Research Journal of Information Technology

ISSN 1815-7432

Research Journal of Information Technology 7 (2): 80-100, 2015
ISSN 1815-7432 / DOI: 10.3923/rjit.2015.80.100
© 2015 Academic Journals Inc.

Performance Analysis of Signed-Digit {0,1,3}-NAF Scalar Multiplication Algorithm in Lopez-Dahab Model

Sharifah Md. Yasin, Ramlan Mahmod and Rozi Nor Haizan Nor
Faculty of Computer Science and Information Technology, University Putra Malaysia, UPM, Serdang, 43400, Selangor, Malaysia

Corresponding Author: Sharifah Md. Yasin, Faculty of Computer Science and Information Technology, University Putra Malaysia, UPM, Serdang, 43400, Selangor, Malaysia

ABSTRACT

Scalar multiplication is a major operation in an elliptic curve cryptosystem. It is the mostly costly and time consuming operations. This study proposes a new signed-digit {0,1,3}-NAF scalar multiplication algorithm for elliptic curve over binary field with the scalar multiplier in base 2 and using digits {0, 1, 3}. The digit 3 requires tripling operations in the execution of the scalar multiplication algorithm. Thus, a tripling formula is also proposed and the proof of the formula is presented in this study. Complexity analysis is carried out to compare the proposed scalar multiplication algorithm with the addition-subtraction algorithm. At average case analysis, the proposed scalar multiplication algorithm has better performance than the addition-subtraction algorithm exceptionally when only one digit 3 occurs in the scalar multiplier. When compared with traditional NAF scalar, the proposed scalar has better performance except when the Hamming weight and the bit-length of the proposed scalar and the traditional NAF are the same.

Key words: Elliptic curve cryptosystem, scalar multiplication, elliptic curve, binary field, Hamming weight

INTRODUCTION

Elliptic curve cryptography is proposed by Miller and Koblitz in mid 80's. Elliptic Curve Cryptosystem (ECC) is used to secure digital information. The security of an elliptic curve cryptosystem is based on Elliptic Curve Discrete Logarithm Problem (ECDLP) over the points on the elliptic curves. Security services provided by ECC are key agreement, digital signatures and encryption (Morales-Sandoval and Feregrino-Uribe, 2006). ECC is favourable in memory constraint devices since ECC uses shorter key size for the same security level with its competitor the RSA cryptosystem. Also, current mobile technology creates more challenges to ECC implementation (Paryasto *et al*., 2009).

In related studies, the major problem in the implementation of ECC is how to compute the scalar multiplication kP efficiently. The scalar multiplication (kP) is also a major operation in Elliptic Curve Cryptosystem (ECC). It is the most costly and time consuming operation. It is computed as Q = kP = P+P+... +P (k times), where P and Q are points on an elliptic curve E and scalar k is a very large integer. The operation is a consecutive sum of points that can be performed using elliptic curve point additions when the sum is using two different point (P+Q) and using elliptic curve point doublings when the sum is using the same point (P+P). Several methods have been developed to improve the scalar multiplication operation. Some of the method involves with recording the scalar into a different representation which gives result to a minimum number of nonzero digits in the scalar. Some method consists of rewriting or formulation of the point operation formula.

The objective of this study is to reduce the complexity of the scalar multiplication by recording the scalar into a new representation, formulation of a new tripling formula and introducing a new scalar multiplication algorithm that utilizes the proposed scalar and the proposed tripling.

ELLIPTIC CURVE IN LOPEZ-DAHAB MODEL

There are two types of elliptic curve that are widely used in cryptography: Binary GF(2^m) and prime GF(p) fields. Scalar multiplication involved with elliptic curve point operations and field operations. Elliptic curve point operations involved with point additions and point doublings. Whereas, elliptic curve field operation involved with multiplication, inversion, squaring and adding. In most research, either Lopez-Dahab or Jacobian coordinates is used. Both coordinates are free of inversion operation since inversion is the most costly field operation. Lopez-Dahab or Jacobian coordinates are also called projective coordinates.

This study mainly focuses on elliptic curve over binary field. A non-supersingular elliptic curve equation of characteristic 2 or defined over a binary field has the following form:

$$E: y^2+xy = x^3+ax^2+b \tag{1}$$

where, a, $b \in GF(2^m)$, $b \neq 0$. If P = (x, y), then the inverse of P is -P = (x, x+y).

For Lopez-Dahab coordinates, Eq. 1 is transformed to Lopez-Dahab projective form by substituting $x = X/Z$ and $y = Y/Z^2$, then by clearing the denominator, Lopez-Dahab projective equation is given below (Lopez and Dahab, 1999; Qingwei, 2008):

$$Y^2+XYZ = X^3Z+aX^2Z^2+bZ^4 \tag{2}$$

Every equivalent class of the triples (X, Y, Z), on the projective plane, with $Z \neq 0$, can be mapped back to the affine point by $x = X/Z^\alpha$ and $y = Y/Z^\beta$. There is only one point at the infinity O that can be represented with Z = 0. Also, there is only one element that can be represented with Z = 1, which is corresponding to the affine coordinates (Longa, 2007):

$$(X/Z^\alpha,\ Y/Z^\beta,\ 1) \tag{3}$$

Lopez-Dahab point addition: Consider $P = (X_0, Y_0, Z_0)$, $Q = (X_1, Y_1, Z_1)$ such that $P \neq \pm Q$ then $P \oplus Q = (X_2, Y_2, Z_2)$. Lopez and Dahab (1999) proposed a general addition formula that needs 14M+6S+8A, where M, S and A are multiplication, squaring and addition, respectively:

$$\begin{array}{lll} A_0 = Y_1Z_0^2 & D = B_0 + B_1 & G = D^2(F + aE^2) \\ A_1 = Y_0Z_1^2 & E = Z_0Z_1 & H = CF \\ B_0 = X_1Z_0 & F = DE & X_2 = C^2 + H + G \\ B_1 = X_0Z_1 & Z_2 = F^2 & J = D^2A_0 + X_2 \\ C = A_0 + A_1 & I = D^2B_0E + X_2 & Y_2 = HI + Z_2J \end{array} \tag{4}$$

Higuchi and Takagi (2000) improved Lopez-Dahab general addition formula to 13M+4S+9A, where, $P = (X_1, Y_1, Z_1)$, $Q = (X_2, Y_2, Z_2)$ and $P \oplus Q = (X_3, Y_3, Z_3)$ where:

$$\begin{array}{ll} A = X_1Z_2 & G = Y_1Z_2^2 \\ B = X_2Z_1 & H = Y_2Z_1^2 \\ C = A^2 & I = G + H \\ D = B^2 & J = IE \\ E = A + B & Z_3 = FZ_1Z_2 \\ F = C + D & \end{array}$$

$$\begin{array}{l} X_3 = A(H + D) + B(C + G) \\ Y_3 = (AJ + FG)F + (J + Z_3)X_3 \end{array} \tag{5}$$

