$T_{15} = T_{14} + T_{13} = 4405P$,
$i = 15$,	$e_{16} = 1$,	$T_{16} = T_{15} + G_2 = 4415P$,
$i = 16$,	$e_{17} = 2$,	$T_{17} = T_{15} + T_{14} = 7127P$,
$i = 17$,	$e_{18} = 0$,	$T_{18} = T_{17} + T_{16} = 11542P$,
$i = 18$,	$e_{19} = 0$,	$T_{19} = T_{18} + T_{17} = 18669P$,
$i = 19$,	$e_{20} = 0$,	$T_{20} = T_{19} + T_{18} = 30211P$,
$i = 20$,	$e_{21} = 0$,	$T_{21} = T_{20} + T_{19} = 48880P$,
$i = 21$,	$e_{22} = 0$,	$T_{22} = T_{21} + T_{20} = 79091P$,
$i = 22$,	$e_{23} = 0$,	$T_{23} = T_{22} + T_{21} = 127971P$,
$i = 23$,	$e_{24} = 0$,	$T_{24} = T_{23} + T_{22} = 207062P$.

5 Discussion

In this section, we discuss our results and make comparison with some previous methods in the literature.

In an experiment carried out in [7] showed that the best case of GRASC method was chain of length 258, hence similar result holds for the GRAC method. Note that GRASC method is renamed as GRAC method. This is

Table 1: Computational costs using mixed coordinates

	Addition	Doubling
Operation	$\mathcal{A}+\mathcal{A}=\mathcal{J}^C$	$2\mathcal{A}=\mathcal{J}$
Costs	$5[m]+3[s]$	$2[m]+4[s]$

Table 2: Average cost of doubling-free scalar multiplication algorithms for 160 bit integers.

Algorithm	Coordinate	# [m]
Fibonacci-and-add [15]	NewADD	2311
Signed Fib-and-add [15]	NewADD	2088
Window Fib-and-add [15]	NewADD	1960
EAC-320 [15]	NewADD	2112
GRAC-258	Mixed	1907

because we avoided the use of subtraction operation in the scalar multiplication; rather, we took inversion of a point where ever subtraction was involved. This facilitated us in proposing an elegant scalar multiplication algorithm. Note that the worst case of GRAC method includes 26 points in the storage, details shown in the appendix. These points are discarded once being used during the scalar multiplication process.

The new point addition formula (NewADD) proposed by Meloni [15] is applicable to addition chains which involves Fibonacci type of additions. The GRAC method lacks the continues Fibonacci pattern, hence we choose mixed coordinates to compute the cost for the proposed scalar multiplication algorithm. We have selected the best case of mixed coordinates [5] for addition and doubling operations to compute the cost of GRAC based algorithm as shown in Table 1. The total computational cost for GRAC based scalar multiplication algorithm involves $(\ell-1)$ additions and one doubling, which is given by the following formula.

$$\#[m]=(5[m]+3[s])(\ell-1)+(2[m]+4[s])\,.$$

In Table 2, we compare the efficiency of GRAC based algorithm with other doubling-free algorithms proposed in [15]. Considering the best case of GRAC-258, it is evident from Table 2 that our proposed algorithm has outperformed Fibonacci-and-add by 18%, signed Fib-and-add by 9%, Window Fib-and-add by 3% and EAC-320 by 10%.

6 Conclusion

In this paper we have proposed a SPA resistant scalar multiplication algorithm. Thus, we have proposed an explicit algorithm (GRAC method) for the previously proposed GRASC method to facilitate in presenting an elegant scalar multiplication algorithm. Our proposed GRAC based scalar multiplication algorithm has preceded other scalar multiplication algorithm by 3% to 18%. Further work may include finding chains of much shorter lengths in order to improve the computational cost of the GRAC based scalar multiplication algorithm. Also, if one could reduce the storage content, then GRAC based algorithm could be more applicable to elliptic curve cryptosystems where constraint memory devices such as smart cards needs to be implemented.

References

[1] Avanzi, R.M., Cohen, H., Doche, C., Frey, G., Lange, T., Nguyen, K., and Vercauteren, F., “Handbook of Elliptic and Hyperelliptic Curve Cryptography”. CRC Press, 2005.

[2] Blake, I.F., Seroussi, G., and Smart, N.P., “Elliptic Curves in Cryptography”. Number 256 in *London Mathematical Society Lecture Note Series*. Cambridge University Press, 1999.

[3] Bos, J., and Coster, M., “Addition chain heuristics”. *Advances in Cryptology-CRYPTO’89*, volume 435 of *Lecture Notes in Computing Science*, pages 400-407. Springer-Verlag, 1989.

[4] Byrne, A., Meloni, N., Crowe, F., Marnane, W.P., Tisserand, A., and Popovici, E.M., “SPA reisistant Elliptic Curve Cryptosystem using Addition Chains”. *International Conference on Information Technology-ITNG’07*, pp.995-1000, 2007.

[5] Cohen, H., Miyaji, A., and Ono, T., “Efficient elliptic curve exponentiation using mixed coordinates”. *Advances in Cryptology-ASIACRYPT’98*, volume 1514 of *Lecture Notes in Computing Science*, pages 51-65. Springer-Verlag, 1998.

[6] Fong, K., Hankerson, D., Lòpez, J., and Menezes, A., “Field inversion and point halving revisited”. *IEEE Transactions on Computers*, 53(8):1047-1059, Aug.2004.

[7] Goundar, R.R., Shiota, K., and Toyonaga, M., “New Strategy for Doubling-free Short Addition-Subtraction Chain”. *International Journal of Applied Mathematics*, volume 2, number 3, Dec. 2007.

