Improved Scalar Multiplication Algorithm in Affine Coordinate System using Elliptic Net

Zuren Razali[1, a], Norliana Muslim[2, b*], Saliyah Kahar[3, c] Faridah Yunos[4, d] and Kamsiah Mohamed[5, e]

[1]Faculty of Communication, Visual Art and Computing, Universiti Selangor Selangor, Malaysia

[2]Faculty of Engineering and Life Sciences, Universiti Selangor Selangor, Malaysia

[3]Faculty of Communication, Visual Art and Computing, Universiti Selangor Selangor, Malaysia

[4]Faculty of Science Universiti Putra Malaysia Selangor, Malaysia

[5]Faculty of Communication, Visual Art and Computing Universiti Selangor Selangor, Malaysia

[a]juerazali03@gmail.com, [b]norliana_muslim@unisel.edu.my, [c]saliyah_kahar@unisel.edu.my, [d]faridahy@upm.edu.my, [e]kamsh@unisel.edu.my

Article Info
Page Number: 1775-1788
Publication Issue:
Vol. 71 No. 3 (2022)

Abstract. In elliptic curve encryption, scalar multiplication (SM) is the most expensive and time-consuming operation. The elliptic curve cryptography attracts interest since it offers the same high security with a lower key length, owing to the advancement of modern technologies. Thus, this study designed a new scalar multiplication algorithm using six blocks of the elliptic net in a double and double-add method that cost 12M+6S in each block. This study also proposed a new formula for double block via the elliptic net method that saves four multiplications and four squaring from the prior double step. Experimental results over prime field p were conducted using safe curves namely numsp384t1 and numsp512t1, with equivalence sequences that satisfied gcd(p-1, 3). In the case of the 384-bits, results indicate that the developed scalar multiplication algorithm accelerates the running time by 65.96 % compared to the binary method, 44.81 % compared to the elliptic net without equivalent sequences, 30.28 % compared to the elliptic net with temporary variables, and 19.71 % compared to the seven blocks of the elliptic net with Karatsuba method. In a similar comparison for the 512-bits case, the proposed algorithm attained are 67.23 %, 44.65 %, 30.37 %, and 22.64 % faster, respectively.

Keywords: division polynomials, elliptic net, scalar multiplication, six block, Twisted Edwards curve.

Article History
Article Received: 12 January 2022
Revised: 25 February 2022
Accepted: 20 April 2022

Introduction

In the past, the elliptic curve cryptography (ECC) was introduced by [1]. Meanwhile, [2] also implement the ECC concept for a cryptosystem over binary field. In terms of security, ECC

provides equal security to Rivest-Shamir-Adleman (RSA) scheme but with a smaller key size, resulting in it to become faster and requiring less hardware than RSA. Because ECC utilises multiplication, it is also less computationally expensive than RSA, which depends on exponential computation. However, ECC security depends on parameters, besides the length [3]. To date, though ECC and RSA use similar mathematical problems, there is insufficient evidence to conclude that ECC is more secure than RSA because the security of RSA is based on the integer factorisation problem, while the security of ECC is based on the elliptic curve discrete logarithm problem [4].

The efficiency of ECC depends on the scalar multiplication (SM) algorithm [5]. SM is a major operation in ECC applications and protocols. It is defined by the process of finding nP where n is a positive integer and P is the point on the elliptic curve. It is a prominent element to implement a discrete logarithm-based system and a time-consuming operation in ECC [6, 7]. Recently, the advancement of the Internet of Things has centered on small-scale technological devices. These advancements lead to higher usage of memory and power consumption, in which smaller devices have limited capacity. Therefore, it is important to improve the computational efficiency of the SM algorithm [8].

The binary method, which is based on repeated addition and doubling of point P, was the traditional method to compute SM. The formula of addition and doubling in Affine coordinate involves an inverse operation, which results in the most expensive operation over prime field [9]. The author in [10] proposed a simplified double and double-add algorithm over prime field that executes the computation of SM via elliptic net values using eight terms in each iteration loop. Further improvements on double and double-add equations are done by [11] and [12] using the elliptic net with temporary variables and the elliptic net with the Karatsuba method, respectively. Nevertheless, the double and double-add algorithm in the literature for computing SM algorithm over prime field in Affine coordinate has high complexity due to the inversion process [13].

The Weierstrass curves have been used to compute the elliptic net SM algorithm. Meanwhile, the Twisted Edwards curves are used in the testing environment since the curves satisfy the properties of the elliptic net and contain a complete group law compared to the Weierstrass curves. Moreover, [14] indicate that the Twisted Edwards curves are the fastest for the elliptic curve Diffie-Hellman scheme over prime field. To date, there has not been a prior study that computes the elliptic curve SM algorithm via the elliptic net with fewer than eight intermediate variables. Therefore, the current research aims to design the SM algorithm over prime field using the optimal method for the Twisted Edwards curves in the running environment. Also, the primary purpose of this research is to reduce the complexity of the SM algorithm in ECC.