Al-Daoud *et al.* (2002) proposed point addition formula for mixed affine-Lopez-Dahab coordinates. If $a \in \{0,1\}$, then only eight general multiplications are required. Consider $P = (X_1, Y_1, 1)$, $Q = (X_2, Y_2, Z_2)$ such that $P \neq \pm Q$, P is an affine and Q is a Lopez-Dahab coordinates, then, $P \oplus Q = (X_3, Y_3, Z_3)$ is given by:

$$\begin{array}{ll} A = Y_2 + Y_1Z_2^2 & Z_3 = C^2 \\ B = X_2 + X_1Z_2 & X_3 = A^2 + C(A + B^2 + aC) \\ C = BZ_2 & Y_3 = (D + X_3)(AC + Z_3) + (Y_1 + X_1)Z_3^2 \\ D = X_1Z_3 & \end{array} \tag{6}$$

Lopez-Dahab point doubling: The doubling formula is given by Lopez and Dahab (1999) and the cost is 5M+5S+4A. For a = 1, the following equations requires 4M+5S+3A. The doubling formula is given below:

If $P = (X_1, Y_1, Z_1)$ then $2P = 2(X_1, Y_1, Z_1) = (X_2, Y_2, Z_2)$ where:

$$\begin{array}{ll} A = Z_1^2 & Z_2 = AC \\ B = bA^2 & X_2 = C^2 + B \\ C = X_1^2 & Y_2 = (Y_1^2 + aZ_2 + B)X_2 + Z_2B \end{array} \tag{7}$$

Lange (2004) improved Lopez-Dahab doubling formula and it takes 5M+4S+5A including one multiplication by a. If $P = (X_1, Y_1, Z_1)$ then $2P = (X_2, Y_2, Z_2)$, the Eq. 8 is as the following:

$$\begin{array}{ll} S = X_1^2 & Z_2 = T^2 \\ U = S + Y_1 & X_2 = U^2 + R + aZ_2 \\ T = X_1Z_1 & Y_2 = (Z_2 + R)X_2 + S^2Z_2 \\ R = UT & \end{array} \tag{8}$$

REVIEW OF SIGN-DIGIT RECORDING

Scalar representation can be improved by recording the scalar to a different representation with a minimal Hamming weight. The Hamming weight is a measurement used on the number of nonzero digit in a scalar. This study focuses on signed numbers representation similar with the traditional NAF scalar. There are two modes of recordings; right-to-left and left-to-right recording. Selection of radix or digit set for a scalar must satisfies the characteristics of the scalar multiplication algorithm or implementation technology (Phillips and Burgess, 2004). Proper selection of radix and digit set for the scalar can promote an increment of the frequency of useful digits such as zero and a reduction in the total number of nonzero digits to represent a number.

Left-to-right recording is a real-time recording because the recorded scalar is used straight away for scalar multiplication algorithm. This is possible because scanning digits of the scalar for recording and scalar multiplication algorithm are done using the same mode, which is from left-to-right. In the literature, this type of recording promotes better memory usage and mostly preferred for memory constraint devices (Khabbazian *et al.*, 2005). Whereas, in the right-to-left recording, the recorded digits are saved before it is used in the scalar multiplication algorithm. This is because scanning digits of the scalar for recording and scalar multiplication algorithm initiated from different modes that is the recording mode is from right-to-left and the scalar multiplication mode is from left-to-right. Generally, this type of recording needs an additional n-bit RAM for storage, where n is the bit size of the scalar (Okeya *et al.*, 2004).

Right-to-left {-1,0,1}-NAF recording: Reitwiesner (1960) proposed a right-to-left NAF recording which converts a binary number into a traditional NAF using digit {-1,0,1}. In this study, it is labeled as {-1,0,1}-NAF to differentiate with the proposed {0,1,3}-NAF. NAF means non-adjacent form which ensures that there is no consecutive non-zero digit in the scalar k. In the {-1,0,1}-NAF recording, a binary number of the form $r = (r_{m-1}, ..., r_0)_2$ with $r_i \in \{0,1\}$, converted into a canonical form $r' = (r'_m, r'_{m-1}, ..., r'_0)_{SD2}$ with $r'_i \in \{\bar{1},0,1\}$ using Algorithm 1.

Algorithm 1: Right-to-left NAF recording

```
Input: A binary number $r = (r_{m-1},..., r_0)_2$
Output: $r' = (r'_m, r'_{m-1},...,r'_0)_{NAF}$
$c_0 \leftarrow 0; r_{m+1} \leftarrow 0; r_m \leftarrow 0$
for i from 0 to m do
    $c_{i+1} \leftarrow \lfloor (c_i + r_i + r_{i+1})/2 \rfloor$
    $r'_i \leftarrow c_i + r_i - 2c_{i+1}$
return $(r'_m, r'_{m-1}, ..., r'_0)_{NAF}$
```

Left-to-right recording: Joye and Yen (2000) proposed an optimal left-to-right signed-digit recording algorithm. This algorithm used look-up table as shown in Table 1. This recording method does not have NAF property but still has a minimal Hamming weight and as efficient as the Reitweisner's Algorithm 2.

Algorithm 2: Left-to-right minimal weight signed-digit recording

```
Input: $(r_{m-1},..., r_0)_2$
Output: $(r'_m, r'_{m-1},....,r'_0)_{SD2}$
$b_m \leftarrow 0; r_m \leftarrow 0; r_{-1} \leftarrow 0; r_{-2} \leftarrow 0;$
for i from m down to 0 do
    $b_{i-1} \leftarrow \lfloor (b_i + r_{i-1} + r_{i-2})/2 \rfloor$
    $r'_i \leftarrow -2b_i + r_i + b_{i-1}$
return $(r'_m, r'_{m-1},...., r'0)_{SD2}$
```

Table 1: Left-to-right signed-digit recording (Joye and Yen, 2000)

b_i	r_i	r_{i-1}	r_{i-2}	b_{i-1}	r'_i
0	0	0	x	0	0
0	0	1	0	0	0
0	0	1	1	1	1
0	1	0	x	0	1
1	0	1	x	1	$\bar{1}$
1	1	0	0	0	$\bar{1}$
1	1	0	1	1	0
1	1	1	x	1	0

x = 0 or 1

METHODOLOGY

In this study, a new scalar multiplication algorithm is proposed. In this study, the following steps are carried out: Propose a new scalar representation; propose a new point operation and propose a new scalar multiplication algorithm.

Propose a new scalar representation: In this study, we propose a recording technique based on Joye and Yen (2000) that converts a binary number into a new representation in base 2 with digit {0,1,3}. Mode of the recording is from Left-to-Right (LR). The new scalar adopted the Non-Adjacent Form (NAF) property. Consider a binary number, $r = (r_{m-1}, ..., r_0)_2$ where $r_i \in \{0,1\}$ and m is the bit length of r. Let h denotes the Hamming weight of r and p be the number of digit 3 in r where:

$$0 < p \leq \frac{m}{2}$$

Algorithm 3 (Yasin *et al.*, 2014) and Table 2 are used to convert a binary r into:

$$r' = \sum_{i-1}^{m} r_i' 2^i$$

where, $r_i' \in \{0,1,3\}$. Then, h_1 denotes the Hamming weight of r'. Also, r' can be written as $r' = (r_{m-1}', ..., r_0')_{(0,1,3)-NAF}$. Special case in Table 3 is a special s-block is defined for the case where $r = (r_{m-1}, ..., r_{i+1}, r_i, r_{i-1}, r_{i-2},..., r_{s-1}, r_s,..., r_1, r_0)_2$ such that $r_{i+1} = 0$ and $r_s = 0$. Therefore, s-block consists of consecutive digit '1' where $r_i = r_{i-1} = r_{i-2} = ... = r_{s-1} = 1$. This special case occurs when the count of digit '1' in the s-block is greater than or equal to 2.

