Elliptic Curve Scalar Multiplication Operation: a Survey Study

Ayat Waleed Khaled
Department of Mathematics, Faculty of Education for Girls,
University of Kufa,
Iraq
ayaatw.alkhazali@uokufa.edu.iq
Orcid.org/ 0009-0008-7739-0787

Najlae Falah Hameed Al Saffar
Department of Mathematics, Faculty of Computer Science and Mathematics,
University of Kufa,
Iraq
najlaa.hameed@uokufa.edu.iq
Orcid.org/0000-0001-5599-2885

DOI: http://dx.doi.org/10.31642/JoKMC/2018/100226
Received Apr. 30, 2023. Accepted for publication Jun.6, 2023

Abstract*—Scalar multiplication is the fundamental operation in the elliptic curve cryptosystem. It involves calculating the integer multiple of a specific elliptic curve point. It involves three levels: field, point, and scalar arithmetic. Scalar multiplication will be significantly more efficient overall if the final level is improved. By reducing the hamming weight or the number of operations in the scalar representation, one can raise the level of scalar arithmetic. This paper reviews some of the algorithms and techniques that improve the elliptic curve scalar multiplication in terms of the third level.*

Keywords— Elliptic curve; scalar multiplication; Signed binary method; NAF; ω − NAF; MOF; DRM and CRT.

I. INTRODUCTION

Since the mid-1980s, when Miller [1] and Koblitz [2] independently introduced the elliptic curve cryptosystem (ECC), Experimental evidence supports the claim that ECC offers the same level of security as RSA for smaller key sizes [4]. Researchers have been interested in the Elliptic curve scalar multiplication ($ECSM$) operation. The ECC was developed through the creation of $ECSM$ operation processes, which is considered the most important part of ECC as it is involved in the key generation, key exchange, encryption, decryption processing, and elliptic curve digital signature algorithm [6]. that is why there a lot of interest in this part. Many authors gave enhancements to this operation, see e.g. [7], [8], [9], [11], and [12]. among other works, have all been published.

The security of ECC depends on the difficulty of the mathematical hard problem $ECDLP$. This comes from the fact that no sub-exponential algorithms exist to solve it. Indeed, the efficiency of any algorithm is dependent on its running time. So it can be a polynomial time algorithm, exponential time algorithm, sub-exponential time algorithm, or fully exponential time algorithm. All these can be computed using the big O notation

Three computational levels are involved in the $ECSM$ operation's structure, as was already mentioned: In the scalar arithmetic level, efforts have been mainly focused on developing efficient algorithms for representing the scalar k that minimizes the number of operations needed to compute of the $ECSM$.

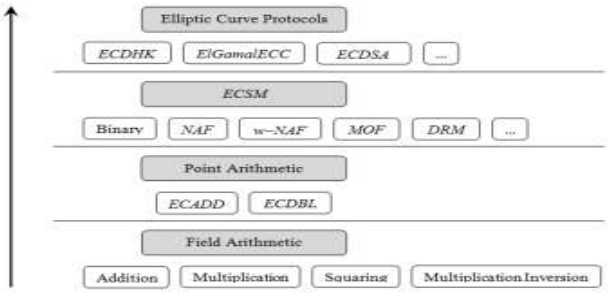

Fig 1. Mathematical ECC Operation Hierarchy

Six $ECSM$ algorithms were examined in this work. Some of the conventional scalar recoding algorithms have been investigated and examined including the binary method [10], the Non-Adjacent Form (NAF) method [13], Window $\omega -$ $-Non\ Adjacent\ Form(\omega - NAF$) method [14], Mutual Opposite Form (MOF) method [15], Direct Recoding Method (DRM) [16], and Complementary Recoding Technique (CRT) [17].

The findings of this study will be helpful for subsequent research and work. It will assist in identifying the future algorithm that is the most effective while also assisting in removing the current constraints. As a result, performance will be improved while maintaining the same level of security.

