J. Basic. Appl. Sci. Res., 2(5)4577-4586, 2012
© 2012, TextRoad Publication
ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

CCS Representation: A New Non-Adjacent Form and its Application in ECC

Abdalhossein Rezai and Parviz Keshavarzi

Electrical and Computer Engineering Faculty, Semnan University, Semnan, Iran

ABSTRACT

This paper introduces CCS (complementary canonical sliding window) representation which is an efficient class of the non-adjacent-form (NAF). The CCS representation is an extension of the complementary representation. The proposed algorithm for converting the integer from the binary representation to the CCS representation uses the complementary method, the canonical recoding method and the sliding window method consecutively. Using Markov chain, we proved that the average Hamming weight of the CCS representation is $\frac{39n}{39w+80}$ for n-bit integer with window width *w*. In elliptic curve cryptosystem (ECC) implementation, the CCS representation is applied on the scalar multiplication to reduce the average number of the point addition/subtraction operation. Our analysis shows that the average Hamming weight of the CCS representation is reduced compared to other representations. Therefore, using the CCS representation in the scalar multiplication, the average number of the point addition/subtraction operation is reduced compared to other scalar multiplication algorithms considerably.

Keywords: signed-digit representation, non-adjacent form (NAF), scalar multiplication, canonical recoding, elliptic curve cryptosystem (ECC), high-speed arithmetic, public-key cryptography.

1 INTRODUCTION

Elliptic curve cryptosystem (ECC) which was introduced independently by Miller [1] and Koblitz [2], offers shorter keys and faster performance in comparison with other existing public key cryptosystems. These properties make ECC more suitable for using in the limited environments such as wireless sensor network, PDA and smart carts [3, 4, 5]. Thus ECC is now one of a well established and standardized public key cryptosystems (PKCs) [6].

The most important operation in ECC is the scalar multiplication. But, this operation is the most time consuming operation [5]. So, the major research efforts focused on the speed improvement of the scalar multiplication, especially on the integer representation of the scalar which plays an important role in the performance of the scalar multiplication [7, 8].

There are many research efforts to reduce the average Hamming weight of the integer representation such as: the non-adjacent form (NAF) [9, 10, 11, 12], the generalized NAF (gNAF) which is known as an efficient class of radix-r representation [13], the left-to-right (L2R) signed-digit radix-r representation [14, 15, 16], the width-*w* NAF (*w*NAF) [17, 18, 19, 20, 21, 22, 23, 24], the width-*w* radix-*r* NAF (*wr*NAF) [8, 9, 25], and the complementary representation [26, 27, 28, 29,30].

This paper presents CCS (complementary canonical sliding window) representation as a novel and efficient class of NAF. The CCS representation is extension of Chang et al.'s complementary representation [26]. For converting an integer from the binary representation to the CCS representation, the proposed algorithm uses the complementary method, the canonical recoding method and the sliding window method consecutively. The Markov chain is also used to prove that the average Hamming weight of the CCS representation is $\frac{39n}{39w+80}$ for n-bit integer with window width *w*. Moreover, the proposed CCS representation is applied on the scalar multiplication to reduce the average number of the point addition/subtraction operation. Our analysis shows that the average Hamming weight of the CCS representation and thereby the average number of required point addition/subtraction in the proposed scalar multiplication algorithm is reduced considerably.

The rest of this paper is organized as followings: section 2 briefly describes the background of the proposed representation algorithm. Section 3 presents the proposed representation and analyzed it. Section 4 investigates the CCS representation affects on the scalar multiplication. Results and discussion is presented in section 5. Finally, conclusion is given in section 6.

2 Background

2.1 Complementary recoding:

A complementary representation of an integer $k = \sum_{i=0}^{n-1} k_i.2^i$, $k_i \in \{0,1\}$ is a unique signed binary string which satisfies the following equation [26, 27, 28]:

$$k = \sum_{i=0}^{n-1} k_i.2^i = 2^n - \bar{k} - 1 \qquad (1)$$

***Corresponding Author:** Abdalhossein Rezai, Electrical and Computer Engineering Faculty, Semnan University, Semnan, Iran. Tel: +98-913-3060811,email: rezaie@acecr.ac.ir

where $\bar{k}$ is 1's complement of k and it is shown as $\bar{k} = \bar{k}_{n-1}\bar{k}_{n-2}\cdots\bar{k}_1\bar{k}_0$ in which

$$\begin{cases} \overline{k_i} = 0 & \text{if } k_i = 1 \\ \overline{k_i} = 1 & \text{if } k_i = 0 \end{cases} \quad \text{for } i = 0,1,...,n-1 \qquad (2)$$

For example for $k = (7967)_{10} = (1111100011\ 111)_2$, the number of bits in the binary representation is n=13 and the 1's complement of k is $\bar{k} = (0000011100\,000)_2$. So based on (1), the complementary representation of k is as follows: $k = 2^{13} - (0000011100000) - 1 = 8192 - 128 - 64 - 32 - 1$.