Algorithm 3: Left-to-right {0,1,3}-NAF recording algorithm

```
Input: $r = (r_{m-1}, ..., r_0)_2$
Output: $r' = (r'_m, r'_{m-1}, ..., r'_0)_{(0,1,3)-NAF}$
$b_m \leftarrow 0$; $r_m \leftarrow 0$; $r_{-1} \leftarrow 0$; $r_{-2} \leftarrow 0$; $r'_m \leftarrow 0$
for i from m-1 down to 0 do
    $b_i \leftarrow \lfloor (b_{i+1}+r_i+r_{i-1})/2 \rfloor$
    if $[(b_{i+1}, r_{i+1}, r_i, r_{i-1}, b_i)$ = {(row1) or (row 3) or
(row 5) or (row 6) or (row 8) or (row 9) or (row 10) or (row 13) or (row 15)}] then
    if $[(b_{i+1}, r_{i+1}, r_i, r_{i-1}, b_i)$ = {(row 2) or (row 4) or (row 7)}] then $r'_i = 1$
    if $[(b_{i+1}, r_{i+1}, r_i, r_{i-1}, b_i)$ = {(row 11) or (row 12) or (row 14)}] then $r'_i = 3$
return $(r'_m, r'_{m-1}, ..., r'_0)$
```

Table 2: New left-to-right signed-digit {0,1,3}-NAF recording

Input						
b_{i+1}	r_{i+1}	r_i	r_{i-1}	b_i	Special case	r'_i
0	0	0	x	0		0
0	0	1	0	0		1
0	0	1	1	1	*(Count the number of consecutive '1' in the s-block)	
					if (((no. of '1')%2) = =0)	
					then $r'_i = 0$	0
0	0	1	1	1	*(Count the number of consecutive '1' in the s-block)	
					if (((no. of '1')%2) = =1)	
					then $r'_i = 1$	1
0	1	0	x	0		0
0	1	1	1	1	r'_{i+1} = 1 OR 3	0
1	0	1	0	1		1
1	0	1	1	1		0
1	1	0	0	0		0
1	1	0	1	1	r'_{i+1} = 1 OR 3	0
1	1	0	1	1	$r'_{i+1} = 0$	3
1	1	1	0	1	$r'_{i+1} = 0$	3
1	1	1	0	1	$r'_{i+1} = 1$	0
1	1	1	1	1	$r'_{i+1} = 0$	3
1	1	1	1	1	r'_{i+1} = 1 OR 3	0

x = 0 or 1 and $b_i = \lfloor (b_{i+1}+r_i+r_{i-1})/2 \rfloor)$

Table 3: Point operations using Lopez-Dahab and Jacobian coordinates for elliptic curve over binary field

	Cost operation	
Point operation	Lopez-Dahab	Jacobian
Addition	Lopez and Dahab (1999)	Cohen and Frey (2006)
	Cost = 14M+6S	Cost = 16M+3S
	Higuchi and Takagi (2000)	
	Cost = 13M+4S	
Doubling	Lopez and Dahab (1999)	Cohen and Frey (2006)
	Cost = 5M+5S	Cost = 5M+5S
	Lange (2004)	
	Cost = 5M+4S	
Mixed addition	Al-Daoud *et al.* (2002)	Dimitrov *et al.* (2005)
	Cost = 9M+5S	Cost = 15M+7S
General tripling	None	None

M: Multiplication and S: Squaring, addition

The proposed recording algorithm is implemented in Visual C++ ver 6.0. The conversion of a binary expansion to a new {0,1,3}-NAF representation is running successfully.

Hamming weight of the traditional {-1,0,1}-NAF scalar: Performance of a scalar is based on the Hamming weight (H_w) and bit-length of the scalar k. Heuberger and Prodinger (2007) studied a relation between the expected Hamming weight of the {-1,0,1}-NAF scalar when recorded from a binary expansion with length n and Hamming weight h. The expected Hamming weight of {-1,0,1}-NAF satisfies the following theorem (Heuberger and Prodinger, 2007):

- **Theorem 1:** The expected Hamming weight of {-1,0,1}-NAF is asymptotically equal to:

$$\left(\frac{1-4(\alpha-\frac{1}{2})^2}{3+4(\alpha-\frac{1}{2})^2}\right)\times n \quad (9)$$

Where:

$$\alpha \approx \frac{h}{n}$$

and α is in the interval (0,1). The expected Hamming weight of {-1,0,1}-NAF is equal to n/3 when:

$$\alpha = \frac{1}{2}$$

otherwise, the scalar is smaller.

- **Theorem 2:** If h is fixed and n tends to infinity, the expected Hamming weight of {-1,0,1}-NAF scalar is:

$$h + O(\frac{1}{n^2}) \quad (10)$$

- **Theorem 3:** If h is large (i.e., at least n/2), the expected Hamming weight of {-1,0,1}-NAF scalar is:

$$\frac{n}{3}+\frac{4}{9}+\frac{2\sqrt{2}(7+(-1)^n)}{9\pi}\times\frac{1}{\sqrt{n}}+O(\frac{1}{n}) \quad (11)$$

- **Theorem 4:** Consider the Hamming weight of binary and the Hamming weight {-1,0,1}-NAF scalars as two random vectors. Then, the covariance is:

$$\frac{2}{3}+O(\frac{n}{2^n})$$

The coordinates of the random vector asymptotically becomes independent and normally distributed.

Non-adjacency property: Based on the proving technique used by Joye and Yen (2000), each row in Table 3 is examined for the non-adjacency property, $r'_i . r'_{i+1} = 0$. The proof of non-adjacency property is as shown below (Yasin, 2011):

Row 1:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 0, 0, x)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, x, r_{i-2})$ and $r_{i-2} = 0$ or 1

- From Table 2, $r'_i = 0$ and $r'_{i-1} = 0$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 2:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 0, 1, 0)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 1, 0, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 1$ and $r'_{i-1} = 0$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 3:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 0, 1, 1)$
- (Refer special case)
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, r_{i-2})$ and $r_{i-2} = 1$
- From Table 2, $r'_i = 0$, we have $r'_{i-1} = 3$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 4:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 0, 1, 1)$
- (Refer special case)
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, r_{i-2})$ and $r_{i-2} = 1$
- From Table 2, $r'_i = 1$, then $r'_{i-1} = 0$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 5:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 1, 0, x)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, x, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ and four different cases for r'_{i-1}:

Case 1: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, 1, 0)$, then $r'_{i-1} = 1$
Case 2: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, 1, 1)$, then $r'_{i-1} = 0$ or 1
Case 3: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, 0, 0)$, then $r'_{i-1} = 0$
Case 4: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, 0, 1)$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 6:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (0, 1, 1, 1)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ and there are two cases for r'_{i-1}:

Case 1: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, 0)$, then $r'_{i-1} = 3$
Case 2: If $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, 1)$, then $r'_{i-1} = 3$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 7:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 0, 1, 0)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 0, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 1$ and $r'_{i-1} = 0$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 8:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 0, 1, 1)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ and $r'_{i-1} = 3$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 9:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 0, 0)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (0, 0, 0, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ and $r'_{i-1} = 0$
- Therefore, it is proven that $r'_i. r'_{i-1} = 0$

Row 10:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 0, 1)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 0, 1, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ or 3 and there are two cases for r'_i:

Case 1: For $r'_i = 0$ and $r'_{i-2} = 0$, then $r'_{i-1} = 1$
Case 2: For $r'_i = 3$ and $r'_{i-2} = 1$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 11:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 0, 1)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 0, 1, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, we have $r'_i = 0$ or 3, there are two cases for r'_{i-1}:

Case 1: For $r'_i = 0$ and $r'_{i-2} = 0$, then $r'_{i-1} = 1$
Case 2: For $r'_i = 3$ and $r'_{i-2} = 1$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 12:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 1, 0)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 0, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ or 3 and there are two cases for r'_{i-1}:

Case 1: For $r'_i = 0$ and $r'_{i-2} = 0$, then $r'_{i-1} = 0$
Case 2: For $r'_i = 3$ and $r'_{i-2} = 1$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 13:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 1, 0)$
- Hence $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 0, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ or 3 and there are two cases for r'_{i-1}:

Case 1: For $r'_i = 0$ and $r'_{i-2} = 0$, then $r'_{i-1} = 0$
Case 2: For $r'_i = 3$ and $r'_{i-2} = 1$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.

Row 14 and 15:

- $(b_{i+1}, r_{i+1}, r_i, r_{i-1}) = (1, 1, 1, 1)$
- Hence, $(b_i, r_i, r_{i-1}, r_{i-2}) = (1, 1, 1, r_{i-2})$ and $r_{i-2} = 0$ or 1
- From Table 2, $r'_i = 0$ or 3 and there are two cases for r'_i:

Case 1: For $r'_i = 0$ and $r'_{i-2} = 0$, then $r'_{i-1} = 3$
Case 2: For $r'_i = 3$ and $r'_{i-2} = 1$, then $r'_{i-1} = 0$

Therefore, it is proven that $r'_i. r'_{i-1} = 0$.
From the above, each row in Table 2 is proven to have the non-adjacency property.

Proposed {0,1,3}-NAF vs. traditional {-1,0,1}-NAF scalars: Analysis of the scalar k is carried out using 144 data of binary number with 24 bit-length. Each binary data (k) is converted into traditional {-1,0,1}-NAF and proposed {0,1,3}-NAF scalars by using Algorithm 1 and 3, respectively. Then, at average case, the Hamming weight and bit-length of each data is compared and analyzed. The average case is defined as:

$$\frac{h}{l} = \frac{1}{2}$$

where, h is the Hamming weight of binary and l is the corresponding bit-length. Table 4 shows the average case for Hamming weight differences after conversion from binary.

Table 4: Hamming weight of {0,1,3}-NAF and {-1,0,1}-NAF after conversion from binary of l = 24 bit-length (Average case)

p	h	h_1	h_2
1	12	11	12
2	12	10	12
3	12	9	9
3	12	9	10
3	12	9	12
3	12	9	9
3	12	9	10
3	12	9	12
3	12	9	9
3	12	9	12
3	12	9	9
4	12	8	12
5	12	7	7
5	12	7	10
5	12	7	10
5	12	7	7
6	12	6	10
6	12	6	8
6	12	6	9

p = No. of digit 3 in the {0,1,3}-NAF, h, h_1 and h_2 are Hamming weight of binary, {0,1,3}-NAF and {-1,0,1}-NAF, respectively

This data are derived from 144 data and for values h = 12. In Table 4, as we go down the table p varies from 1-6 and 6 is the maximum value of p when h = 12. The values of h and l are fixed so that the variation of p, h_1 and h_2 can be monitored. Cases for the Hamming weight $h_1 = h_2$ only occur for data No. 3, 6, 9, 11, 13 and 16. Whereas, the remaining data are having the Hamming weight $h_1<h_2$. This indicates an improvement in the Hamming weight when using the proposed scalar.

Propose new point tripling operation: This section focuses on elliptic curve point operations. Research development on point operations mostly involved with reducing the number of multiplications in the point operation. Some research involved with formulation of new formula with reduced number of multiplications. Review on Lopez-Dahab and Jacobian coordinates for elliptic curve over binary field are shown in Table 3.

Traditionally, a tripling can be computed as 3P = 2P+P where one doubling (Eq. 8) followed by one general addition (Eq. 5). The cost is (5M+4S)+(13M+4S) = (18M+8S). In this study, a tripling operation is proposed using one doubling (Eq. 8) followed by one mixed addition (Eq. 6). The mixed addition (Al Daoud *et al.*, 2002) is using affine and Lopez-Dahab coordinates since it has better cost than the general addition. Theoretically, the cost for the proposed tripling is given below:

$$(5M+4S)+(9M+5S) = (14M+9S) \tag{12}$$

When Eq. 12 is compared with the cost of traditional tripling, the tripling with mixed addition has better cost than the traditional tripling.

The proposed point tripling formula is shown below. Let P = $(X_1, Y_1, 1)$, then, 3P = 2P+P = $(X_2, Y_2, Z_2)+(X_1, Y_1, 1) = (X_3, Y_3, Z_3)$ where:

$$\begin{aligned}
A &= (Z_2 + E)X_2 + Z_2^2U \\
B &= X_2 + X_1Z_2 \\
C &= BZ_2 \\
D &= X_1Z_3 \\
E &= UX_1 \\
Z_2 &= X_1^2 \\
Z_3 &= C^2 \\
X_2 &= U^2 + E + aZ_2 \\
X_3 &= A^2 + C(A + B^2) + aZ_3 \\
Y_3 &= (D + X_3)(AC + Z_3) + (Y_1 + X_1)Z_3^2
\end{aligned} \tag{13}$$

By treating the field addition cost as negligible, the cost of Eq. 13 is (12M+7S). When compared with Eq. 12, it saved 2M+2S. Also, when compared with traditional tripling, the new formula saved 6M+1S. For NIST curve, the value of a is equal to 1 and the cost of the proposed tripling formula becomes (10M+7S).

Proof of the proposed tripling formula: We need to show that the proposed tripling formula is equivalent with the affine tripling formula. Since there is no existing tripling formula in affine, we need to derive the affine formula first, then we will compare the formula with our tripling formula and they should be equivalent.