Literature Review

The division polynomial of the Twisted Edwards curves is analogous to the elliptic curves in the Weierstrass form. The polynomials characterise the n-torsion points of the Twisted Edwards curves for a positive integer n. The general form of the Twisted Edwards curves is

$ax^2 + y^2 = 1 + dx^2y^2$ where the coefficient a and d are integrals ($a \neq d$) and nonzero. The division polynomials of the Edwards curves that satisfy the properties of EDS are as follows:

$$W_0 = 0, W_1 = 1, W_2 = \frac{(a-d)(1+y)}{2x(1-y)} \tag{1}$$

$$W_3 = \frac{(a-d)^3\left(a + 2ay_1 - 2dy^3 - dy^4\right)}{\left(2(1-y)\right)^4} \tag{2}$$

$$W_4 = \frac{2(a-d)^6\, y(1+y)\left(a - dy^4\right)}{x\left(2(1-y)\right)^7} \tag{3}$$

For $n \geq 5$, W_n can be defined recursively using:

$$W_{2n+1} = W_{n+2}W_n^3 - W_{n-1}W_{n+1}^3 \tag{4}$$

$$W_2W_{2n} = W_n\left(W_{n+1}W_{n-1}^2 - W_{n-2}W_{n+1}^2\right) \tag{5}$$

The auxiliary polynomials for the Twisted Edwards curves are denoted by $\phi_n = \frac{(1+y)W_n^2}{(1-y)} - \frac{4W_{n-1}W_{n+1}}{(a-d)}$ and $\omega_n = \frac{2W_{2n}}{(a-d)W_n}$. The SM via the EN consists of the process of double and double-add in the algorithm. The researcher in [15] introduces the double and double-add algorithms to compute the sequences of the elliptic divisibility sequences (EDS). Given the initial values of an EDS, the technique computes the n-th term of the sequences in log (n) time. This method allows the sequences to start elsewhere with the given initial terms of the EDS. For clarity, W_n functions as the terms of the EN.

Then, [16] generates a block centred at the scalar consisting of eight consecutive EN terms centred at W_n. The first five terms of the EN can be calculated using Eqs. (1) to (3) and the next terms of the EN can be calculated using the recurrences formula as Eqs. (4) and (5). The double and double-add via EN algorithm was utilised by [16] for computing the cryptographic pairing. In 2014, [10] adapted the algorithm to compute the SM method using the short Weierstrass division polynomials with the cost of 26M+6S in each double and double-add step. Also, [11] reduced the cost to 16M+10S in each double and double-add step with additional temporary variables. Recent improvements on a similar scheme were done by [12] using seven blocks of the elliptic net with the Karatsuba method and the required lemma and the explicit formula upon Twisted Edwards curves is as below:

Lemma 1 [12]:

Let $\{\psi_n\}$ be the proper elliptic divisibility sequences over F_p with p elements $\psi_2 \neq 0$. Then there exists an elliptic net W_n over F_p which is equivalent 1778ot he sequence $\{\psi_n\}$ such that: $W_n = c^{n^2-1}\psi_n$.

Theorem 1 [12]**:** Let W_n defined from Lemma 1 and $W_2 = c^3\psi_2$. If $P = (x_1, y_1)$ on the Twisted Edwards curves over F_p, then $nP = (x_n, y_n)$ can be computed as follows:

$$(x_n, y_n) = \left(\frac{(a-d)\sigma_n W_n^2 c}{2W_{2n}}, \frac{\sigma_n - W_n^2 c^2}{\sigma_n + W_n^2 c^2} \right) \tag{6}$$

where,

$$\sigma_n = \frac{(1+y)W_n^2 c^2}{1-y} - \frac{4W_{n-1}W_{n+1}}{a-d} \tag{7}$$

Note that, to reduce the number of operations in double and double-add methods, Lemma 1 is utilised to set $W_2 = 1$ upon the Twisted Edwards curves. This step saves 4M in each loop. Algorithm 1 highlights the latest SM algorithm via EN proposed by [12] using Theorem 1.

Algorithm 1: SM algorithm via EN [12]

Input: $n = (n_l n_{l-1} \ldots n_0)_2$ with $n_l = 1$, $P \in E(F_p), a, d \in E, K = \widehat{W}(2)$, $L = \widehat{W}(3)$, $M = \widehat{W}(4)$, $\tilde{\alpha} = \widehat{W}(2)^{-1}$, $A = (1-y)^{-1}$, $B = (a-d)^{-1}$, c such that $c^3 = \tilde{\alpha}$, $R = c^{2(1-n^2)}$, $S = c^{-2n^2}$ and $T = c^{1-4n^2}$.