In this example, the Hamming weight of the integer k is reduced from 10 in the binary representation into 5 in the complementary representation which saves 5 point addition operations in the scalar multiplication. As each point addition requires 2S+2M+I (two finite field squaring (S), two finite field multiplication (M) and one finite field inverse (I) operation) [27, 28], this representation saves 10S+10M+5I finite field operations in this example.

2.2 Canonical recoding:

A signed-digit representation of an integer B is a sequence of digits $B = (b_{n-1}b_{n-2}...b_1b_0)$ such that $B = \sum_{i=0}^{n-1} b_i.2^i$, Where $b_i \in \{-1,0,1\}$. This integer representation is introduced by Booth [31]. The Booth's representation does not guarantee minimal Hamming weight for the integer representation. Reitwinsner [9] presented a canonical recoding method to convert integer from the binary representation to the signed digit representation. This recoding method which is also called non-adjacent form (NAF) guarantees the minimal Hamming weight. Algorithm 1 is used to convert an integer from the binary representation into its canonical representation.

Algorithm 1: The canonical recoding algorithm

Input: $B= (b_{n-1}b_{n-2}...b_1b_0)_2$
Output: $D= (d_nd_{n-1}...d_1d_0)_{SD}$
1. $c_0 := 0$;
2. For i = 0 to n
3. $c_{i+1} := \lfloor (b_i + b_{i+1} + c_i)/2 \rfloor$;
4. $d_i := b_i + c_i - 2c_{i+1}$;
5. Return D;

This algorithm scans input integer B from the least significant bit (LSB) to the most significant bit (MSB) and it is called right-to-left (R2L) algorithm. The average Hamming weight of an n-bit canonical recoded integer is at about $\frac{n}{3}$ [32].

For example for $B = (7967)_{10} = (1111100011111)_2$, the canonical recoded integer is $D = (10000\bar{1}00100000\bar{1})_{SD}$. In this example the Hamming weight of B is reduced from 10 in the binary representation into 4 in the canonical representation which saves 6 point additions/subtractions or 12S+12M+6I finite field operations in the scalar multiplication in comparison with the binary representation.

3 The proposed integer representation

3.1 The CCS representation

The CCS (complementary canonical sliding window) representation of an integer k is a sequence of digits $k = (k_{l-1}, k_{l-2}, ...k_1, k_0)$ where $k_i \in \{0, \pm1, \pm3, ..., \pm(2^w - 1)\}$. This novel signed-digit representation uses three methods: the complementary method, the canonical recoding method and the sliding window method. We proposed algorithm 2 to convert an integer from the binary representation to the CCS representation.

Algorithm 2: The proposed CCS recoding algorithm

Input: $k= (k_{n-1}k_{n-2}...k_1k_0)_2$
Output: $C= (c_{m-1}c_{m-2}...c_1c_0)_{SD}$
1. Count Hamming weight of k, denote as H(k)
2. If H (k)> n/2 then
3. $B = \bar{k}$;
4. Perform algorithm 1 and sliding window method on B to obtain k_{SD};
5. $C=2^n-k_{SD}-1$;
6. Else
7. B=k;
8. Perform algorithm 1 and sliding window method on B to obtain k_{SD}
9. $C=k_{SD}$;
10. End if
11. Return C;

In this algorithm, the input is an n-bit integer k= $(k_{n-1}k_{n-2}\ldots k_1k_0)_2$. The output is the CCS representation C= $(c_{m-1}c_{m-2}\ldots c_1c_0)_{SD}$. In the first step of this algorithm, H(k) as the Hamming weight of k is determined. Then, the Hamming weight of k is compared to the average Hamming weight of the binary representation ($\frac{n}{2}$). If H(k) is greater than $\frac{n}{2}$, the CCS representation is computed based on 1's complement representation of scalar k as follows:

$C=2^n-k_{SD}-1$;

In this case, The Hamming weight of the integer $B=\bar{k}$ is less than $\frac{n}{2}$ and the k_{SD} is obtained by applying the canonical recoding method and the sliding window method on $B=\bar{k}$.

