Fast Scalar Multiplication on Elliptic Curve Cryptography in Selected Intervals Suitable For Wireless Sensor Networks

Youssou Faye[1], Herve Guyennet[1], Ibrahima Niang[2], and Yanbo Shou[1]

[1]Femto-st DISC, Franche-Comte University, France,
yfaye@femto-st.fr,hguyenne@femto-st.fr,yshou@femto-st.fr
[2]Department of Mathematics and Computer Sciences, UCAD University, Senegal
iniang@ucad.sn

Abstract. In Wireless Sensor Networks (WSNs), providing a robust security mechanism with limited energy resources is very challenging because of sensor node's limited resources (computation, bandwidth, memory). Asymmetric-key can fulfill the requirement, but if the number of nodes is large, symmetric-key cryptography is the best natural method because of its scalability. Asymmetric-key cryptography is power-hungry; nevertheless, Elliptic Curve Cryptosystems (ECC) are feasible and more flexible for sensor nodes. Scalar multiplication is the most widely used operation on ECC. Various methods for fast scalar multiplication exist, but they are based on the binary/ternary representation of the scalar. In this paper, we present a novel technique to make fast scalar multiplication on Elliptic Curve Cryptosystems over prime field for light-weight embedded devices like sensor nodes. Our method significantly reduces the computation of scalar multiplication by an equivalent representation of points based on point order in a given interval. Since our technique can act as a support for most existing methods, after an analytical and efficiency analysis, we implement and evaluate its performance in different scenari.

Keywords: Elliptic Curve Cryptography, Fast Scalar Multiplication, Wireless Sensor Networks

1 Introduction

Security in Wireless Sensor Networks has attracted more and more attention in recent years. Symmetric cryptography is the most suitable application in constrained platforms such as sensor devices. For a large number of devices, the natural method employed is the asymmetric key cryptography algorithm because of its scalability. Compared to other asymmetric cryptosystems like RSA, Elliptic Curve Cryptography is an emerging favorite due to its shorter key length requirements for the same level of security strength [1]. The mathematical hierarchy of elliptic curve involves three arithmetic levels: scalar arithmetic, point

arithmetic and field arithmetic [2]. Point operations involve points addition and doubling, tripling or quadrupling (or similar operations). Scalar multiplication denoted by kP where P represents a point on the ellipic curve and k represents a scalar. The scalar multiplication is the central and most time-consuming operation in ECC because it is used for key generation, encryption/decryption of data and signing/verification of digital signatures. To perform fast computation of scalar multiplication, which is the major computation involved in ECC, much research has been devoted to the point arithmetic level and the scalar arithmetic [3,4,5,6,7,8]. On the scalar arithmetic level, the double-and-add technique is the traditional binary algorithm which is based on points operations, namely doubling of a point and addition of points. Also, well-known algorithms, such as Non-Adjacent Form (NAF), window NAF, and sliding window [3], can effectively reduce the number of point operations. Again, some other algorithms, such as double-base chains, have been developed to compute faster scalar multiplication by using binary and ternary representation[4],[5],[6]. Thus, algorithms, based on the aforementioned algorithms, optimize faster scalar multiplication[7],[8].
In addition, on point arithmetic, some schemes use algebraic substitutions of the multiplication operations with squaring operations and other cheaper field operations such as addition, subtraction and multiplication or division by a small constant [9].
Recently, a new concept of using multiprocessor architectures to process several operations of scalar multiplication simultaneously has been developed. At the point arithmetic level, some algorithms parallelize ECC formulas to reduce the time complexity of scalar multiplication [9]. At the scalar arithmetic level, the algorithm in [10] parallelizes the series of doubling and addition operations of a point on the binary algorithm with two processor architectures. For other solutions, parallelization is done by partitioning the scalar into n equal-length bit substrings on multiprocessor architectures [11],[12]. In very recent research, this partitioning technique is used on the sensor nodes in [13].
In this paper, we propose a method to accelerate scalar multiplication. For a given scalar multiplication kP, we replace it with an equivalent representation dP where the scalar $d < k$. Our technique is based on point order and the negative of point. Current research shows this is the first method based on this technique. All the above mentioned algorithms can basically use our technique to perform faster computation on scalar multiplication.
The rest of this paper is organized as follows: Section 2 describes some preliminaries about ECC over prime fields; in Section 3, we present our new scalar reduction. After outlining the context of our first contribution (Section 3.1), we describe the new scalar reduction (Section 3.2), and make respectively, an analytical evaluation and efficiency analysis (Section 3.3) and (Section 3.4). In order to verify our claims we implemented a simulator in Java and analyze its performance (Section 3.5). Finally, Section 4 is related to our conclusions and perspectives.