Output: (x_n, y_n).

1. $V \leftarrow [-K, -1, 0, 1, K, L, M]$

2. For i from $l-1$ down to 0 do

2.1 If $n_i = 0$ then

$$V_{2i} \leftarrow (S_i - S_{i+1})(P_i + P_{i+1}) - R_i + R_{i+1}$$

$$V_{2i+1} \leftarrow ((S_i - S_{i+2})(P_i + P_{i+2}) - R_i + R_{i+2})\tilde{\alpha}$$

$$V_6 \leftarrow (S_3 - S_4)(P_3 + P_4) - R_3 + R_4.$$

2.2 Else

$V_{2i} \leftarrow \left(\left(S_i - S_{i+2}\right)\left(P_i + P_{i+2}\right) - R_i + R_{i+2}\right)\tilde{\alpha}$

$V_{2i+1} \leftarrow \left(S_{i+1} - S_{i+2}\right)\left(P_{i+1} + P_{i+2}\right) - R_{i+1} + R_{i+2}$

$t_1 \leftarrow V_3 V_5;\ t_2 \leftarrow V_4^2;\ t_3 \leftarrow inverse\left(V_2\right)$

$V_6 \leftarrow \left(t_1\delta - t_2\beta\right)t_3$

3. $V \leftarrow \left[V_0, V_1, V_2, V_3, V_4, V_5, V_6, V_7\right]$

4. $S_0 \leftarrow V_2^2;\ S_1 \leftarrow V_3^2;\ S_2 \leftarrow V_4^2$

5. $J \leftarrow V_3\left(V_5 S_0 - V_1 S_2\right)$

6. $K \leftarrow \left[\left(V_2 + V_4\right)^2 - S_0 - S_2\right] / 2$

7. $L \leftarrow ABS_1 R - 4KC$

8. $M \leftarrow 2J$

9. $N \leftarrow L + S_1 R$

10. $O \leftarrow inverse\left(MN\right)$

11. $x_n \leftarrow \left(a - d\right) gLS_1 ON$

12. $y_n \leftarrow \left(L - S_1 R\right) OM$

13. Return $\left(x_n, y_n\right)$.

Based on Algorithm 1, Lines 2.1 and 2.2 denote the double and double-add steps executed for l −1 times for l bit length. This means double is executed at h −1 times, whereas double-add is performed for l−h times with the cost of 14M + 8S and 15M + 8S, respectively. Line 7 contains the formula for W_{2n}. After implementing the squaring trade-off as shown in Line 6, 1M is converted into 1S, proving that Line 7 needs 3M. Lines 4 to 9 refer to variables that are previously calculated using the last EN block generated for n_0 based on Eqs. (6) to (7), which demands 11M + 3S. Lines 11 and 12 contain the explicit formula upon the Twisted Edwards curves, therefore these lines require 7M. Let CP denotes the cost of Lines 4 to 12, where CP needs 14M + 4S. All in all, the overall cost of Algorithm 2 includes double, double-add, and CP costs. The expected cost of Algorithm 1 is calculated using the following proposition:

Proposition 1 [12]: For a sufficiently large scalar $n = \left(n_l, n_{l-1}, ..., n_0\right)_2$, the complexity of Algorithm 1 is denoted by

$$C_{[11]} = (l-h)\,double + (h-1)\,doubleAdd + CP \tag{7}$$

Methodology

This section proposes an elliptic net block using six terms by discarding two terms of each block. In contrast to [10], [11], [12], and [17], this study discarded the first term W_{k-3} and the eighth term W_{k+4} on the block centred at K, considering the block consisted of EN terms from W_{k-2} until W_{k+3}. This saves the cost of updating the block in each iteration. Thus, the block centred at 2K consists of EN term from W_{2k-2} until W_{2k+3} and the block centred at 2K+1 consists of EN term from W_{2k-1} until W_{2k+4}. However, the formula for W_{2k+4} requires $W_k, W_{k+1}, W_{k+2}, W_{k+3}$ and W_{k+4} while W_{2k-2} requires $W_{k-3}, W_{k-2}, W_{k-1}, W_k$ and W_{k+1} which are not available on the block centred at K using the six terms. In the current study, new formulas for W_{2k-2} and W_{2k+4} are obtained by substituting n = 2 and m = 2k into Eq. (5). That means,

$$W_{2k+4} = \left(W_{2k+3}W_{2k+1}W_2^2 - W_3W_1W_{2k+2}^2\right)/W_{2k} \tag{8}$$

$$W_{2k-2} = \left(W_{2k+1}W_{2k-1}W_2^2 - W_3W_1W_{2k}^2\right)/W_{2k+2} \tag{9}$$

From Eqs. (8) and (9), it is necessary to calculate W_{2k-1}, W_{2k} and W_{2k+1} to acquire W_{2k-2} terms, while W_{2k+1}, W_{2k+2} and W_{2k+3} for W_{2k+4} terms. Note that the inverse of W_{2k} and W_{2k+2} can be replaced by cheaper shifts divisions of 2 and subtraction [18] using the binary inversion algorithm.