Consider P = (x_1, y_1) and Q = (x_2, y_2), where P and Q are affine coordinates in elliptic curve over binary field, Q ≠ -P and Q ≠ P. Then, R = (x_3, y_3) = P+Q has the following Eq. 14 (Cohen and Frey, 2005):

$$\begin{aligned}
\lambda &= (y_2 + y_1)/(x_2 + x_1) \\
x_3 &= \lambda^2 + \lambda + x_1 + x_2 + a \\
y_3 &= \lambda(x_1 + x_3) + x_3 + y_1
\end{aligned} \tag{14}$$

If Q = P, then R = P+P = 2P is called as elliptic curve point doubling has the following Eq. 15 (Cohen and Frey, 2006):

$$\begin{aligned}
\lambda &= x_1 + y_1/x_1 \\
x_3 &= \lambda^2 + \lambda + a \\
y_3 &= (x_1 + x_3)\lambda + x_3 + y_1
\end{aligned} \tag{15}$$

For affine coordinates, to generate an affine tripling formula, we need one doubling (Eq. 15), followed by one addition (Eq. 14). Then, consider P = (x_1, y_1), then using Eq. 15, then 2P = (x_2, y_2) is as the following:

$$\begin{aligned}
x_2 &= \lambda_1^2 + \lambda_1 + a \\
y_2 &= (x_1 + x_2)\lambda_1 + x_2 + y_1 \\
\lambda_1 &= \frac{y_1 + x_1^2}{x_1}
\end{aligned} \tag{16}$$

Secondly, using Eq. 14 and 16 compute point tripling as:

$$3P = 2P + P = (x_2, y_2) + (x_1, y_1) = (x_3, y_3)$$

Then, the tripling in affine 3P = (x_3, y_3) is given below:

$$\begin{aligned} x_3 &= \lambda_2^2 + \lambda_2 + x_1 + x_2 + a \\ y_3 &= (x_1 + x_3)\lambda_2 + x_3 + y_1 \\ \lambda_2 &= \frac{y_1 + y_2}{x_1 + x_2} \end{aligned} \tag{17}$$

Thirdly, using Eq. 13, we need to prove that:

$$\frac{X_3}{Z_3} = x_3$$

and:

$$\frac{Y_3}{Z_3^2} = y_3$$

The processes are shown below:

$$\begin{aligned} \frac{X_3}{Z_3} &= \frac{A^2 + C(A + B^2) + aZ_3}{C^2} = \frac{A^2}{C^2} + \frac{A}{C} + \frac{B^2}{C} + a \\ &= \frac{[(Z_2 + E)X_2 + Z_2^2U]^2}{(X_2 + X_1Z_2)^2 Z_2^2} + \frac{[(Z_2 + E)X_2 + Z_2^2U]}{(X_2 + X_1Z_2)Z_2} + \frac{(X_2 + X_1Z_2)^2}{(X_2 + X_1Z_2)Z_2} + a \\ &= \frac{(Y_2 + Y_1Z_2)^2}{(X_2 + X_1Z_2)^2 Z_2^2} + \frac{(Y_2 + Y_1Z_2)}{(X_2 + X_1Z_2)Z_2} + \frac{(X_2 + X_1Z_2)^2}{(X_2 + X_1Z_2)Z_2} + a \end{aligned}$$

Use $Z_1 = 1$, also:

$$\lambda_2 = \frac{y_1 + y_2}{x_1 + x_2}$$

Then:

$$\begin{aligned} x_3 &= \frac{(y_1 + y_2)^2}{(x_1 + x_2)^2} + \frac{(y_1 + y_2)}{(x_1 + x_2)} + \frac{(x_1 + x_2)^2}{(x_1 + x_2)} + a \\ x_3 &= \lambda_2^2 + \lambda_2 + x_1 + x_2 + a(\text{proven}) \end{aligned} \tag{18}$$

$$\frac{Y_3}{Z_3^2} = \frac{(D+X_3)(AC+Z_3)+(Y_1+X_1)Z_3^2}{Z_3^2}$$
$$= \frac{(X_1Z_3+X_3)\left[(Y_2+Y_1Z_2^2)BZ_2+Z_3\right]+(Y_1+X_1)Z_3^2}{(X_2+X_1Z_2)^4Z_2^4}$$
$$= \frac{(X_1Z_3+X_3)\left[(Y_2+Y_1Z_2^2)BZ_2\right]}{(X_2+X_1Z_2)^4Z_2^4} + \frac{(X_1Z_3+X_3)Z_3}{(X_2+X_1Z_2)^4Z_2^4} + Y_1 + X_1$$
$$= \frac{(X_1Z_3+X_3)\left[(Y_2+Y_1Z_2^2)\right]}{(X_2+X_1Z_2)^3Z_2^3} + \frac{(X_1Z_3+X_3)}{(X_2+X_1Z_2)^2Z_2^2} + Y_1 + X_1$$

Use $X_3 = x_3 . Z_3$, then substitute $Z_2 = 1$ and $Z_3 = 1$:

$$y_3 = \frac{(x_1+x_3)(y_1+y_2)}{(x_1+x_2)} + x_1 + x_3 + y_1 + x_1 \tag{19}$$
$$y_3 = (x_1+x_3)\lambda_2 + x_3 + y_1 \text{(proven)}$$

Thus, the tripling Eq. 13 in Lopez-Dahab coordinates is the same as the affine tripling formula in Eq. 17. Algorithm 4 for the proposed tripling is shown below:

Algorithm 4: Point tripling with mixed addition

```
Input: Two points P = ($X_1$, $Y_1$, 1) and 2P = ($X_2$, $Y_2$, $Z_2$) where, P is in affine and 2P is in Lopez-Dahab coordinates.
Output: Point 3P = ($X_3$, $Y_3$, $Z_3$)
1.  $T_1 \leftarrow X_1$
2.  $T_2 \leftarrow Y_1$
3.  $Z_2 \leftarrow (T_1)^2$
4.  $T_3 \leftarrow Z_2+T_2$
5.  $T_4 \leftarrow T_3*T_1$
6.  $T_5 \leftarrow a$
7.  $X_2 \leftarrow (T_3)^2$
8.  $X_2 \leftarrow X_2+T_4+(T_5*Z_2)$
9.  $T_6 \leftarrow (Z_2)^2$
10. $T_6 \leftarrow T_6*T_3$
11. $T_6 \leftarrow T_6+(Z_2+T_4)*X_2)$
12. $T_4 \leftarrow X_2+(T_1*Z_2)$
13. $Z_2 \leftarrow T_4*Z_2$
14. $T_3 \leftarrow (T_6)^2$
15. $T_4 \leftarrow (T_4)^2$
16. $T_4 \leftarrow T_4+T_6+(T_5*Z_2)$
17. $T_4 \leftarrow T_4*Z_2$
18. $X_3 \leftarrow T_4+T_3$
19. $Z_3 \leftarrow (Z_2)^2$
20. $T_4 \leftarrow T_1*Z_3$
21. $T_6 \leftarrow (T_6*Z_2)+Z_3$
22. $T_4 \leftarrow T_4+X_3$
23. $T_6 \leftarrow T_6*T_4$
24. $T_3 \leftarrow (Z_3)^2$
25. $T_3 \leftarrow (T_1+T_2)*T_3$
26. $Y_3 \leftarrow T_6+T_3$
27. return (3P)
```

Propose new scalar multiplication algorithm: In this study, a new scalar multiplication algorithm is proposed namely signed-digit {0,1,3}-NAF scalar multiplication algorithm.