2 Preliminaries on Elliptic Curves over prime fields

In this section, a brief background description about ECC over finite prime fields is given. An elliptic curve E over finite field $\mathbb{F}$ (of order n) denoted by $E(\mathbb{F})$ can be defined by the long Weierstrass equation [2]:

$$E : y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6 \ . \tag{1}$$

where a_1, a_2, a_3, a_4 *and* a_6 are elements in $\mathbb{F}$.
The field generally used in cryptography relates to the prime field denoted by $\mathbb{F}_p$ where p= q^m and q a prime number called the characteristic of $\mathbb{F}_p$. If $q^m = p$, $\mathbb{F}_p$ is called a prime field. In this paper, we work with a prime field $\mathbb{F}_p$, where $p > 3$. For prime fields, if the characteristic is more than 3, the Weierstrass equation can transform to:

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

where a and b $\in \mathbb{F}_p$.
To be used for cryptography, the necessary condition is the discriminant of polynomial:

$$f(x) = x^3 + ax + b, \triangle = 4a^3 + 27b^2 \neq 0. \tag{3}$$

The points (x, y), where integer coordinates x, y satisfy the above equation and the point at infinity denoted by ∞ is also a point on the curve and form an abelian group. The group law mainly consists of two basic operations: point doubling ($2P$) and point addition $(P+Q)$ where P and Q are two different points on the curve.
Given $P=(x_p,y_p)$ and $Q=(x_q,y_q)$ two points ($\neq \infty$) which are on the elliptic curve over $\mathbb{F}_p$ denoted by $E(\mathbb{F}_p)$. The points addition $P+Q=(x_{pq},y_{pq})$ or point doubling $2P=P+Q=(x_{pq},y_{pq})$ if $P=Q$ can be calculated as:

$$\begin{cases} x_{pq} = \lambda^2 - x_p - x_q \\ y_{pq} = \lambda(x_p - x_{p+q}) - y_p \end{cases} \tag{4}$$

$$\begin{cases} \lambda = \dfrac{y_q - y_p}{x_q - x_p}, if P \neq Q \\ \lambda = \dfrac{3x_p^2 + a}{2y_p}, if P = Q \end{cases} \tag{5}$$

The negative of a point only involves the change of at most one of its coordinate values in the point representation. For example, the negative of a point $P = (x_p,y_p)$ is its reflection in the x-axis: the point $-P$ is $(x_p,-y_p)$. Notice that for each point P on an elliptic curve, the point $-P$ is also on the curve.

3 New scalar reduction method

3.1 Context

In this section, we present a new improvement in the scalar arithmetic level. This improvement is based on specific reduction of the scalar in a selected interval.

Assume that $\mathbb{F}_p$ has a characteristic greater than 3. Let $E(\mathbb{F}_p)$ be an elliptic curve over a prime field. Let $\#E(\mathbb{F}_p)$ denotes the number of points of the elliptic curve over $E(\mathbb{F}_p)$. $\#E(\mathbb{F}_p)$ is also called the order of group of points. A well-known theorem of Hasse states that [2]: $|\#E(\mathbb{F}_p)$ - p - 1 $\leq 2\sqrt{p}$ |. Let $\mathbb{G}$ be a cyclic group of $E(\mathbb{F}$p) of order n generated by a base point P (namely generator point). The points in $\mathbb{G}$ are expressed as multiples of P: $\mathbb{G}=\langle P\rangle=\{\infty, P, 2P, ...,$ (n-2)P, (n-1)$P\}\subseteq E(\mathbb{F}_p)$ with $nP=\infty$. The order of point P (denoted by $\#P$) is n.