Results and discussions

The current study proposes new double and double-add methods by utilising the elliptic net block created in the previous section. The authors in [10] used temporary variables S_i and P_i as an array of six elements that costed 6M + 6S in both methods. [11] and [10] utilised the same S_i and P_i by adding two groups of intermediate variables $A_i, B_i, C_i, D_i,$ and E_i for each double and double-add function. The formula was assigned to the algorithm, with each formula consisting of repeated multiplications. For instance, $W_{k-2}W_k$ was used in W_{2k-1} and W_{2k} formulas, $W_{k-1}W_{k+1}$ was used in W_{2k-1}, W_{2k+1} and W_{2k+2} formulas, whereas W_kW_{k+2} was used in W_{2k}, W_{2k+1} and W_{2k+3} formulas. The number of multiplications was reduced using the intermediate variables for the repeated multiplication W_{k-2} and W_k such that $a = W_{k-2}W_k$, followed by $b = W_{k-1}W_{k+1}$, and $c = W_kW_{k+2}$. The repeated operation of squaring was reduced using e, f, and g such that $e = W_{k-1}^2$, $f = W_k^2$, and $g = W_{k+1}^2$.

If the formula, in terms of intermediated variables, is rewritten with W_{2k-1}, W_{2k} and W_{2k+1} as R, S and T, then the new formula will be $R = af - be$, $S = (ag - ce)/W_2$ and $T = bg - cf$. This

equation costs 6M if the division by W_2 is neglected, since this changes to one using the properties of equivalence sequences [17]. The authors in [11] mention that the computation of $2R = 2(af - be)$, $2S = 2(ag - ce)$, and $2T = 2(bg - cf)$ can be calculated using A, B, C, D and E such that $A = (e + f)(a - b)$, $B = (e - f)(a + b)$, $C = (e + g)(a - c)$, $D = (e - g)(a + c)$, and $E = 2(f - g)(b + c)$. Then, the terms 2R, 2S, and 2T are calculated as $R = (A - B)/2$, $S = (C - D)/2$, and $T = \left[(C + D) - (A + B) - E\right]/2$, proving that R, S, and T can be computed using 5M instead of 6M. This method calculates W_{2k-1}, W_{2k} and W_{2k+1}. Let W_{2k+2} and W_{2k+3} be U and V and the intermediate variables d for the repeated multiplication W_{k+1} and W_{k+3} such that $d = W_{k+1}W_{k+3}$, and the repeated operation squaring is $h = W_{k+2}^2$. As a result, the formulas for U and V are $U = (bh - df)/W_2$ and $V = (ch - dg)$.

This formula computes U and V with 4M which can be reduced to 3M using intermediate variables F, G, and H, such that $F = (f + h)(b - d)$, $G = (f - h)(b + d)$, and $H = (g + h)(c - d)$. Hence, $U = (F - G)/2W_2$ and $V = H - ((A + B) - (C + D) + F + G)/2$. Note that from the intermediate variables assigned, $A + B = 2ae - 2bf$, $C + D = 2ae - 2cg$, $F + G = 2(bf - dh)$ and $F - G = 2(bh - df)$. Since $bh - df = U \cdot W_2$, substituting $bh - df$ with the F – G equation results in $F - G = 2(U \cdot W_2)$, and $U = (F - G)/2W_2$. Next, plugging $H = (g + h)(c - d)$, $(A + B) = 2(ae - bf)$, $(C + D) = 2(ae - cg)$, and $(F + G) = 2(bf - dh)$ into V yield the following:

$$\begin{aligned} V &= H - ((A + B) - (C + D) + F + G)/2 \\ &= (g + h)(c - d) - \left(2ae - 2bf - \left(2ae - 2cg\right) + \left(2bf - 2dh\right)\right)/2 \\ &= cg - dg + ch - dh - \left[(cg - bf) + (bf - dh)\right] \\ &= ch - dg. \end{aligned}$$

□

The new ENSM algorithm can be designed based on the generation of the elliptic net value in double and double-add processes with the initial block is the block centred at one. The scalar has a binary representation with the last iteration of the elliptic net block producing the elliptic net values for the block centred at scalar n. The formula for x_n and y_n in Theorem 1 is simplified by using temporary variables to optimise the inversion cost.

$$\begin{cases} Q^{-1} = (QR)^{-1} R \\ R^{-1} = (QR)^{-1} Q \end{cases} \tag{10}$$

Eq. (10) shows that the inverse of Q and R can be computed by inversing QR. This merge the two inversions into one inversion with two multiplications. Algorithm 2 depicts a new design of SM algorithm based on the proposed double and double-add with the multiple point formula upon the Twisted Edwards curves.