Algorithm 5: New signed-digit {0,1,3}-NAF scalar multiplication

```
Input: k is recorded as {0,1,3}-NAF,
$k = \sum_{i=0}^{m-1} b_i 2^i$ where, $b_i \in \{0,1,3\}$, $P \in E(F_{2^n})$
Output: kP
1.  P: = P(x1,y1)
2.  3P: = Tripling (P)
3.  if ($b_{m-1}$ = 1) then
4.  Q: = P
5.  if ($b_{m-1}$ = 3) then
6.  Q: = 3P
7.  for i from m-2 down to 0 do
8.  Q: = double (Q)
9.  if $b_i$ = 1 then
10. Q: = add (P,Q)
11. if $b_i$ = 3 then
12. Q: = add (3P,Q)
13. return (Q = kP)
```

In Algorithm 5, line 7 performs exactly m-1 times. In general, if the Hamming weight of the {0,1,3}-NAF scalar is h_1, then the expected running time of Algorithm 5 is given below:

$$\text{Cost (Algorithm 5)} = (h_1\text{-}1)\ A+(m\text{-}1)\ D+1T \quad (20)$$

where, h_1 and m are the Hamming weight and bit length of the signed-digit {0,1,3}-NAF scalar, respectively. Aso, A, D and T are point addition, doubling and tripling, respectively. Equation 20 is proven true when it is tested on 144 data.

Algorithm 6: Addition-subtraction algorithm

```
Input: $NAF(k) = \sum_{i=0}^{l-1} k_i 2^i$ and $P \in E(F_q)$
Output: Q = kP, where, $Q \in E(F_q)$
1. Q←∞
2. for i = l-2 downto 0
3. Q = 2Q
4. if ($k_i$ = 1) then Q = Q+P
5. if ($k_i$ = -1) then Q = Q-P
6. return Q
```

In Algorithm 6, line 3 performs exactly l-1 times. In general, if the Hamming weight of the {-1,0,1}-NAF scalar is h_2, then the expected running time of Algorithm 6 is given below:

$$\text{Cost (Algorithm 6)} = (h_2\text{-}1)A+(\ell\text{-}1)D \quad (21)$$

where, A and D are point addition and point doubling, respectively.

Complexity analysis: Complexity analysis provides running time estimation for performing scalar multiplication for any system specification with any parameters (Sullivan, 2007). The complexity

Table 5: Estimation cost of the addition-subtraction and the proposed scalar multiplication algorithms at average case (h and l are fixed)

					Estimation cost (μsec)		
p	h_1	h_2	l_1	l_2	Addition-subtraction	Proposed signed-digit {0,1,3}-NAF	Cost reduction (%)
1	11	12	24	24	49211.3	49894.0	-1.4
2	10	12	24	24	49211.3	47964.8	2.5
3	9	9	24	24	43423.8	46035.6	-6.0
3	9	10	23	25	46570.0	44818.7	3.8
3	9	12	24	25	50428.3	46035.6	8.7
4	8	12	24	25	50428.3	44106.5	12.5
5	7	7	24	24	39565.5	42177.3	-6.6
5	7	10	24	24	45353.0	42177.3	7.0
6	6	10	23	25	46570.0	39031.2	16.2
6	6	8	23	25	42711.6	39031.2	8.6
6	6	9	23	25	44640.8	39031.2	12.6

is based on the number of point operations involved per scalar throughout the execution of the scalar multiplication algorithm.

In scalar multiplication operation, the traditional {-1,0,1}-NAF scalar using addition-subtraction algorithm (Hankerson *et al.*, 2004) and the proposed {0,1,3}-NAF scalar using the proposed signed-digit {0,1,3}-NAF scalar multiplication algorithm. In this study, complexity analysis is carried out to compare performance of the proposed scalar multiplication algorithm with the traditional addition-subtraction algorithm.

For complexity analysis, there same 144 random binary data of 24 bit-length. Cost of the Algorithm 5 and 6 are measured based on the expected cost in Eq. 20 and 21. For average case analysis, only data with h = 12 are selected and displayed in 5. Cost estimation in Table 5 is derived by using the following point operation cost:

$$A = 9M+5S;\ D = (5M+4S) \text{ and } T = (12M+7S)$$

Also, the following field operation cost of Multiplication (M) and Squaring (S) as used in Edoh (2009):

$$M = 148.34 \text{ microseconds};\ S = 118.82 \text{ microseconds}$$

(p = No. of digit 3; h_1, h_2 = Hamming weight of the proposed {0,1,3}-NAF and {-1,0,1}-NAF, respectively; l_1, l_2 = bit size of the proposed {0,1,3}-NAF and {-1,0,1}-NAF, respectively).

RESULTS

Table 4 provides overall cost estimation for addition-subtraction and the proposed scalar multiplication algorithms. The values of p, Hamming weight and bit length of a scalar k are significant parameters in the performance of the scalar multiplication algorithm. To study the behavior of these parameters, the values of h and l are fixed (i.e., h = 12 and l = 24).

Based on Table 5, the value of p is increasing from 1 to 6, that is from data (a) until data (k). The variation of h_1, h_2, l_1 and l_2 are monitored. For all cases, the values of h_1 are lower than h_2, (i.e., $h_1<h_2$), indicate that the Hamming weight of the proposed {0,1,3}-NAF scalar is better than

the Hamming weight of the traditional {-1,0,1}-NAF. Also, by observation, the values of h_1 are decreasing from data (a) to data (k). For some cases, the values of l_1 are lower than l_2 indicate that the bit length of the proposed {0,1,3}-NAF scalar is better than the bit length of the {-1,0,1}-NAF scalar. By observation, for cases $l_1 = l_2$, means that there is no changes in bit length after the digit conversion.

From Table 5, the percentage of cost reduction helps to identify significant improvement of the scalar multiplication algorithms. The highest percentage of cost reduction is 16.2, which happens when p = 6. Negative percentage of cost reduction of -1.4, -6 and -6.6 indicate no improvement in the cost of the proposed algorithm. The value p = 1 may be the reason for the negative value in percentage of cost reduction in data (a). Thus, the value $h_1 = h_2$ and $l_1 = l_2$ may be the reason for the negative values in the percentage of cost reduction in data (c) and (g).