3.2 Description of our new scalar reduction method

In our approach, we replace the point, namely kP, in the main scalar multiplication operation by an equivalent representation point dP (k and d two scalars and $k > d$) in the interval $[\lfloor n/2\rfloor+1, n\text{-}1]$, where $\lfloor n/2\rfloor$ denotes the integer-part function of $n/2$. As the negative of a point is obtained freely, we use it to make fast computation. Given the point $P=(x_p, y_p)$ in affine coordinates, to compute the negative (inverse) of the point $kP=(x_{kp}, y_{kp})$, we can compute $kP=(x_{kp}, y_{kp})$ and then change the sign on the y-coordinate (y_{kp}). Notice that for each point P on an elliptic curve, the point $-P$ is also on the curve. Thus, we can replace the point kP by an equivalent point representation dP utilizing the negative of point. For a secret scalar k (integer number), by the point kP, we get an equivalent points representation dP by following the equations, in a general case:

$$\begin{cases} a. & \text{If } k > \text{n }, kP = dP \quad where\ d = (k - \lfloor k/2\rfloor.n); \\ b. & \text{If k} \in]\lfloor \text{n}/2\rfloor, \text{n-1}], kP = dP \quad where\ d = (k - n); \\ c. & \text{If k} \in]0, \lfloor \text{n}/2\rfloor], kP = dP \quad where\ d = k; \\ d. & \text{If } k{=}n \text{ or 0 or } -n, kP = \infty; \\ e. & \text{If k} \in [\text{-(n-1)}, \text{-}\lfloor \text{n}/2\rfloor[, kP = dP \quad where\ d = (n + k); \\ f. & \text{If k} \in [\text{-}\lfloor \text{n}/2\rfloor, 0[, kP = dP \quad where\ d = k; \\ g. & \text{If k <- n, } k\text{P}{=}dP \quad \text{where } d{=}k{+}n.\lfloor|\text{k}|/2\rfloor. \end{cases} \tag{6}$$

In Elliptic Curve Cryptography k $\in]0, (\text{n-1})]$, we get an equivalent representation for point dP by equations (6.b) and (6.c) following:

$$\begin{cases} (b) & \text{If k} \in]\lfloor \text{n}/2\rfloor, \text{n-1}], kP = dP \quad where\ d = (k - n); \\ (c) & \text{If k} \in]0, \lfloor \text{n}/2\rfloor], kP = dP \quad where\ d = k. \end{cases}$$

We use an example of the above description in order to better express our reduction method.

Example: We choose the prime number p =23. Note that this process mainly reflects our new protocol. In real cases, the p is much bigger than this. If we consider the elliptic E over $\mathbb{F}_{23}$ defined by $E(\mathbb{F}_{23})$: $y^2{=}x^3{+}$x$+1$, then # $E(\mathbb{F}_{23}){=}28$, $E(\mathbb{F}_{23})$ is a cyclic group. Let $P(0,1)$ be a generator point. The points in $E(\mathbb{F}_{23})$ are shown below:

P=(0, 1)	$2P$=(6, -4)	$3P$=(3, -10)	$4P$=(-10, -7)
$5P$=(-5, 3)	$6P$=(7, 11)	$7P$=(11, -3)	$8P$=(5, -4)
$9P$=(-4, -5)	$10P$=(12, 4)	$11P$=(1, -7)	$12P$=(-6, 3)
$13P$=(9, -7)	$14P$=(4, 0)	$15P$=(9, 7)	$16P$=(-6, 3)
$17P$=(1, 7)	$18P$=(12, -4)	$19P$=(-4, 5)	$20P$=(5, 4)
$21P$=(11, -3)	$22P$=(7, -11)	$23P$=(-5, -3)	$24P$=(-10, -7)
$25P$=(3, 10)	$26P$=(6, 4)	$27P$=(0, -1)	$28P$=∞

For this example, the general and elliptic curve cryptography cases can be shown respectively in Figure 1 and Figure 2. On one hand, we can see the