On the other hand, when H(k) is less than $\frac{n}{2}$, the CCS representation is computed as follows:

$C=k_{SD}$

In this case, B=k has Hamming weight less than $\frac{n}{2}$ and k_{SD} is obtained by applying the canonical recoding method and the sliding window method on B=k.

In the CCS algorithm, similar to [33, 34, 35] the sliding window method scans the integer from the least significant digit (LSD) to the most significant digit (MSD) according to the state machine as shown in figure 1.

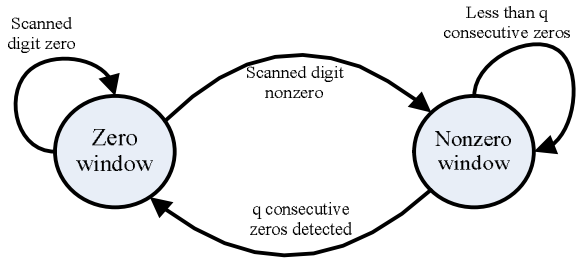

Figure 1: The sliding window state machine used in the CCS algorithm

This sliding window state machine starts from the zero window state (ZWS), and then the integer digits are checked one by one. If the incoming digit is zero, the finite state machine stays in ZWS, but if the incoming digit is nonzero, the finite state machine switches to the nonzero window state (NZWS). This state will not change as long as q consecutive zeros had not been collected. If this condition occurs, the automaton switches to ZWS. Otherwise, if *w* digits can be collected, the finite state machine stores the nonzero window and stays in NZWS to generate another nonzero window. It should be noted that after generating one digit of the canonical representation, the sliding window method is applied on it.

For example for k= $(1897423)_{10}$= $(111001111001111001111)_2$, the Hamming weight of the binary representation is 15, which is greater than $\frac{n}{2}=\frac{21}{2}=10.5$. So, the CCS representation is computed as following:

B= (00011000011000011000), after applying algorithm 1 on B, it convert to $D=(0001010\bar{1}0001010\bar{1}0001010\bar{1}0000)_{SD}$ and after using the sliding window method with *w*=3 and q=2 it convert to $k_{SD}=\{(000)(10\bar{1})(000)(10\bar{1})(000)(10\bar{1})(0000)\}_{SD}$. So, the result is $C=2^{21}-196608-3072-48-1$.

In this example the Hamming weight of the CCS representation is only 5, however, the Hamming weight of the binary representation, NAF and complementary representation is 15, 8 and 8 respectively.

As another example for $k=(24607)_{10}=(110000000011111)_2$, the Hamming weight of the binary representation is 7, which is less than $\frac{n}{2}=\frac{15}{2}=7.5$. So, the CCS representation is computed as following:

$B=(110000000011111)_2$, after applying algorithm 1 on B, it convert to $D=(10\bar{1}0000000100001\bar{1})_{SD}$ and after using the sliding window method with *w*=3 and q=2 it convert to $k_{SD}=\{(10\bar{1})(0000000)(1)(0000)(\bar{1})\}_{SD}$. So, the result is $C=24576+32-1$.

In this example the Hamming weight of the CCS representation is only 3, however, the Hamming weight of the binary representation, NAF and complementary representation is 7, 10 and 4 respectively.

3.2 Computational complexity analysis of the CCS representation

Assume that, k is an n-bit binary integer. In order to compute the average Hamming weight of the CCS representation, we can model it by using two Markov chains: one of them after using the complementary method and the canonical recoding technique, another after applying the sliding window method on the previous results.

According to [32] and based on the canonical recoding algorithm, all possible inputs and corresponding outputs of algorithm 1 are listed in table 1.

Table 1: State transition table for the canonical recoding algorithm

State		Output	Next State	
s_i	(b_{i+1}, b_i, c_i)	(d_i, c_{i+1})	$b_{i+2} = 0$	$b_{i+2} = 1$
s_0	(0, 0, 0)	(0, 0)	s_0	s_4
s_1	(0, 0, 1)	(1, 0)	s_0	s_4
s_2	(0, 1, 0)	(1, 0)	s_0	s_4
s_3	(0, 1, 1)	(0, 1)	s_1	s_5
s_4	(1, 0, 0)	(0, 0)	s_2	s_6
s_5	(1, 0, 1)	(1, 1)	s_3	s_7
s_6	(1, 1, 0)	(1, 1)	s_3	s_7
s_7	(1, 1, 1)	(0, 1)	s_3	s_7