Algorithm 2 : New design of SM algorithm via EN in Affine coordinate

Input: Integer $n=(d_{l-1}d_{l-2}\ldots d_0)_2$, Point $P=(x,y),a$ and $d\in E, K=W_2, L=W_3, M=W_4$ and $\alpha=W_2^{-1}$, $A=1+y$, $B=(1-y)^{-1}$ $C=(a-d)^{-1}$, and g such that $g^3=\alpha$.

Output: $Q=(x_n,y_n)$

1. $V\leftarrow[-K,-1,0,1,K,L,M]$
2. For i from $l-2$ down to 0 do

2.1 If $d_i=0$ then

$V\leftarrow double(V)$

2.2 Else

$V\leftarrow double\text{-}add(V)$

3. Return V
4. $S_0\leftarrow V_1^2; S_1\leftarrow V_2^2; S_2\leftarrow V_3^2$
5. $J\leftarrow V_2\cdot(V_4\cdot S_0-V_0\cdot S_1)$
6. $K\leftarrow\left[(V_1+V_3)^2-S_0-S_2\right]/2$
7. $L\leftarrow AB\cdot S_1\cdot g^2-4K\cdot C$
8. $M\leftarrow 2\cdot J$
9. $N\leftarrow L+S_1g^2$
10. $O\leftarrow inverse(M\cdot N)$ //**Binary inversion algorithm
11. $x_n\leftarrow(a-d)\cdot L\cdot S_1\cdot g\cdot O\cdot N$
12. $y_n\leftarrow(L-S_1g^2)\cdot O\cdot M$
13. Return Q

From Algorithm 2, the inputs are point P, the curve parameters, and the elliptic net values (K, L and M) associated to point P, with α is the inverse of W_2. As a result, g is chosen such that $g^3=\alpha$. The values of A, B, and C are precomputed as they can be calculated using the curves parameters. Line 1 is the initial vector that changes after the application of the equivalence of EN. The cost of the proposed double or double-add is 12M + 6S (Lines 3 and 4). Thus, Lines 3 or 4 are executed consecutively for $l-1$ times. To be more specific, Line 3 is executed when $d_i=0$, while Line 4 is executed when $d_i=1$. More, Lines 6 until 14 are the formula of SM for the elliptic net. Line 7 is the formula for W_{2n} while Lines 8 and 9 are executed to obtain σ_n and $W_{n-1}W_{n+1}$, respectively. After implementing the squaring trade-off as shown in Line 8, this converts 1M into 1S. Thus, Line 9 needs 3M. Line 12 in Algorithm 2 changes the cost to 1I using [19] method. Taking into consideration the cost of double and double-add, Algorithm 2 does not depend on the Hamming weight. The inversion in Line 12 is computed using the binary inversion algorithm [20] by replacing the inversion with cheaper shifts divisions by 2

and subtraction [18]. CP illustrates the cost of multiple points from Lines 6 to 14. The point operational cost of Algorithm 2 is stated in Proposition 2.

Proposition 2: For a sufficiently large scalar $n=(d_{l-1}d_{l-2}\ldots d_0)_2$, the complexity of Algorithm 2 is denoted by

$$C_{\text{Proposed}}=(l-1)\,double+CP \tag{11}$$

Proof. From Algorithm 2, it is clear that Lines 3 and 4 are executed consecutively for every digit d_i in $l-2\le i\le 1$. The double process is only executed when $d_i=0$, whereas double-add when $d_i=1$. Therefore, the cost of Algorithm 2 is the total of Line 2 and CP, which is denoted by the cost from Lines 5 to 13. Then, the cost is $C_{\text{Proposed}}=(l-1)\,double+CP$. □

Table 1 summarises the existence of double and double-add in literature, including the double and double-add proposed in the current study. The similar costs of double and double-add in the proposed method signify the non-reliance on the Hamming weight of the scalar.