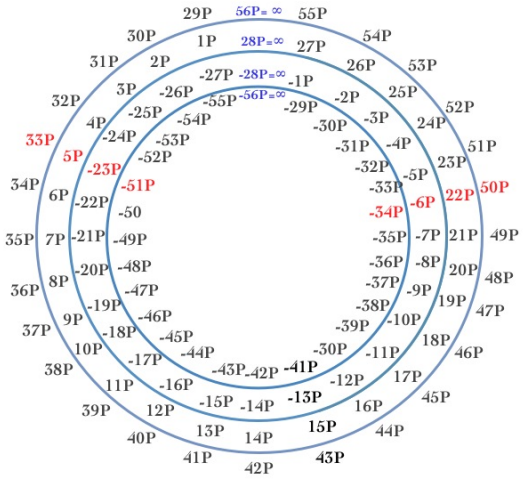

Fig. 1. General elliptic curve case

general case from Figure 1 that the points -34P, -6P, 22P, 50P have the same coordinates. To process the point -6P (the negative of 6P), we compute the point 6P and affix the minus sign to the y-coordinate. Thus, computation of -6P is free and almost equal to computing 6P . The scalar multiplication using our method in the the general case would be computed as follows:

to calculate 50P, we compute -6P by applying formula; (6.a)
to calculate 22P, we compute -6P by applying formula; (6.b)
to calculate -34P, we compute -6P by applying formula. (6.g)

On the other hand, for the special case of ECC, we can see from Figure 2 that computing the points from [15P,16P,.....26P, 27P] can be replaced respectively by [-13P, -12P,.....,-2P, -P]. In this case, computing 27P can be replaced by -P and is almost free.

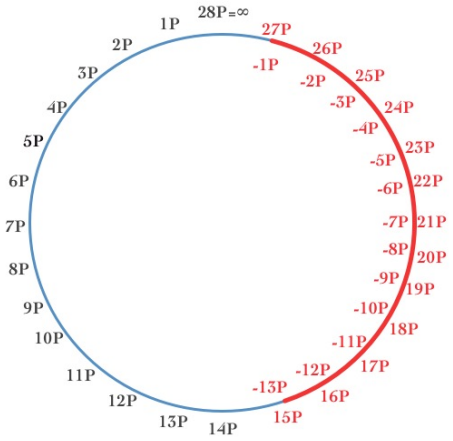

Fig. 2. Elliptic Curve Cryptography case

3.3 Analytical evaluation

Since sensor nodes are in low-power energy, replacing computation kP by computation dP using formula (6.b) in [⌊n/2⌋+1, n-1] can help us deduce faster computation on scalar multiplication. Meanwhile, the scalar k can be chosen only in this interval for computation in WSNs. From formula (7), we can scan all scalars in a given interval:

$$\sum_{k=1}^{n-1} kP = \sum_{k=1}^{\lfloor n/2 \rfloor -1} kP + \lfloor \frac{n}{2} \rfloor P + \sum_{k=\lfloor n/2 \rfloor +1}^{n-1} kP \tag{7}$$

where $\sum_{k=\lfloor n/2 \rfloor +1}^{n-1} kP = \sum_{k=1}^{\lfloor n/2 \rfloor -1} kP + 2 \sum_{k=1}^{\lfloor n/2 \rfloor -1} kP$

By using our method, if we replace respectively [15P,16P,.....26P, 27P] by [-13P, -12P,.....,-2P,-1P] in the interval [⌊$n/2$⌋+1, n-1] the expression : $\sum_{k=\lfloor n/2 \rfloor +1}^{n-1} kP$ can be replaced by $\sum_{k=1}^{\lfloor n/2 \rfloor -1} |k|P$, see formula (9).

$$\sum_{k=1}^{n-1} kP = 2 \sum_{k=1}^{\lfloor n/2 \rfloor -1} kP + \lfloor \frac{n}{2} \rfloor P + 2 \sum_{k=1}^{\lfloor n/2 \rfloor -1} kP \tag{8}$$

By using our method, the equation (8) can be replaced by equation (9):

$$\sum_{k=1}^{\lfloor n/2 \rfloor -1} kP + \sum_{k=1}^{\lfloor n/2 \rfloor -1} |k|P + \lfloor \frac{n}{2} \rfloor P. \tag{9}$$