In this table, the state transitions are produced by considering all 8 states labelled s_0 through s_7. For example s_2 represents $(b_{i+1}, b_i, c_i) = (0,1,0)$. In this state, the output is $(d_i, c_{i+1}) = (1,0)$ which computes from algorithm 1. Thus the next state is $(b_{i+2}, b_{i+1}, c_{i+1}) = (b_{i+2}, 0, 0)$. The complementary method is applied on the integer k to compute B, $P(b_{i+2} = 0) = \frac{3}{4}$ and $P(b_{i+2} = 1) = \frac{1}{4}$ [26]. Thus, there are transitions from the state $s_2 = (0,1,0)$ to the states $s_0 = (0,0,0)$ and $s_4 = (1,0,0)$ with probability $\frac{3}{4}$ and $\frac{1}{4}$ respectively. In this paper, the probability where the state s_i succeeds the state s_j is denoted by P_{ij}. So, from the above analysis $P_{20} = \frac{3}{4}$, $P_{24} = \frac{1}{4}$ and $P_{2j} = 0$ for $j = 1, 2, 3, 5, 6, 7$. Hence, the one step transition probability matrix of this method is given as follow:

$$P = \begin{bmatrix} 3/4 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 \\ 3/4 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 \\ 3/4 & 0 & 0 & 0 & 1/4 & 0 & 0 & 0 \\ 0 & 3/4 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 0 & 0 & 3/4 & 0 & 0 & 0 & 1/4 & 0 \\ 0 & 0 & 0 & 3/4 & 0 & 0 & 0 & 1/4 \\ 0 & 0 & 0 & 3/4 & 0 & 0 & 0 & 1/4 \\ 0 & 0 & 0 & 3/4 & 0 & 0 & 0 & 1/4 \end{bmatrix} \quad (3)$$

Let π_i be the limiting probability of the state s_i for $i = 0, 1, \ldots, 7$. The limiting probability for each state is found by solving the following system of linear equations [32]:

$$\pi.P = \pi \quad (4)$$

$$\sum_{i=0}^{7} \pi_i = 1 \quad (5)$$

This gives

$$\pi = [\frac{27}{52}, \frac{9}{208}, \frac{27}{208}, \frac{3}{52}, \frac{9}{52}, \frac{3}{208}, \frac{9}{208}, \frac{1}{52}]. \quad (6)$$

So, the probability of the zero digit and nonzero digits are $\pi_0 + \pi_3 + \pi_4 + \pi_7 = \frac{10}{13}$ and $\pi_1 + \pi_2 + \pi_5 + \pi_6 = \frac{3}{13}$ respectively. Thus, the average Hamming weight in this representation is $\frac{3n}{13}$ for n-bit length integers.

In order to obtain the average Hamming weight of the CCS representation, another Markov chain is required. This Markov chain has w+1 state. The probability where state 0 is succeeded by state 0 is $P_{00} = \frac{5}{8}$ since $P(D_{i+1} = 0 | D_i = 0) = \frac{\sum_{j=0,3,4,7} \pi_0 p_{0j} + \pi_3 p_{3j} + \pi_4 p_{4j} + \pi_7 p_{7j}}{\pi_0 + \pi_3 + \pi_4 + \pi_7} = \frac{5}{8}$. Similarly, the probability where state 0 is succeeded by state 1 is $P_{01} = \frac{3}{8}$. After w bits have been collected to form a nonzero digit, we have $P_{w0} = \frac{10}{13}$ and $P_{w1} = \frac{3}{13}$. These results are obtained from the previous Markov chain results. So, the one-step transition probability matrix P of the CCS algorithm for w=5 is given as following:

$$P=\begin{bmatrix} 5/8 & 3/8 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 10/13 & 3/13 & 0 & 0 & 0 & 0 \end{bmatrix} \quad (7)$$

The limiting probability for each state is found by solving the following system of linear equations

$$\begin{cases} \frac{5}{8}\pi_0 + \frac{10}{13}\pi_w = \pi_0 \\ \frac{3}{8}\pi_0 + \frac{3}{13}\pi_w = \pi_1 \\ \pi_i = \pi_{i+1} \quad \text{for } 1 \le i \le w-1 \\ \pi_0 + \pi_1 + \cdots + \pi_w = 1 \end{cases} \quad (8)$$