Table 1. Cost of double and double-add

Method	Temporary variables	double	double-add	Total cost	
				double	double-add
[9]	-	2M+2S+2I	9M+2S+4I	2M+2S+2I	9M+2S+4I
[10]	6M+6S	20M	20M	26M+6S	26M+6S
[11]	12M+10S	4M	4M	16M+10S	16M+10S
[12]	7M+6S	7M	8M	14M+8S	15M+8S
Proposed	10M+6S	2M	2M	12M+6S	12M+6S

Performance analysis

In this section, the results of the analysis of complexity which was carried out to compare the performance of the proposed SM algorithm with prior methods in literature are discussed. For the implementation, the analyses in [9] and [10] were conducted on 384-bits and 512-bits using the safe curves of numsp384d1 and numsp512d1, while [12] and the proposed method used the safe curves of numsp384t1 and numsp512t1. The parameters were double and double-add points. The cost for binary method algorithm [9] was based on $C_{BM}=(l-h)\,double+(h-1)\,double\text{-}add$, while [10] and [11] utilised $C_{\text{KR}}=(l-1)\,double+CW$, with CW representing the explicit formula cost upon the

Weierstrass curves. Even though, [12] used Eq. (7), this research considered Eq. (11). To evaluate the cost of point operation for the proposed algorithm, the cost was compared to other proposed functions in the literature. On average, the binary representation with bit length l =384, the Hamming weight is h =192, and l = 512, h = 256. Table 2 summarises the cost of point operations using various SM methods.

Table 2. Point operation cost

Method	384-bits	512-bits
[9]	191 double-add + 192 double	255 double-add + 256 double
[10]	383 double + CW	511 double + CW
[11]	383 double + CW	511 double + CW
[12]	191 double-add + 192 double + CP	255 double-add + 256 double + CP
Proposed	383 double + CP	511 double + CP

According to Table 2, methods by [9] and [12] consisted of double and double-add operations due to the dissimilar cost for each operation. The point operation costs in [10] and [11] were similar since the authors utilised the identical multiple point formula upon the Weierstrass curves over prime field with CW denoting the cost, M denoting the multiplication, S denoting the squaring, and I denoting the denote the inversion. The field operation cost for each method was then evaluated by substituting double and double-add from Table 1. With C_l denoting the field operation cost at bit length l, the field operation costs for 384-and 512-bits for the proposed method were computed by substituting $1\,double = 12M + 6S$ 1 and CP = 14M + 4S into the point operation cost obtained in Table 2. That means,

$$\begin{aligned} C_{384} &= 383\,double + CP \\ &= 383(12M + 6S) + (14M + 4S) \\ &= 4610M + 2302S, \end{aligned}$$

$$\begin{aligned} C_{512} &= 511\,double + CP \\ &= 511(12M + 6S) + (14M + 4S) \\ &= 6146M + 3070S. \end{aligned}$$

The computed C_l values are benchmarked with other methods. Table 3 summarises the field operation costs for the average case in 384-bits and 512-bits.

Table 3. Field operation cost

Method	384-bits			512-bits		
	M	S	I	M	S	I

[9]	2103	766	1148	2807	1022	1532
[10]	9966	2301	1	1329	3069	1
[11]	6136	3833	1	8184	5113	1
[12]	5567	3968	0	7423	4092	0
Proposed	4610	2302	0	6146	3070	0

Based on Table 3, for 384-bits, the costs of M, S, and I for [9], [10], [11], and [12] are 2103M + 766S + 1148I, 9966M + 2301S + 1I, 6136M + 3833S + 1I and 5567M + 3968S, respectively. The costs for these methods rise to 2807M + 1022S + 1532I, 1329M + 3069S + 1I, 8184M + 5113S + 1I and 7423M + 4092S for 512-bits. These were by considering a scalar n with l = 384 and h = 192. Note that the point operational cost in the proposed method was 383 double + CP, one double cost was 12M + 6S, and CP from Algorithm 2 was 14M + 4S. The field operation cost was calculated using double cost (see Table 1) and the total number of multiplications in 384-bits was computed by substituting $1S = 0.87M$ such that:

$$\begin{aligned} TM_{384} &= 383(12M + 6S) + 14M + 4S \\ &= 4610M + 2302S \\ &= 4610M + 2302(0.87) \\ &= 6612.74M. \end{aligned}$$

A similar computation was conducted on 512-bits using $1S = 0.89M$ such that:

$$\begin{aligned} TM_{512} &= 512(12M + 6S) + 14M + 4S \\ &= 6146M + 3070S \\ &= 4610M + 3070(0.89) \\ &= 8878.30M. \end{aligned}$$

The computed total multiplications were benchmarked with other SM methods over prime field using the similar conversion unit proposed by [14]. Fig. 1 depicts the estimated number of multiplications based on double and double-add in various SM methods.