By scanning all scalars k for computation of kP in the interval $[\lfloor n/2 \rfloor+1$, n-1],we can see from formulas (8)-(9), that we gain a rate of $\sum_{k=1}^{\lfloor n/2 \rfloor-1} 2kP$. The speed-up for a given scalar kP is *2(k-(n/2))*. For example, in Figure 2, computing kP=22P, is equal to computing 6P, the speed-up is *2(22-(14/2))= 16P* since *16P+ 6P= 22P*. In fact, the complexity of scalar multiplication is determined by the length bit of k which is equal to$\lfloor log_2(k) \rfloor+1$ or $log_2(k)$ if k=2^x, where x is an integer. In binary representation, $log_2(k)$ can be replaced in our method by:

$$log_2(k - 2(k - \frac{n}{2})) = log_2(k) + log_2(k + \frac{n - 2k}{k}) \tag{10}$$

Thus, the speed-up in length bit is:

$$|log_2(k + \frac{n - 2k}{k})| = |log_2(\frac{|d|}{k})|, \text{ where } log_2(k + \frac{n - 2k}{k}) < 0.$$

Since our reduction method is not possible for n=1, from Figure 3, we can see in interval $[\lfloor n/2 \rfloor+1$, n-1] the sum of all scalars. For a scalar in [2, $\lfloor n/2 \rfloor$] ,

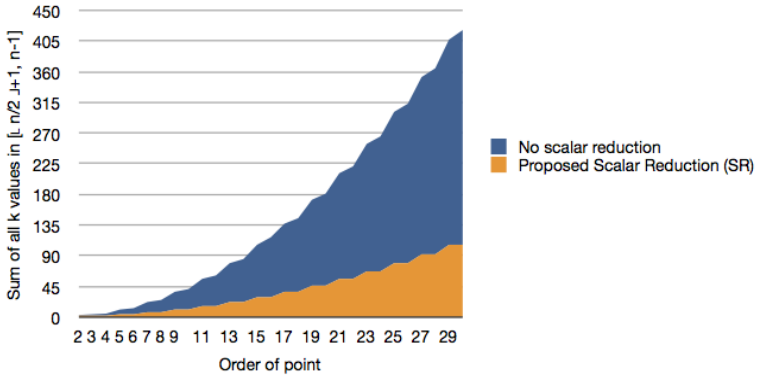

Fig. 3. Sum of all k values function of order n in $[\lfloor n/2 \rfloor+1$, n-1]

our method is not used since its design is only for the interval $[\lfloor n/2 \rfloor+1$, n-1]. Consequently, it is very efficient in Figure 3,in which we only consider scalars in this interval. From formula (11), we make the average computation of all scalar k in interval$[\lfloor n/2 \rfloor + 1, n - 1]$, where our technique can be applied

$$\frac{1}{n - 1 - \lfloor n/2 \rfloor}(\sum_{k=\lfloor n/2 \rfloor+1}^{n-1} kP) = \frac{1}{n - \lfloor n/2 \rfloor}(\sum_{k=1}^{\lfloor n/2 \rfloor-1} kP). \tag{11}$$

The average of scalar k can be found in Figure 4, where we can also apply our technique in the interval$[\lfloor n/2 \rfloor+1$, n-1].

Optimization can be done in the interval $[\lfloor n/2 \rfloor+1$, n-1] by using only even numbers for order n.

If the order n >2 is an even number:$\sum_{k=\lfloor n/2 \rfloor+1}^{n-1} kP$=3$\sum_{k=1}^{\lfloor n/2 \rfloor-1} kP$

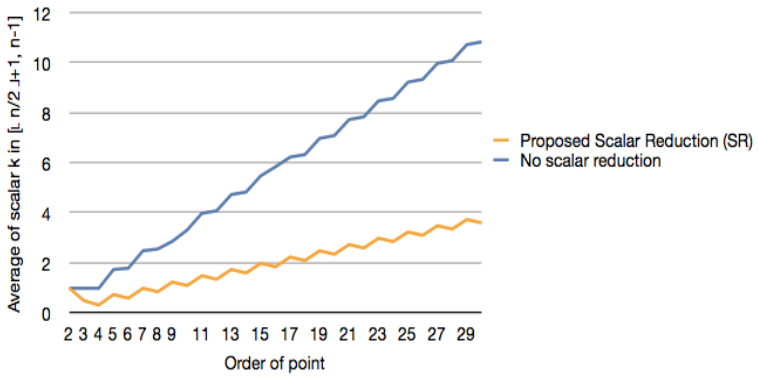

Fig. 4. Average of all k values function of order in [$\lfloor n/2 \rfloor$+1, n-1]