II. ECSM ALGORITHMS

This section examines and evaluates a few ECSM algorithms. To evaluate a scalar multiplication method, it is critical to consider the expected running time. As a result, when the algorithm is coded, it will be provided in detail.

A. Binary Method[5]

The first established exponentiation technique is a binary method, which is a technique for elliptic curve scalar multiplication. It is the computation of the form $Q = kP$ where P and Q are the elliptic curve points and k is an integer. It is accomplished by repeatedly doubling ($ECDBL$) and adding ($ECADD$) points on elliptic curves.

For a binary representation $(k_{n-1}\, k_{n-2} \dots k_1\, k_0)_2$ of an integer a k $k_i = \{0,1\}$, then $k = k_{n-1}2^{n-1} + k_{n-2}2^{n-2} + \cdots + k_1 2^1 + k_0$ and $k_i = \{0,1\}$, $i = 0,1,2, \dots, n-1$. That is $k = \sum_{i=0}^{n-1} k_i\, 2^i$, where $k_i = \{0,1\}$.

This method is called the binary method which scans the bits of k. The binary method for the computation of kP is given in the following:

Algorithm 1. Binary method for $ECSM$.
Input: $(k)_{10} = (k_{l-1}..k_1k_0)_2, P \in E(F_p)$.
Output: $Q = kP$.
1. $Q = P$.
2. For $i = l-1$ down to 0 do
2.1 $Q = 2Q$ $(ECDBL)$.
2.2 If $k_i = 1$ then $Q = Q + P$ $(ECADD)$.
3. Return Q.

Example 1. Let $\boldsymbol{k = 47}$ and $\boldsymbol{P}$ a point on the elliptic curve $\boldsymbol{E}$. Given the binary expansion of $\boldsymbol{k}$ as
$47 = 1.2^5 + 0.2^4 + 1.2^3 + 1.2^2 + 1.2^1 + 1.2^0 = (101111)_2$.

The scalar multiplication denoted by $\boldsymbol{47P}$ using Algorithm 1 would be as follows:
$47\,P = 2(2(\,2\,(\,2(\,2P + P) + P + P + P\,)$.
ECDBL is $l-1$, and ECADD is $\frac{l-1}{2}$. i.e., it requires 4 additions and 5 doublings.

In conclusion, whenever the bit is 1, two elliptic curve arithmetic operations $ECDBL$ and $ECADD$ will be performed and if it is 0, only one operation, $ECDBL$ is required. Thus, if we reduce the number of hamming weights $h(k)$ in the scalar representation, it could accelerate this computation.

The algorithm's running time is determined by how many operations are performed throughout its execution. In order to do that, it is needed to analyze each line of the algorithm in detail. For instance, line 1 of Algorithm 1 is equivalence and it is very fast and it is not included in the running time analysis. Similarly, "line 2" is the elimination of the loop and it is very fast and not inclined to running time. Next, if k_i is 1, then a point addition is performed in" line 2.1". Point addition needs some field operations; because of operation and included in the running time. Since the expected number of ones in the binary representation of k is half of its length $(l/2)$, line 2.1 is expected to run $l/2$ times. Finally, a doubling is performed in line 2.2 for each value of, that is, times cause of the same reasons of point addition needs some field operations; because of which, it is a heavy operation and included in the running time.

B. Non-Adjacent Form (NAF) Method

The hamming weight of the scalar k can be reduced with a signed representation that uses the numbers 0 and ± 1. Among various signed representations, NAF is a canonical representation with less number of hamming weights for integer k. The NAF representation of k was proposed in 1951by(booth).

[3]. And after 10 years (George w. Reitwiesner) [22] proved that every integer could be uniquely represented in this form.