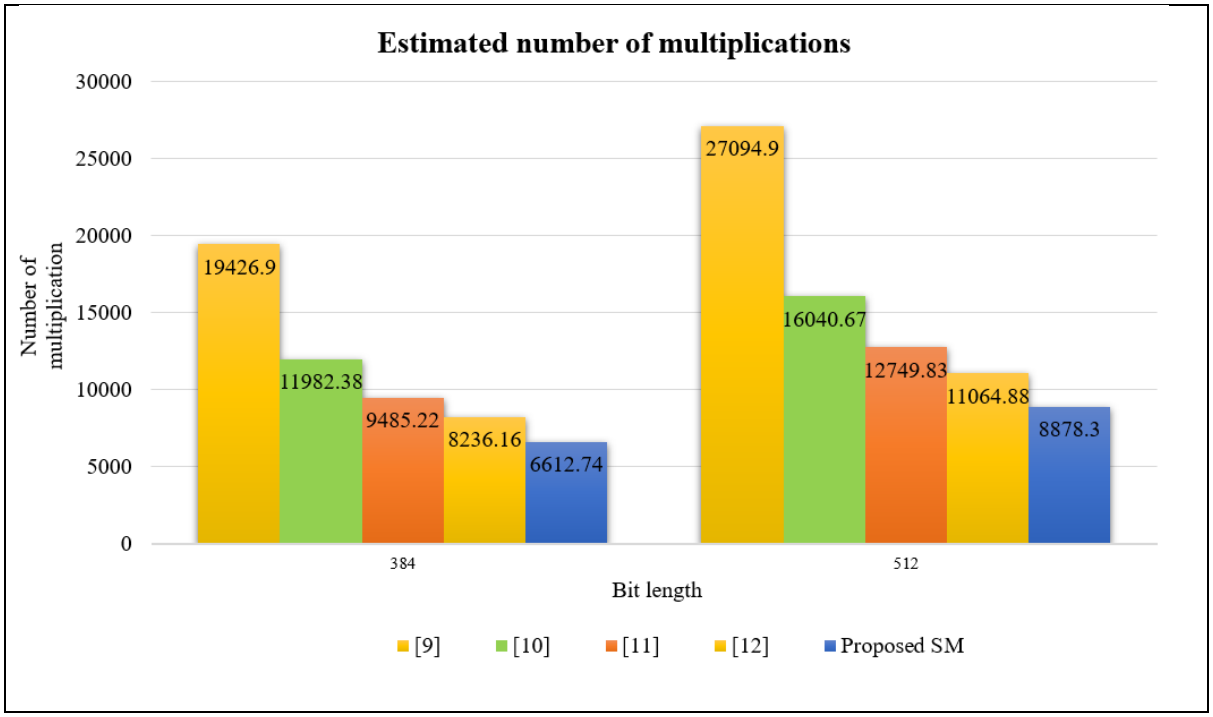

Fig. 1. Estimated number of multiplications

Fig. 1 shows that the proposed method is optimised when compared to [9], [10], [11], and [12]. The cost of methods increases with each increment in bit size. For instance, methods by [9], [10], [11], and [12] required 19426.9M, 11982.38M, 9485.22M. and 8236.16M, respectively, while the proposed method requires 6612.74M for 384-bits. As for the case of 512-bits, the proposed method requires 8878.30M versus 27094.9M for [9], 16040.67M for [10], 12749.83M for [11], and 11064.88M for [12]. For computing the timing in 384-bits, the squaring and inversion were converted to M by multiplying with 0.87 and 14.51, respectively [14]. For 512-bits, the multiplication requires 0.89 and 15.26, respectively. The running time denoted by RT_l at 384-bits can be computed by substituting $1M = 5.8\times10^{-7}s$, such that $RT_{384} = 6612.74M\left(5.8\times10^{-7}\right)s = 0.00384s$. A similar computation is computed for 512-bits. The recorded running time for various SM methods over prime field, in seconds, is tabulated in Table 4.

Table 4. Estimated running time (s)

Method	384-bits	512-bits
[9]	0.01127	0.02303
[10]	0.00695	0.01363
[11]	0.00550	0.01084
[12]	0.00478	0.00941
Proposed	0.00384	0.00755

From Table 4, at 384-bits, the proposed method accelerates the running times by 65.96 % compared to [9], 44.81 % compared to [10], 30.28 % compared to [11], and 19.71 % compared to [12]. On a similar comparison, the percentages of acceleration for 512-bits are computed at 67.23 %, 44.65 %, 30.37 %, and 22.64 %, respectively.

Summary

In this research, a new SM algorithm has been designed using the modified double and double-add formula. The new SM algorithm over prime field is designed in the Affine coordinate without inversion based on the safe curves of numsp384t1 and numsp512t1. For 384-bits, results indicate that the proposed design of SM is 65.96 % better in performance compared to [9], 44.81 % better than [10], 30.28 % better than [11], and 19.71 % better than [12]. On a similar comparison for 512-bits, the performance of the proposed SM algorithm attains 67.23 %, 44.65 %, 30.37 %, and 22.64 % better in performance, respectively.

Funding

This research was funded by the Ministry of Education (Grant code: FRGS/1/2019/ICT05/UNISEL/03/1). The research is a part of research project entitled ‘A New Method in Elliptic Net Scalar Multiplication using Twisted Edward’s Division Polynomials’.