If the order $n \geq 3$ is an odd number: $3\sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP > \sum_{k=\lfloor n/2 \rfloor + 1}^{n-1} kP \geq 2\sum_{k=1}^{\lfloor n/2 \rfloor - 1} kP$. Figure 5 shows the speed-up between even and odd numbers. We can see that

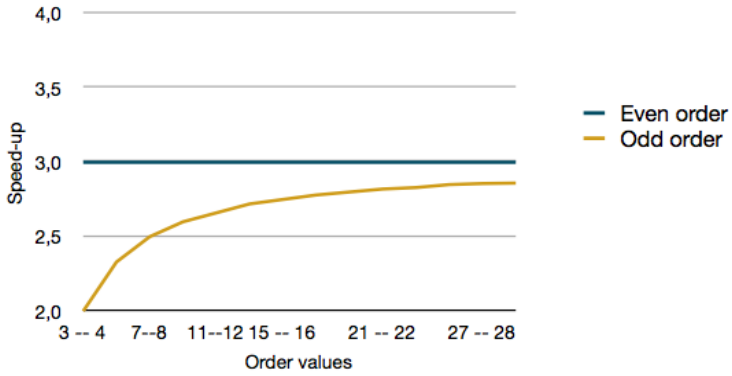

Fig. 5. Speed-up rate between even and odd order

if n is even, the speed-up curve is a horizontal line whose equation is y=3. But if n is odd, the line y=3 is a horizontal asymptote for the speed-up curve. If we work with even order, the speed is three times faster. If the order is odd, the enhancement is > 2, and < 3.

3.4 Efficiency analysis

The unit of this speed-up can be the length bit or the number of doubling/addition operations. For example, NAF needs $log_2(k)$ doubling and $log_2(k)/3$ addition. In the interval $[\lfloor n/2 \rfloor + 1, n-1]$, three cases are possible for the speed-up:

– If $log_2(n)$=x, where x is an integer, our method can speed up the scalar multiplication in interval $[\lfloor n/2 \rfloor + 1, n-1]$ by reducing the length bit of k.
– If $log_2(n)$=x, where x is not an integer, our method can speed up the scalar multiplication only in interval $[2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}, n-1]$
– If k= *(n-1)*, we reach the maximum speed-up, the length bit of the scalar k is maximum (equal to $log_2(n-1)$, where it is equal to one bit for d. In this case, our method does not require computation.

From Table 1 and Table 2, we can see the speed-up in length bit for some values of the scalar k.

Table 1. Speed-up S for some values of k for x integer.

Values of k	$\lfloor n/2 \rfloor + 1$	$>=(\lfloor n/2 \rfloor + 1)$	(n-1)
Speed-up(bits)	1	$1 < S < log_2(k)$	$log_2(k)$

Table 2. Speed-up S for some values of k for x not integer.

Values of k	$2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}$	$>= 2^{\lfloor log_2 \frac{n}{2} \rfloor + 1}$	(n-1)
Speed-up(bits)	1	$1 < S < log_2(k)$	$log_2(k)$

3.5 Performance Evaluation

To test the performance of our solution, we have implemented a simulator in Java. The program is then run on an Intel Core i5-2520 processor taking into account the computing power difference between this processor and a MSP 430 MCU. During the test, we choose an elliptic curve over $\mathbb{F}_p$ using NIST-192 recommended parameters which are given in Table 3. p is the size of prime field $\mathbb{F}_p$, and a, b are coefficients of the simplified Weierstrass form of our curve. $P(x_P, y_P)$ is chosen as the generator point, and its order equals n.

As indicated in equation (6.b), when we need to perform a scalar multiplication kP for cryptographic purpose, we choose $k < n$ where n is the order of the generator point P, which means $nP = \infty$. In addition, theoretically our method works only when $k \in]\lfloor \frac{n}{2} \rfloor, n-1]$. To prove this property, we have chosen 6 values of 192 bits for k which are distributed uniformly in $]0, n-1]$ (see table 4).

We have tested our method using both affine and jacobian coordinates, the scalars are represented respectively in binary and NAF form combined with the proposed scalar reduction method. The test results are given in tables 5 and 6, and illustrated graphically in figure 6 and 7.