The NAF of a scalar k denoted by $NAF(k)$ is the subject of various investigations in different contexts such as elliptic curve scalar multiplication. In 1989 [20] proposed a cryptographic application where the user of the NAF can reduce the number of multiplications in the elliptic curve scalar multiplication. [21] in 1989 proposed to apply the NAF to accelerate the elliptic curve scalar multiplication. [18] in 2000 developed a left to right NAF recoding algorithm. In the same authors [19] in 2002 proposed a representation that has the fewest numbers of non-zero digits of any modified form. Furthermore, there are several works dedicated to NAF representations such as [23],[24],…

The property of this representation is that, of any two consecutive digits, at most one is non zero, Moreover, the length of $NAF(k)$, denoted by $l(NAF(k))$ is at most 1 more bit than its binary representation. This means, fewer $ECADDs$ operations and therefore more efficiency when needed to compute the elliptic curve scalar multiplication. Now, because of the group law of the elliptic curve group [15], the inverse of $P = (x,y) \in E(F_p)$ is $-P = (x, -y\ (mod p)) \in E(F_p)$. Therefore, computing the inverse of any point on an elliptic curve is free and very fast in terms of computational time. That is, in the process of computing the kP, and the minus comes, subtraction of P is performed during this computation, furthermore, it costs the same amount of $ECADD$ operation in the total operation. The signed digit representation there is no unique signed digit representation for any integer k. To get this uniqueness one has to add some conditions on the representation. This condition will be that there are no adjacent non-zeros (using NAF). For example, the number 15 can have several signed-digit representations:
$(01111)_2 = 8 + 4 + 2 + 1 = 15,$
$(1\bar{1}111)_2 = 16 - 8 + 4 + 2 + 1\,1 = 15,$
$(10\bar{1}11)_2 = 16 - 4 + 2 + 1 = 15,$
$(1000\bar{1})_{NAF(15)} = 16 - 1 = 15.$
Only the last representation is NAF

Definition 1: A *NAF* of a positive integer k is a signed digit representation of k to the base $b = 2$, such that $k_i\, k_{i+1} = 0$ for $i \geq 0$. The *NAF*(*k*) is written as $(k_{l-1}\, k_{n-2} \dots k_1\, k_0)_{NAF(k)}$. This form exists and is unique for any integer k [11]. The hamming weight of the $NAF(k)$ is minimal among all signed digit representations of k [28]. Fortunately, the number of bits in the $NAF(k)$ is at most one more than the number of bits in the binary form of *k*. calculating the $NAF\ (k)$ of any integer k will illustrate in Algorithm 2 [3].

Algorithm 2: Computing $NAF(k)$ of an integer k
Input: A scalar $(k)_{10} = (k_{l-1}..k_1k_0)_2$
Output: $N = (k_{l-1}..k_1k_0)_{NAF(k)}$
$i = 1\,; c = k$
While $c > 0$
If c odd
$N(i) = 2 - (c \bmod 4)$
$c = c - N(i)$
Else
$N(i) = 0$
End if
$c = \frac{c}{2}\,; i = i + 1$
End while
Return N

Remarks: The performance of Algorithm 2 can be summarized as:

1. If k is an even integer, then 0 will be taken, and continue with $\frac{k}{2}$.
2. If $k \equiv 1(mod\ 4)$, then 1 will be taken, and continue with $\frac{k-1}{2}$ which is an even integer that guarantees a 0 in the next step.
3. If $k \equiv 3 \equiv -1(mod\ 4)$, then -1 will be taken, and continue with $\frac{k+1}{2}$ which is an even integer that guarantees a 0 in the next step.

This measure produces zero after each non-zero digit, which means this representation has a low hamming weight.