Cases where, $l_1 < l_2$ (i.e., $l_1 = 23$) happens in data (d), (i), (j) and (k). Table 6 shows tracing of Algorithm 5 and 6 using data (d). In Table 6, Algorithm 6 has 24 iterations whereas Algorithm 5 has 22 iterations. The same cases happen for data (i), (j) and (k). In conclusion, reduction of bit length (i.e., $l_1 = 23$) occurs because 3 is the MSD for the proposed {0,1,3}-NAF scalar. At the same

Table 6: Tracing algorithms using p = 3; $h_1 = 9$; $h_2 = 10$; $l_1 = 23$; $l_2 = 25$

Addition-subtraction (Algorithm 6)		{0,1,3}-NAF scalar multiplication (Algorithm 5)	
i	$\mathrm{NAF}(k) = \sum_{i=0}^{l-1} k_i 2^i$ $k_i \in \{-1, 0, 1\}$ k = 10-10010-100-10010-100-1000101	i	{0, 1, 3}-NAF, $k = \sum_{i=0}^{l-1} b_i 2^i$ $b_i \in \{0,1,3\}$ k = 30001010300010103000101
24	P	22	3P
23	2P	21	2 (3P) = 6P
22	2 (2P)-P = 3P	20	2 (6P) = 12P
21	2 (3P) = 6P	19	2 (12P) = 24P
20	2 (6P) = 12P	18	2 (24P)+P = 49P
19	2 (12P)+P = 25P	17	2 (49P) = 98P
18	2 (25P) = 50P	16	2 (98P)+P = 197P
17	2 (50P)-P = 99P	15	2 (197P) = 394P
16	2 (99P) = 198P	14	2 (394P)+3P = 791P
15	2 (198P) = 396P	13	2 (791P) = 1582P
14	2 (396P)-P = 791P	12	2 (1582P) = 3164P
13	2 (791P) = 1582P	11	2 (3164P) = 6328P
12	2 (1582P) = 3164P	10	2 (6328P)+P = 12657P
11	2 (3164P)+P = 6329P	9	2 (12657P) = 25314P
10	2 (6329P) = 12658P	8	2 (25314P)+P = 50629P
9	2 (12658P)-P = 25315P	7	2 (50629P) = 101258P
8	2 (25315P) = 50630P	6	2 (101258P)+3P = 202519P
7	2 (50630P) = 101260P	5	2 (202519P) = 405038P
6	2 (101260P)-P = 202519P	4	2 (405038P) = 810076P
5	2 (202519P) = 405038P	3	2 (810076P) = 1620152P
4	2 (405038P) = 810076P	2	2 (1620152P)+P = 3240305P
3	2 (810076P) = 1620152P	1	2 (3240305P) = 6480610P
2	2 (1620152P)+P = 3240305P	0	2 (6480610P)+P = 12961221P
1	2 (3240305P) = 6480610P		
0	2 (6480610P)+P = 12961221P		
	Cost = 9mA+24D		Cost = 8mA+22D+1T

time, increment of bit-length (i.e., $l_2 = 25$) occurs in data (d), (i), (j) and (k) for the traditional NAF scalar. Thus, when digit 3 is at the MSD, the proposed scalar multiplication algorithms has less number of iteration than the addition-subtraction algorithm. Thus, the proposed scalar multiplication algorithm achieved higher percentage of cost reduction than in normal cases.

DISCUSSION

Generally, at average case:

- The cost of the proposed scalar multiplication algorithm is better than the cost of the addition-subtraction method exclusive for the following cases:

Case 1: When $p = 1$
Case 2: When $h_1 = h_2$ and $l_1 = l_2$

- When the Most Significant Digit (MSD) of the proposed {0,1,3}-NAF scalar is 3:
- Reduction of bit length occurs in l_1 (i.e., $l_1 < l$)
- The proposed scalar multiplication algorithm has less number of iteration than the addition-subtraction algorithm illustrated in Table 6. Thus, the percentage of cost reduction is better than in normal cases

From Table 5, parameters h_1 and h_2 affect the cost of the scalar multiplication algorithm. Therefore, the following mathematical analysis is carried out in order to identify the behavior of h_1 and h_2.

From Eq. 20:

$$\Rightarrow \text{Cost (Algorithm 5)} = (9h_1+5l_1-2)M+(5h_1+4l_1-2)S \tag{22}$$

where, $l_1 = l$ or $l-1$ and $h_1 = h-p$. The case $l_1 = l-1$ happens in Table 5 for data (d), (i), (j) and (k). For this case, the proposed scalar multiplication algorithm has l-1 number of instructions.

From Eq. 21:

$$\Rightarrow \text{Cost (Algorithm 6)} = (h_2-1)\,A+(l_2-1)\,D \tag{23}$$

The point operation cost such as mixed Addition (A), Doubling (D) and Tripling (T) are (9M+5S), (5M+4S) and (12M+7S), respectively.

Then, from Eq. 23:

$$\begin{aligned}\text{Cost (Algorithm 6)} &= (h_2-1)(9M+5S)+(l_2-1)(5M+4S)\\ &= 9(h_2-1)M+5(h_2-1)S+5(l_2-1)M+4(l_2-1)S\\ &= (9h_2+5l_2-14)M+(5h_2+4l_2-5)S\end{aligned} \tag{24}$$

where, $l_2 = l$ or $l+1$. The case $l_2 = l+1$ happens in Table 5, for data (d), (e), (f), (i), (j) and (k). For this cases, the addition-subtraction algorithm has l+1 instructions to be executed.

Suppose that:

$$\text{Cost (Algorithm 5)} < \text{Cost (Algorithm 6)}$$

Then, from Eq. 22 and 24:

$$\begin{gathered}\Rightarrow(9h_1+5l_1-2)M+(5h_1+4l_1-2)S<(9h_2+5l_2-14)M+(5h_2+4l_2-5)S\\ \Rightarrow(9h_1+5l_1-9h_2-5l_2+12)M<(5h_2+4l_2-5h_1-4l_1-3)S\\ \Rightarrow(9h_1+5l_1-9h_2-5l_2+12)M<(5h_2+4l_2-5h_1-4l_1-7)S\end{gathered} \tag{25}$$

In some cases, when a binary of length l is recorded into the proposed {0,1,3}-NAF representation, either l is maintained (i.e., $l_1 = l$) or reduced by 1 bit (i.e., $l_1 = l-1$). Also, when a binary of length l is recorded into a {-1,0,1}-NAF representation, either l is maintained (i.e., $l_2 = l$) or increased by 1 bit (i.e., $l_2 = l+1$). Therefore, we need to consider analysis of the scalar multiplication cost for two cases:

Case 1: $l_1<l_2$ (where $l_1 = l-1$ and $l_2 = l+1$)
Case 2: $l_1= l_2 = l$

Case 1: $l_1<l_2$ where $l_1 = l-1$ and $l_2 = l+1$

From Eq. 25:

$$\begin{gathered}\Rightarrow(9h_1+5(l-1)-9h_2-5(l+1)+12)M<(5h_2+4(l+1)-5h_1-4(l-1)-7)S\\ \Rightarrow(9h_1-9h_2+2)M<(5h_2-5h_1+1)S\\ \Rightarrow(9(h_1-h_2)+2)M<-(5(h_1-h_2)+1)S\end{gathered}$$

Using M = 148.34 and S = 118.82 (Edoh, 2009):

$$\begin{gathered}\Rightarrow(9(h_1-h_2)+2)(148.34)<(-(5(h_1-h_2)+1)(118.82)\\ \Rightarrow 1335.06(h_1-h_2)+296.68<-594.1(h_1-h_2)+118.82\\ \Rightarrow (1335.06+594.1)(h_1-h_2)<118.82-296.68\\ \Rightarrow (1929.16)(h_1-h_2)<-177.86\\ \Rightarrow (h_2-h_1)>177.86/1929.6\\ \Rightarrow (h_2-h_1)>0.09\end{gathered} \tag{26}$$