Parameter	NIST-192 recommended values
p	$2^{192} - 2^{64} - 1$
a	-3
b	0x 64210519 e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1
x_P	0x 188da80e b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012
y_P	0x 07192b95 ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811
n	0x ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22831

Table 3. NIST-192 recommended elliptic curve parameters

k	Value in hexadecimal
$n/6$	0x 2aaaaaaa aaaaaaaa aaaaaaaa 99a5295e 58bca19d 9e2306b2
$n/3$	0x 55555555 55555555 55555555 334a52bc b179433b 3c460d65
$n/2$	0x 7fffffff ffffffff ffffffff ccef7c1b 0a35e4d8 da691418
$2n/3$	0x aaaaaaaa aaaaaaaa aaaaaaaa 6694a579 62f28676 788c1aca
$5n/6$	0x d5555555 55555555 55555555 0039ced7 bbaf2814 16af217a
$n-1$	0x ffffffff ffffffff ffffffff 99def836 146bc9b1 b4d22830

Table 4. Values of k chosen for performance evaluation

NAF	SR	$n/6$	$n/3$	$n/2$	$2n/3$	$5n/6$	$n-1$
		6579	6572	7604	6555	6931	7471
√		6282	6326	5317	6239	6698	5114
	√	6578	6573	7600	6416	6600	27
√	√	6279	6325	5320	6445	6556	28

Table 5. Running times (ms) using affine coordinates (SR: Scalar reduction)

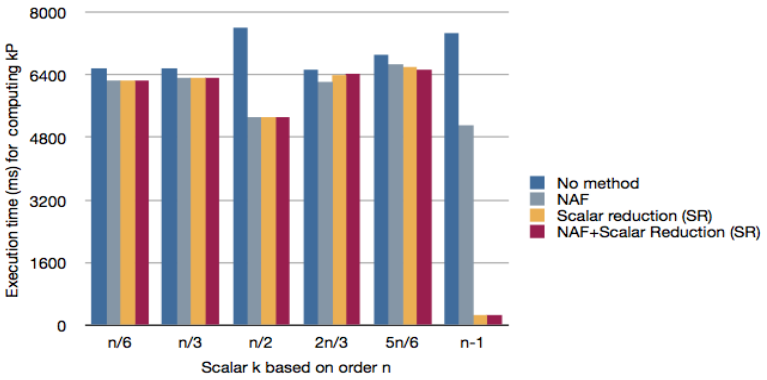

Fig. 6. Running times (ms) using affine coordinates

In both cases, we can notice that, firstly when the scalar is in NAF form, the computation is faster than the one using binary form. Secondly when $k \in]0, \frac{n}{2}]$, we cannot apply the proposed scalar reduction method. However, if $k \in]\frac{n}{2}, n-1]$,

we may accelerate slightly the computation by reducing the scalar. Especially when the value of k is close to $n-1$, the computation can be done instantaneously since $(n-1)P = -P$.

NAF	SR	$n/6$	$n/3$	$n/2$	$2n/3$	$5n/6$	$n-1$
		3066	3102	3621	3072	3202	3520
√		3053	3074	3592	3071	3189	3541
	√	3070	3100	3622	3030	3107	9
√	√	3050	3075	3597	3194	3136	9

Table 6. Running times (ms) using jacobian coordinates (SR: Scalar reduction)

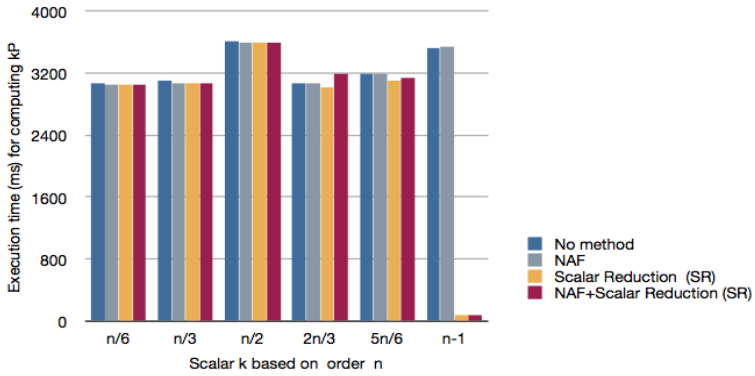

Fig. 7. Running times (ms) using jacobian coordinates

When we use jacobian coordinates, as we don't need to repeat the modular inverse (see equation (5)), the computation is obviously faster than the first case. Same as the results obtained using affine coordinates, our scalar reduction method can be used during the computation only when $k \in]\frac{n}{2}, n-1]$. When it's applied, the scalar is reduced, and the computation can run faster.