Example 2: NAF of $k = 47$ is $47 = (101111)_2$
$i = 1\,, c = k = 47$
If c odd
$N(1) = 2 - (47 \bmod 4)$,
$2 - 3 = -1$ $N = \bar{1}$
$C = c - N(1)$
$47 - (-1) = 48$
$c = \frac{c}{2} = \frac{48}{2} = 24,\ i = 1 + 1,$
$i = 2\,,\ c = 24$ *is even* $N = 0\bar{1}$
$C = \frac{c}{2} = \frac{24}{2} = 12\,,\ i = 2 + 1$
$i = 3\,,\ c = 12$ *is even* $N = 00\bar{1}$
$C = \frac{c}{2} = \frac{12}{2} = 6\,,\ i = 3 + 1$
$i = 4\,,\ c = 6$ *is even* $N = 000\bar{1}$
$C = \frac{c}{2} = \frac{6}{2} = 3\,,\ i = 4 + 1$
$i = 5\,,\ c = 3$ *is odd*
$N(5) = 2 - (3 \bmod 4)$
$2 - 3 = -1$
$c = c - N(5)$
$3 - (-1) = 4$ $N = \bar{1}000\bar{1}$
$C = \frac{c}{2} = \frac{4}{2} = 2$
$i = 6\,,\ c = 2$ *is even* $N = 0\bar{1}000\bar{1}$
$C = \frac{c}{2} = \frac{2}{2} = 1\,,\ i = 6 + 1$
$i = 7\,, c = 1$ *is odd*
$N(7) = 2 - (1 \bmod 4)$
$2 - 1 = 1$ $N = 10\bar{1}000\bar{1}$
$c = 1 - 1 = 0$
$47 = (101111)_2 = (10\bar{1}000\bar{1})_{NAF(k)}$

The mechanism to compute $NAF(47)$, according to Algorithm 2, is shown in" $Table$ 1".

Table 1. Computing a $NAF(47)$

i	c	$N(i)$	N
1	47	$\bar{1}$	$\bar{1}$
2	24	0	$0\bar{1}$
3	12	0	$00\bar{1}$
4	6	0	$000\bar{1}$
5	3	$\bar{1}$	$\bar{1}000\bar{1}$
6	2	0	$0\bar{1}000\bar{1}$
7	1	1	$10\bar{1}000\bar{1}$

The method for computing the $ECSM$ (kP) with the scalar k in the NAF form is as follows:

Algorithm 3: NAF method for $ECSM$
Input: $(k_{l-1}..k_1k_0)_{NAF(k)}$ $P \in E(F_p)$.
Output: $Q = kP$.
1. $Q = P$.
2. For $i = l - 1$ down to 0 do
2.1 $Q = 2Q$ $(ECDBL)$.
2.2 If $k_i = 1$ then $Q = Q + P$.
2.3 If $k_i = -1$ then $Q = Q - P$.
3. Return Q.

This algorithm performs ($l - 1$ ECDBL + $\frac{l-1}{3}$ ECADD) as average, where l is the length of $NAF(k)$.

Example 3: Let $k = 686$ and p a point on the elliptic curve. Now, the binary representation of k is (1010101110) so the cost is exactly equal to (9ECDBL + 5 ECADD). While the $NAF(686)$ is (10101010010), the cost is equal to (10 ECDBL + 4 ECADD).

Remark: In spite of the $NAF(k)$ having the low hamming weight, it will not if the binary representation of k already has the same hamming weight. As $(99)_2 = (1100011)_2 = (10\bar{1}00\bar{1}0\bar{1})_{NAF}$**.** $(1100011)_2P = 99P$, the cost of operations, while the cost of $(10\bar{1}00\bar{1}0\bar{1})_{NAF(99)}P = 10$ operations for any point P on the elliptic curve. This happens because the length of $NAF(k)$ is more than the length of k in binary form. In that case [c1] has produced an algorithm to prevent this action.

According to the running time Then the running time according to this assumption for all mentioned algorithms will be as follows For instance, "$line$ 1" of Algorithm 3 is equivalence and it is very fast and is not included in the running time analysis. Similarly, "$line$ 2" is a determination of the loop and it is very fast and not included in the running time. Next, if k_i is 1, then a point addition is performed in "line 2.1". Point addition needs some field operations; because this is a heavy operation and includes the running time.