[8] Hankerson, D., Menezes, A., and Vanstone, S., “Guide to Elliptic Curve Cryptography”. Springer-Verlag, 2004.

[9] Knuth, D., “Fundamental Algorithms”. *The Art of Computer Programming*, volume 1, Addision-Wesley,(1981).

[10] Knuth, D., “Seminumerical Algorithm (arithmetic)”. *The Art of Computer Programming*, volume 2, Addision-Wesley,(1981).

[11] Koblitz, N., "Elliptic curve cryptosystems". *Mathematics of Computation*, 48(177):203-209, Jan. 1987.

[12] Kocher, P., Jaffe, J., and Jun, B., "Differential power analysis". volume 1666 of *Lecture Notes in Computing Science*, pages 388-397. Springer-Verlag, 1999.

[13] Lou, D.C., and Chang, C.C., "An adaptive exponentiation method". *The Journal of Systems and Software*, volume 42, pp.59-69, 1998.

[14] Miller, V.S., "Uses of elliptic curves in cryptography". In H.C. Williams, editor, *Advances in Cryptology-CRYPTO'85*, volume 218 of *Lecture Notes in Computing Science*, pages 417-428. Springer-Verglag, 1986.

[15] Nicolas, M., "New Point Addition Formulae for ECC Applications". *Arithmetic of Finite Fields*, volume 4547 of *Lecture Notes in Computing Science*, pages 189-201. Springer-Verlag, 2007.

[16] Menezes, A.J., vanOorschot, P.C., and Vanstone, S.A., "Handbook of Applied Cryptography". CRC Press, 1997.

[17] Vorobiev, N., "Fibonacci Numbers". Birkhuser Verlag, 2002.

[18] Yacobi, Y., "Exponentiating faster with addition chains". *Advances in Cryptology-EUROCRYPT'90*, volume 473 of *Lecture Notes in Computing Science*, pages 222-229. Springer-Verglag, 1990.

Appendix

Here, we give an estimation of the storage capacity. That is the number of g_j's required in GRAC method for a 160 bit integer. Note that the number of g_j's are same as the number of new u_i being computed.

We have $\left\{u_0 = k\,, u_1 = [\frac{k}{\phi}]\,, u_2 = u_0 - u_1\,, u_3 = u_1 - u_2\,, \ldots\right\}$.

Note that Fibonacci sequence gives the optimum result, therefore we have a desire to maintain such pattern for the GRAC method. Thus, we will check at every step to maintain the property of a Fibonacci sequence given by equation (4).

For $i = 1$

$$\left|u_1 - \frac{k}{\phi}\right| \quad \leq \quad \frac{1}{2}\,.$$

For $i = 2$

$$\begin{aligned}
\left|u_2 - \frac{k}{\phi^2}\right| &= \left|u_0 - u_1 - \frac{k}{\phi^2}\right| \\
&= \left|k - u_1 - \frac{k}{\phi^2}\right| \\
&= \left|k - \frac{k}{\phi} - \frac{k}{\phi^2} + \frac{k}{\phi} - u_1\right| \\
&= \left|\frac{k}{\phi^2}(\phi^2 - \phi - 1) + \frac{k}{\phi} - u_1\right|\,, \\
&= \left|\frac{k}{\phi} - u_1\right| \text{(by Theorem 2.1)} \\
&\leq \frac{1}{2}\,.
\end{aligned}$$

For $i = 3$

$$\begin{aligned}
\left|u_3 - \frac{k}{\phi^3}\right| &= \left|u_1 - u_2 - \frac{k}{\phi^3}\right| \\
&= \left|u_1 - \frac{k}{\phi} + \frac{k}{\phi^2} - u_2 - \frac{k}{\phi^3} + \frac{k}{\phi} - \frac{k}{\phi^2}\right| \\
&= \left|(u_1 - \frac{k}{\phi}) + (\frac{k}{\phi^2} - u_2) - \frac{k}{\phi^3}(1 - \phi^2 + \phi)\right|\,, \\
&= \left|u_1 - \frac{k}{\phi}\right| + \left|\frac{k}{\phi^2} - u_2\right| \text{(by Theorem 2.1)} \\
&\leq \frac{1}{2} + \frac{1}{2} = 1\,.
\end{aligned}$$

Hence, in general we have

$$\left|u_i - \frac{k}{\phi^i}\right| \leq \left|u_{i-2} - \frac{k}{\phi^{i-2}}\right| + \left|u_{i-1} - \frac{k}{\phi^{i-1}}\right|\,.$$

Assuming MAXIMALGAP to be 11, we will check the number of terms exist before MAXIMALGAP is exceeded. It follows that

$$\begin{aligned}
i = 4: &\quad \tfrac{1}{2} + 1 &= \tfrac{3}{2}\,, \\
i = 5: &\quad 1 + \tfrac{3}{2} &= \tfrac{5}{2}\,, \\
i = 6: &\quad \tfrac{3}{2} + \tfrac{5}{2} &= 4\,, \\
i = 7: &\quad \tfrac{5}{2} + 4 &= \tfrac{13}{2}\,, \\
i = 8: &\quad 4 + \tfrac{13}{2} &= \tfrac{21}{2}\,, \\
i = 9: &\quad \tfrac{13}{2} + \tfrac{21}{2} &= 17\,.
\end{aligned}$$

The MAXIMALGAP is exceeded at $i = 9$, therefore at $i = 10$, a new u_i is computed. Hence, we could infer that in worst case, a new u_i is computed once in every 10th term for GRAC method. Therefore, the maximum number of points in the storage for an average chain of length 258 will be 26.