In conclusion, for case $l_1< l_2$ (i.e., $l_1 = l-1$ and $l_2 = l+1$), from Eq. 26, the proposed scalar multiplication algorithm has better cost than the addition-subtraction algorithm when $(h_2-h_1)>0.09$:

Case 2: $l_1 = l_2 = l$

From Eq. 25:

$$\begin{gathered}\Rightarrow(9h_1+5l-9h_2-5l+12)M<(5h_2+4l-5h_1-4l-7)S\\ \Rightarrow(9h_1+5l-9h_2-5l+12)M<(5h_2+4l-5h_1-4l-7)S\\ \Rightarrow(9(h_1-h_2)+12)M<(-5(h_1-h_2)-7)S\end{gathered}$$

Using M = 148.34 and S = 118.82 (Edoh, 2009):

$$\begin{aligned}
&\Rightarrow (9(h_1\text{-}h_2)+12)(148.34)<(-5(h_1\text{-}h_2)-7)(118.82)\\
&\Rightarrow 1335.06(h_1\text{-}h_2)+1780.08<-594.1(h_1\text{-}h_2)-831.74\\
&\Rightarrow (1335.06+594.1)(h_1\text{-}h_2)<-831.74-1780.08\\
&\Rightarrow (h_1\text{-}h_2)<2611.82/1929.16\\
&\Rightarrow (h_2\text{-}h_1)>1.35
\end{aligned} \tag{27}$$

In conclusion, for case $l_1 = l_2 = l$, from Eq. 27, the proposed scalar multiplication algorithm has better cost than the addition-subtraction algorithm (Hankerson *et al.*, 2004) when $(h_2\text{-}h_1)>1.35$.

From Eq. 26 and 27, for average case, we can conclude that the proposed scalar multiplication algorithm has better cost than the addition-subtraction algorithm for the following cases:

Case 1: For $\ell_1 < \ell_2$, the Hamming weight of h_1 and h_2 must satisfy: $(h_2\text{-}h_1)>0$
Case 2: For $l_1 = l_2 = l$, the Hamming weight of h_1 and h_2 must satisfy: $(h_2\text{-}h_1)>1$

When the proposed tripling is compared with Eq. 12, it saved 14% multiplications and 22% squarings. Also, when compared with traditional tripling, the proposed tripling saved 33% multiplications and 12.5% squaring. For NIST curve, the value of a is equal to 1 and more reduction in scalar multiplication cost is achieved.

CONCLUSION

The digit 3 in the {0,1,3}-NAF scalar need tripling operations in the execution of the proposed scalar multiplication algorithm. At average case, the digit 3 also helps to minimize the Hamming weight of the scalar multiplier. From the result, at average case the proposed signed-digit {0,1,3}-NAF scalar multiplication algorithm is better than the addition-subtraction algorithms except when only one digit 3 in the proposed scalar. Finally, the proposed scalar multiplication algorithm is better than the addition-subtraction algorithms when the Hamming weight and the bit-length of the proposed {0,1,3}-NAF scalar is not equal to the Hamming weight and the bit-length of the {-1,0,1}-NAF scalar.

REFERENCES

Al-Daoud, E., R. Mahmod, M. Rushdan and A. Kilicman, 2002. A new addition formula for elliptic curves over GF(2^n). IEEE Trans. Comput., 51: 972-975.

Cohen, H. and G. Frey, 2005. Handbook of Elliptic and Hyperelliptic Curve Cryptography. CRC Press, USA.

Cohen, H. and G. Frey, 2006. Handbook of Elliptic and Hyperelliptic Curve Cryptography. Chapman and Hall/CRC, USA.

Dimitrov, V., L. Imbert and P.K. Mishra, 2005. Efficient and Secure Elliptic Curve Point Multiplication Using Double-Base Chains. In: Advances in Cryptology-ASIACRYPT, Roy, B. (Ed.). Springer, New York, pp: 59-78.

Edoh, 2009. Elliptic curve cryptography on pocket PC's. Int. J. Secur. Appl., 3: 23-33.

Hankerson, D., S. Vanstone and A. Menezes, 2004. Guide to Elliptic Curve Cryptography. Springer-Verlag, New York, ISBN-13: 9780387952734, Pages: 311.

Heuberger, C. and H. Prodinger, 2007. The Hamming weight of the non-adjacent-form under various input statistics. Periodica Mathematica Hungarica, 55: 81-96.

Higuchi, A. and N. Takagi, 2000. A fast addition algorithm for elliptic curve arithmetic in GF(2^n) using projective coordinates. Inform. Proc. Lett., 76: 101-103.

Joye, M. and S.M. Yen, 2000. Optimal left-to-right binary signed-digit recording. IEEE Trans. Comput., 49: 740-748.

Khabbazian, M., T.A. Gulliver and V.K. Bhargava, 2005. A new minimal average weight representation for left-to-right point multiplication methods. IEEE Trans. Comput., 54: 1454-1459.

Lange, T., 2004. A note on lopez-dahab coordinates. http://eprint.iacr.org/2004/323.pdf

Longa, P., 2007. Accelerating the scalar multiplication on elliptic curve cryptosystems over prime fields. Master's Thesis, University of Ottawa, Canada.

Lopez, J. and R. Dahab, 1999. Improved algorithms for elliptic curve arithmetic in GF(2^n). Proceedings of the 5th Annual International Workshop, August 17-18, 1998, Kingston, Ontario, Canada, pp: 201-212.

Morales-Sandoval, M. and C. Feregrino-Uribe, 2006. GF(2^m) arithmetic modules for elliptic curve cryptography. Proceedings of the IEEE International Conference on Reconfigurable Computing and FPGA's, September 20-22, 2006, San Luis Potosi, pp: 1-8.

Okeya, K., K. Schmidt-Samoa, C. Spahn and T. Takagi, 2004. Signed binary representations revisited. Proceedings of the 24th Annual International Cryptology Conference, August 15-19, 2004, California, USA., pp: 123-139.

Paryasto, M.W., Kuspriyanto, S. Sutikno and A. Sasongko, 2009. Issues in elliptic curve cryptography implementation. Internetworking Indonesia J., 1: 29-33.

Phillips, B. and N. Burgess, 2004. Minimal weight digit set conversions. IEEE Trans. Comput., 53: 666-677.

Qingwei, L., 2008. Efficient VLSI architectures for MIMO and cryptography systems. Ph.D. Thesis, Oregon State University, Corvallis.

Reitwiesner, G.W., 1960. Binary arithmetic. Adv. Comput., 1: 231-308.

Sullivan, N., 2007. Fast algorithms for arithmetic on elliptic curves over prime fields. Master's Thesis, University of Calgary, Calgary, Alberta.

Yasin, S.M., 2011. New Signed-Digir 0,1,3}-NAF scalar multiplication algorithm for elliptic curve binary field. Ph.D. Thesis, University Putra Malaysia, Malaysia.

Yasin, S.M., R.N.H. Nor, J. Din and M. Zaitun, 2014. {0, 1, 3}-NAF representation and algorithms for lightweight elliptic curve cryptosystem in lopez dahab model. Int. Rev. Comput. Software, 9: 1541-1547.