C. Window $\boldsymbol{\omega}$ *−Non Adjacent Form* ($\omega - NAF$) *Method*

$\omega - NAF$ has been proposed by [5] in 1997 where $w \geq 2$, in 2002 [7] proved that the $\omega - NAF$ is optimal. A $\omega -$

NAF of a positive integer k is an expression $\sum_{i=0}^{l-1} k_i 2^i$, where each non zero k_i is odd, $|k_i| < 2^{\omega-1}$ and $k_{l-1} \neq 0$. For $\omega \geq 2$, at most one of any ω consecutive bits is non-zero.

The length of $\omega - NAF(k)$ does not exceed one more than the length of the binary form of k. And the hamming weight of $\omega - NAF(k)$ is $\frac{1}{\omega+1}$.

Algorithm 4: Computing $\omega - NAF(k)$ of a Scalar k
Input: A scalar $(k)_{10}$
Output: $N = (k_{l-1}..k_1k_0)_{NAF(k)}$
$i = 1$
While $k \geq 1$
If k odd
$N(i) = 2 - (k \; mods \; 2^{\omega})$
$k = k - N(i)$
Else
$N(i) = 0$
End if
$k = \frac{k}{2}$; $i = i + 1$
End while
Return N

The function $mods$ in Algorithm 4 is an extra code designed as follows:

Coding of $mods$ Function
If $k \; mods \; 2^{\omega} \geq \frac{2^{\omega}}{2}$
$N(i) = (k \; mods 2^{\omega}) - 2^{\omega}$
Else
$N(i) = (k \; mods 2^{\omega})$

Example 4: Let $k = 47$
i = 1 , k = 47 is odd

$N(1) = 2 - (47 \; mods \; 8), \quad 7 \geq \frac{2^3}{2}$
$N(1) = (47 \; mods \; 8) - 8$
$7 - 8 = -1 \quad N = \bar{1}$
$k = 47 - (-1) = 48$
$k = \frac{k}{2} = \frac{48}{2} = 24, \quad i = 1 + 1$
$i = 2, \; k = 24 \quad is \; even \quad N = 0\bar{1}$
$k = \frac{k}{2} = \frac{24}{2} = 12, \; i = 2 + 1$
$i = 3, \; k = 12 \quad is \; even \quad N = 00\bar{1}$
$k = \frac{k}{2} = \frac{12}{2} = 6, \; i = 3 + 1$
$i = 4, \; k = 6 \; is \; even \quad N = 000\bar{1}$
$k = \frac{6}{2} = 3, \; i = 4 + 1$
$i = 5, \quad k = 3 \quad is \; odd$
$N(5) = 2 - (3 \; mods \; 8)$
$= 3 \quad N = 3000\bar{1}$

The mechanism to compute $\omega - NAF(47)$, according to Algorithm 4, is shown in" $Table$ 2".

Table 2. Computing a $3 - NAF(47)$

i	k	$N(i)$	N
1	47	$\bar{1}$	$\bar{1}$
2	24	0	$0\bar{1}$
3	12	0	$00\bar{1}$
4	6	0	$000\bar{1}$
5	3	3	$3000\bar{1}$

Calculation $ECSM$ using $\omega - NAF(k)$ is a general version of the usual $NAF(k)$. Nevertheless, there is pre-computation in step. The cost of Algorithm 4 is as follows: for Lines 1 and 2 there is 1 $ECDBL$ and $2^{\omega-2} - 1$ $ECADD$ respectively; then as pre-computation cost, there is $1ECDBL + 2^{\omega-2} - 1ECADD$. The cost of line 2.1 is l $ECDBL$ while it is $\frac{1}{\omega+1}$ECADD. That is, the total cost of the algorithm is $(1 + l)ECDBL + (2^{\omega-2} - 1 + \frac{1}{\omega+1})$ $ECADD$, then $4ECDBL$ and $1ECADD$.