According to the results of performance evaluation, our scalar reduction method does speed up the computation of scalar multiplication on a standard NIST-192 elliptic curve. The acceleration rate strongly depends on the value of the scalar used. The scalar k can be reduced if $k \in]\frac{n}{2}, n-1]$, and once applied, the computation task can be simplified and carried out more quickly.

4 Conclusion

In this paper, we have proposed a novel method based on point order and the negative of point to speed up the computation of scalar multiplication on elliptic

curve cryptosystems. On one hand, the proposed method will significantly reduce the computation time in the interval $[\lfloor n/2 \rfloor + 1, n\text{-}1]$. On the other hand, we show that the usage of even order is more efficient than odd order. Our method is a very suitable tool for embedded devices such as WSNs. Also, it can be easily applied to almost all existing fast scalar multiplication methods as shown in NAF. Thus, that's comparisons are nor required regarding some existing schemes. Additionally, a thorough analysis and simulation based on evaluations will show that the proposed solution does speed up the computation of scalar multiplication on a standard NIST-192 elliptic curve. Our future research plans will be oriented towards experimenting our current technique on real sensor nodes with elliptic curves over finite prime fields.

References

1. M.J.B. Robshaw, and Y.L.Yin, Elliptic Curve Cryptosystems, An RSA hbomtories Technical Note, Revised June 27, 1997.
2. D. Hankerson, A. Menezes and S. Vanstone, Guide to Elliptic Curve Cryptography, Springer-Verlag, 2004.
3. D. M. Gordon, A Survey of Fast Exponentiation Methods, Journal of Algorithms, Academic Press, 1998, pp. 129-146.
4. V. Dimitrov V., L. Imbert, and P. K. Mishra, Efficient and secure elliptic curve point multiplication using double-base chains, Lectures Notes in Computer Science, 3788, 59-78 (2005).
5. M. Ciet, M. Joye, K. Lauter and P.L. Montgomery,Trading inversions for multiplications in elliptic curve cryptography, Designs, Codes, and Cryptography, 39, 189-206 (2006).
6. N. Meloni and M. A. Hasan, Elliptic curve scalar multiplication combining Yaos algorithm and double bases, in CHES 2009, 2009, pp. 304316.
7. M. Tian, Y. Wang, S. Xu, "An Efficient Elliptic Curves Scalar Multiplication Algorithm Suitable for Wireles Network" Second International Conference on Networks Security, Wireless Communication and trusted Computing, 2010, pp. 95-98, IEEE Computer Society 2010
8. V. Imai, E. Masato, Faster Multi-Scalar Multiplication Based on Optimal Double-Base Chains, World Congress on Internet Security (WorldCIS-2012), pp 93-98, IEEE 2012
9. P.LongaandA.Miri,Fast and Flexible Elliptic Curve Point Arithmetic over Prime Fields, in IEEE Transactions on Computers, Vol. 57, No 3, pp. 289-302, 2008.
10. B. Ansari, H. Wu "Parallel Scalar Multiplication for Elliptic Curve Cryptosystems" Communications, International Conference on Circuits and Systems, 2005. Proceedings. IEEE 2005 , pp 71-73, vol 1, 2005.
11. K. Wu, D. Li, H. Li, C. Yu, "Partitioned Computation to Accelerate Scalar Multiplication for Elliptic Curve Cryptosystems", 15th International Conferance on Parallel and Distributed Systems, 2009, pp. 551-555
12. Lim, C., Lee, P.: More flexible exponentiation with precomputation. In: Advances in Cryptology - CRYPTO94, Springer (1994) 95107
13. Y. Shou, H. Guyennet, and M. Lehsaini, "Parallel Scalar Multiplication on Elliptic Curves in Wireless Sensor Networks", 14th Int. Conf. on Distributed Computing and Networking (ICDCN), LNCS 7730,pp 300-314, Bombay, India, Jan 2013.