References

1. V.S. Miller, Use of elliptic curves in cryptography, in H.C. Williams (Ed.), Advances in Cryptology – CRYPTO '85, Lecture Notes in Computer Science, 218, Springer-Verlag, Berlin, 1986, pp. 417–426. DOI: https://doi.org/10.1007/3-540-39799-X_31
2. N. Koblitz, Constructing elliptic curve cryptosystems in characteristic 2, in A.J. Menezes, S.A. Vanstone (Eds.), Advances in Cryptology – CRYPT0 '90, Lecture Notes in Computer Science, 537, Springer, Berlin, 1991, pp. 156–167. DOI: https://doi.org/10.1007/3-540-38424-3_11
3. A.J. Zargar, M. Manzoor, T. Mukhtar, Encryption/decryption using elliptical curve cryptography, Int. J. Adv. Res. Comput. Sci. 8 (2017) 48–51. DOI: https://doi.org/10.26483/ijarcs.v8i7.4208
4. D. Mahto, D.K. Yadav, RSA and ECC: A comparative analysis, Int. J. Appl. Eng. Res. 12 (2017) 9053–9061.
5. R.K. Kadu, D.S. Adane, A novel efficient hardware implementation of elliptic curve cryptography scalar multiplication using vedic multiplier, Int. J. Simul. Syst. Sci. Technol. 19 (2018) 38.1–38.10. DOI: https://doi.org/10.5013/IJSSST.A.19.06.38
6. F. Alkhudhayr, T. Moulahi, A. Alabdulatif, Evaluation study of elliptic curve cryptography scalar multiplication on raspberry Pi4, Int. J. Adv. Comput. Sci. Appl. 12 (2021) 472–479. DOI: 10.14569/IJACSA.2021.0120954.
7. M. Bafandehkar, S.M. Yasin, R. Mahmod, Optimizing (0, 1, 3)-NAF recoding algorithm using block-method technique in elliptic curve cryptosystem, J. Comput. Sci. 12 (2016)

534–544. DOI: 10.3844/jcssp.2016.534.544.
8. Ł. Chmielewski, P.M.C. Massolino, J. Vliegen, L. Batina, N. Mentens, Completing the complete ECC formulae with countermeasures, J. Low Power Electron. Appl. 7 (2017) 1–13. DOI: 10.3390/jlpea7010003.
9. H. Hisil, K.K. Wong, G. Carter, E. Dawson, Twisted Edwards curves revisited, in Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT '08), Springer-Verlag, Berlin, 2008, pp. 326–343. https://doi.org/10.1007/978-3-540-89255-7._
10. N. Kanayama, Y. Liu, E. Okamoto, K. Saito, T. Teruya, S. Uchiyama, Implementation of an elliptic curve scalar multiplication method using division polynomials, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E97-A, 2014, pp. 300–302. DOI: http://dx.doi.org/10.1587/transfun.E97.A.300
11. S.R.S. Rao, Z. Hu, C.A. Zhao, Division polynomial-based elliptic curve scalar multiplication revisited, IET Inf. Secur. 13 (2019) 614–617. DOI: 10.1049/iet-ifs.2018.5361.
12. N. Muslim, F. Yunos, Z. Razali, Enhanced scalar multiplication algorithm over prime field using elliptic net, Manuscript submitted for publication, 2022.
13. S. Agievich, S. Poruchnik, V. Semenov, Small scalar multiplication on Weierstrass curves using division polynomials, Mat. Vopr. Kriptogr. 13 (2022) 17–35. DOI: https://doi.org/10.4213/mvk406
14. O.W. Eide, Elliptic curve cryptography – Implementation and performance testing of curve representations, Master's thesis, University of Oslo, 2017.
15. R. Shipsey, Elliptic divisibility sequences, Doctoral dissertation, University of London, 2000.
16. K. Stange, Integral points on elliptic curves and explicit valuations of division polynomials, Can. J. Math. 68 (2016) 1120–1158. DOI: 10.4153/CJM-2015-005-0.
17. B. Chen, C. Hu, C.A. Zhao, Note on scalar multiplication using division polynomials, IET Inf. Secur. 11 (2017) 195–198. DOI: 10.1049/iet-ifs.2015.0119.
18. D. Hankerson, A.J. Menezes, S. Vanstone, Guide to elliptic curve cryptography. Springer-Verlag, New York, 2004.
19. S. Liu, X. Heng, Y. Li, Anti-SPA scalar multiplication algorithm on Twisted Edwards elliptic curve, Int. J. Netw. Secur. 22 (2020) 1015–1021. DOI: 10.6633/IJNS.202011.
20. T. Kudithi, R. Sakthivel, An efficient hardware implementation of finite field inversion for elliptic curve cryptography, Int. J. Innov. Technol. Explor. Eng. 8 (2019) 827–832. DOI: 10.35940/ijitee.i7605.078919.