Table 3. Computing the cost of a $w - NAF(47)$

47	$47P$	Cost of computing $47P$
$(101111)_2$	$2(2(2(2(2P + P) + P) + P + P$	5 $ECDBL$ + $4ECADD$
$(10\bar{1}000\bar{1})_{NAF(47)}$	$2(2(2(2(2(2P - P) - P$	6 $ECDBL$ + $2ECADD$
$(3000\bar{1})_{3-NAF(47)}$	$2(2(2(3P+)3P$	4 $ECDBL$ + $1ECADD$

D. Mutual Opposite Form (MOF) Method

In 2004 [27] introduced an algorithm to compute the $ECSM$, called the mutual opposite form (MOF). They also proved that this form is unique for all positive integers. There are several works dedicated to MOF representations such as [16] [17] MOF satisfies the following properties:

1. Signs of adjacent non-zero regardless of 0 bits) are opposite.
2. M non-zero bits and the least non-zero bit are 1 and -1 respectively.

The idea of the MOF method is summarized by converting the binary string of the scalar k into signed digit representation by computing $mk = 2k - k$ where $(-)$ stands for a bitwise subtraction. Algorithm 5 is for the conversion of a scalar k into the MOF.

Algorithm 5: Computing MOF of a Scalar k
Input: A scalar $(k_{n-1}..k_{n-2}k_1k_0)_2$
Output: $MOF(k) = (mk_n \; mk_{n-1}..mk_1mk_0)_{MOF(k)}$
Set $mk_n = k_{n-1}$
For $i = n - 1$ down to 1
$mk_i = k_{i-1} - k_i$
$mk_0 = -k_0$
Return N

Example 5: Let $k = 47$
$47=(101111)_2$
$i = n - 1$
$i = 6 - 1 = 5, mk_n = k_{n-1} \; mk_6 = k_{6-1},$
$i = 5, \quad k_5 = 1, \quad mk_5 = k_4 - k_5$

$0-1=-1$ $N=1\bar{1}$
$i=4\,,k_4=0\,,mk_4=k_3-k_4$
$1-0=1$ $N=1\bar{1}1$
$i=3\,,k_3=1\,,mk_3=k_2-k_3$
$1-1=0$ $N=1\bar{1}10$
$i=2\,,k_2=1\,,mk_2=k_1-k_2$
$1-1=0$ $N=1\bar{1}100$
$i=1\,,k_1=1\,,mk_1=k_0-k_1$
$1-1=0$ $N=1\bar{1}1000$
$i=0\,,k_0=1\,,mk_0=-k_0$
$=-1$ $N=1\bar{1}1000\bar{1}$

The mechanism to compute $MOF(47)$, according to Algorithm 5, is shown in "$Table\ 4$ ".

Table 3. Computing a $MOF(47)$

i	k	mk_i	$MOF(k)$
6		1	1
5	1	$\bar{1}$	$1\bar{1}$
4	0	1	$1\bar{1}1$
3	1	0	$1\bar{1}10$
2	1	0	$1\bar{1}100$
1	1	0	$1\bar{1}1000$
0	1	$\bar{1}$	$1\bar{1}1000\bar{1}$

E. Direct Recoding Method (DRM)

In 2010 [28] proposed a new method to present the scalar k in a new form. This method is named the direct recording method (DRM), which has the lowest hamming weight. There are not many works dedicated to DRM representations since it was introduced, with one paper in 2012 by Thilagavathi and Rajeswari (2012). The idea comes from the fact that for any scalar k, there exist s such that $2^s < k < 2^{s+1}$. So, $k = (2^{s+1})_2 - (2^{s+1} - k)_2$.

These formulas have some queries. The following proposition will answer them

Example 6. Let $k = 47$
Then
$2^{5+1} > 47 > 2^5$
$2^6 > 47 > 2^5$
$47 = (2^6)_2 - (2^6 - 47)$
$(64)_2 - (64 - 47)$
$(64)_2 - (17)$
(1000000)-(10001)
So the DRA (47) is $(10\bar{1}000\bar{1})$.

Computing $DRM(47)$, it was decreased from 4 to 3 compared with the hamming weight of $(47)_2$. On the other hand, the hamming weight of DRM is the same compared with the hamming weight of $NAF(47)$ but using only a single operation of bitwise subtraction.

F. Complementary Recoding Technique (CRT)

CRT is one of the techniques to convert a number to a canonical signed binary representation that reduces the Hamming weight [28]. Given the binary representation of a scalar $k = (k_{l-1} \dots k_1 k_0)_2$ the procedure for converting a binary string into the signed binary string using complementary method ([9], [10]) is given below:

$$k = \sum_{i=0}^{l-1} k_i 2^i = (1000 \dots 0)_{(l+1)bits} - \bar{k} - 1$$

Where $\bar{k} = \overline{k_{l-1}}\ \overline{k_{l-2}}\ \overline{k_0}$
And $\bar{k_i} = 0 \quad if\ \ k_i = 1$
$\bar{k_i} = 1 \quad if\ \ k_i = 0$ for $i = 0,1, \dots, l-1$.

Example 6. Let $k = 47$
$(1000000)_{6+1} - (010000) - 1$
$= (10\bar{1}000\bar{1})$
$= 64 - 16 - 1$
$= 47.$

III. CONCLUSION

In the implementation of ECC scalar multiplication is not only the basic computation but also a time-consuming consuming operation. Its Operational efficiency directly determines the performance of ECC. In this paper six elliptic curve scalar multiplication algorithms have been discussed with examples. The efficiency of these algorithms will be summarized in "table 4" in terms of the number of operations the method is the most efficient $\omega - NAF$.

Table 4. The Complexity of Computing Elliptic Curve Scalar Multiplication

K	NAF	$\omega - NAF$	MOF	DRM	CRM
47	$6ECDBL + 2ECADD$	$4ECDBL + 1ECADD$	$6ECDBL + 3ECADD$	$6ECDBL + 2ECADD$	$6ECDBL + 2ECADD$
61	$6ECDBL + 2ECADD$	$6ECDBL + 1ECADD$	$6ECDBL + 3ECADD$	$6ECDBL + 2ECADD$	$6ECDBL + 2ECADD$
87	$7ECDBL + 3ECADD$	$7ECDBL + 2ECADD$	$7ECDBL + 5ECADD$	$7ECDBL + 2ECADD$	$7ECDBL + 3ECADD$
93	$7ECDBL + 3ECADD$	$7ECDBL + 2ECADD$	$7ECDBL + 4ECADD$	$7ECDBL + 2ECADD$	$7ECDBL + 3ECADD$

REFERENCES

[1] V. S. Miller, Use of elliptic curves in cryptography, Advances in Cryptology, Proceedings of CRYPTO'85, LNCS, 218 (1986), 417-426.

[2] N. Koblitz, Elliptic curve cryptosystem, Mathematics of Computation, 48 (1987) 203-209.

[3] Booth, A. D. A Signed Binary Multiplication Technique. The Quarterly Journal of Mechanics and Applied Mathematics(1951). 4 (2): 236–24.

[4] A.D. Booth, A signed binary multiplication technique, Journal of Applied Mathematics, 4(2) (1951), 236-240.

[5] G. W. Reitwiesner, Binary Arithmetic, Advances in computers, 1 (1960), 231-308.

[6] F. Morain, J. Olivos, Speeding up the computations on an elliptic curve using addition subtraction chains, RAIRO Theoretical Informatics and Applications, 24 (1990), 531-543.

[7] K. Okeya, Signed binary representations revisited, Proceedings of CRYPTO'04 (2004), 123-139.

[8] M. Joye, S. Yen, Optimal left to right binary signed digit recoding, IEEE Transactions on Computers, 49 (2000), 740-748.

[9] P. Balasubramaniam, E. Karthikeyan, Elliptic curve scalar multiplication algorithm using complementary ISSN

[10] Najlae. F.H: Development of Some Fast And Efficient Methods For Elliptic Curve Scalar Multiplication Over prime fields, Ph.D. Philosophy, Malaysia, Universiti Putra Malaysia, (2015), 39-57

[11] Kavin, B. P., & Ganapathy, S. (2021). A new digital signature algorithm for ensuring data integrity in the cloud using elliptic curves. *Int. Arab J. Inf. Technol.*, *18*(2), 180-190.

[12] Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., & Wustrow, E. (2014). Elliptic curve cryptography in practice. In *Financial Cryptography and Data Security: 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers 18* (pp. 157-175). Springer Berlin Heidelberg.

[13] Revathi, S. Cloud Data security based on Fuzzy Intrusion Detection system with Elliptic Curve Cryptography (ECC) using Non-Adjacent Form (NAF) Algorithm.

[14] Hossain, M. R., & Hossain, M. S. (2019, February). Efficient FPGA implementation of modular arithmetic for elliptic curve cryptography. In *2019 International conferencConferencericaElectricalr and communication engineering (ECCE)* (pp. 1-6). IEEE.

[15] Paliakou, A. Y., Kulikau, V. P., & Stsiapanau, A. A. (2019, November). Resistance projection welding of sheet metal without the formation of a mutual melting zone in the form of a cast nugget. In *International Conference on Aviamechanical Engineering and Transport (AviaENT 2019)* (pp. 264-270). Atlantis Press.

[16] Feldman, N., & Heffetz, O. A grant to every citizen: Survey evidence of the impact of a direct government payment in Israel. *National Tax Journal*, (2022). *75*(2), 229-263.

[17] Padhy, S., Shankar, T. N., & Dash, . A Comparison among Fast Point Multiplication Algorithms in Elliptic Curve Cryptosystem(2022).

[18] Okeya, K., Schmidt-Samoa, K., Spahn, C., & Takagi, T. (2004, August). Signed binary representations revisited. In *CRYPTO* (Vol. 2004, pp. 23-139).

[19] Maimuț, D., & Matei, A. C. 0(2022). Speeding-Up Elliptic Curve Cryptography Algorithms. *Mathematics*, *10*(19), 3676.

[20] Wu, J., Yu, L., & Khan, Z. (2023). How Do Mutual Dependence and Power Imbalance Condition the Effects of Technological Similarity on Post- acquisition Innovation Performance Over Time? *British Journal of Management*, *34*(1), 195-219.

[21] F. Morain and J. Olivos. Speeding up the computations on an elliptic curve addition-subtraction chains.Theoretical Informatics and Applications,24:531–543, 1990

[22] George W. Reitwiesner. Binary arithmetic.Advances in Computers,1:231–308, 1960

[23] Saffar, N. F. H. A., & Said, M. R. M. (2015). Speeding up the elliptic curve scalar multiplication using a non-adjacent form. *Journal of Discrete Mathematical Sciences and Cryptography*, *18*(6), 801-821.

[24] Masson, S., Sanso, A., & Zhang, Z. (2021). Bandersnatch: a fast elliptic curve built over the bls12-381 scalar field. *Cryptology ePrint Archive*.

[25] Bendimerad, M. Y., double Berrich, L., Masoud, M., Jaradat, Y., Jannoud, I., Jaglan, N., ... & Toulali, I. (2015). Communications Antenna and Propagation.

[26] Vahdati, Z., Yasin, S., Ghasempour, A., & Salehi, M. (2019). Comparison of ECC and RSA algorithms in IoT devices. *Journal of Theoretical and Applied Information Technology*, *97*(16).

[27] Poole, B., Ozair, S., Van Den Oord, A., Alemi, A., & Tucker, G. (2019, May). On variational bounds of mutual information. In *International Conference on Machine Learning* (pp. 5171-5180). PMLR.

[28] Anagreh, M., Vainikko, E., & Laud, P. (2020). Speeding Up the Computation of Elliptic Curve Scalar Multiplication Based on CRT and DRM. In *ICISSP* (pp. 176-184).