Let $p=\pi_1 = \pi_2 = \cdots = \pi_w$ so

$$\begin{cases} \frac{3}{8}\pi_0 - \frac{10}{13}p = 0 \\ \pi_0 + wp = 1 \end{cases} \quad (9)$$

thus

$$\pi_0 = \frac{80}{39w+80} \quad \text{and} \quad p = \frac{39}{39w+80}$$

Therefore, for large n, we can approximate the average Hamming weight of the CCS representation as following:

$$Ham(k) = \frac{39n}{39w+80} \quad (10)$$

4 The scalar multiplication algorithm using the CCS representation

The most important operation in ECC is the scalar multiplication which is defined by $Q = kP = \underbrace{P + P + \cdots + P}_{k \text{ times}}$ where P and Q are the elliptic curve points and k is a scalar. The prevalent method for performing the scalar multiplication is the binary (double-and-add) method [5, 36, 37]. There are two typical algorithms in binary method: the left-to-right (L2R) algorithm and right-to-left (R2L) algorithm. The L2R algorithm scans the scalar bits from the most significant bits while the R2L algorithm processes the scalar bits from the least significant bits. The L2R algorithm is widely used algorithm, because it can speed up the multiplication while the R2L algorithm requires extra memory to store the partial result. The L2R scalar multiplication algorithm [5, 38] is shown in algorithm 3.

Algorithm 3: The binary scalar multiplication algorithm

INPUT: $k=(k_{n-1}k_{n-2}...k_1k_0)_2$ **;P=(x,y);**
OUTPUT: Q=(x',y')=kP;
1. Q ← 0;
2. For i= n-1 to 0 do
3. Q=2Q;
4. If k_i**=1 then Q ← Q+P;**
5. Return Q;

In this algorithm, $k = \sum_{i=0}^{n-1} k_i.2^i$ such that $k_i \in \{0,1\}$. This algorithm scans the scalar bits from left-to-right. When $k_i \neq 0$, the point addition and point doubling operation are performed. But for $k_i = 0$, the point doubling operation is only performed. So, the integer representation (the length and Hamming weight of the scalar k) plays an important role in the performance of the scalar multiplication algorithm.

Using CCS representation, the average Hamming weight of the scalar is reduced from $\frac{n}{2}$ in the binary representation to $\frac{39n}{39w+80}$. Thus, the CCS representation can increase the speed of the scalar multiplication. Algorithm 4 shows how the CCS representation can be used in the scalar multiplication algorithm.

Algorithm 4: The proposed scalar multiplication algorithm

Input: $k=(k_{n-1}k_{n-2}...k_1k_0)_2$;$P=(x,y)$;
Output: $Q=(x',y')=kP$
Parallel begin
{recoding phase}
1. C=CCS(k);
{ pre-computation phase}
2. Compute and store v_iP for odd numbers of v_i
Parallel end
3. Q = 0
{evaluation phase}
4. For i = m - 1 to 0
5. $Q = 2^{l_i} Q$
6. If ($c_i > 0$) then $Q = Q + v_iP$
7. Else If ($c_i < 0$) then $Q = Q - v_iP$
8. Return Q

In this algorithm, m is the number of partitions in CCS representation, l_i is the length of ith partition, and v_i is the ith partition value. In recoding phase of this algorithm, the CCS representation of the scalar k is computed.

In the pre-computation phase of algorithm 4, the LSD of nonzero partition is either 1 or -1. So, the nonzero partition value is always an odd number. Hence, we only require pre-computation of v_iP for odd numbers of v_i in step 2. Moreover, the pre-computation phase of the proposed scalar multiplication algorithm is performed independently from recoding phase. Thus, these two phases can be performed in parallel.

In the evaluation phase of the proposed scalar multiplication algorithm, the point doubling operation is performed l_i times per iteration, but the point addition/subtraction operation is only performed for $c_i \neq 0$ (point addition operation for $c_i > 0$ and point subtraction operation for $c_i < 0$). As v_iP is computed in the pre-computation phase, the point addition/subtraction operation is only performed once for each $c_i \neq 0$.

5 RESULTS AND DISCUSSION

5.1 Evaluation of the CCS representation

As described in section 3.2, the average Hamming weight of the CCS representation for n-bit scalar k is $\frac{39n}{39w+80}$. However, the average Hamming weight of the NAF and complementary representation, gNAF and *wr*NAF are $\frac{n}{3}$, $\frac{n(r-1)}{r+1}$ and $\frac{n(r-1)}{w(r-1)+1}$ respectively.

The average Hamming weight of the CCS representation is reduced in comparison with the following representation by about:

$$1-\frac{\frac{39n}{39w+80}}{n/2}=1-\frac{78}{39w+80}$$ (Binary representation)

$$1-\frac{\frac{39n}{39w+80}}{n/3}=1-\frac{117}{39w+80}$$ (NAF and complementary representation)

$$1-\frac{\frac{39n}{39w+80}}{\frac{(r-1)n}{r+1}}=1-\frac{39(r+1)}{(r-1)(39w+80)}$$ (gNAF)

$$1-\frac{\frac{39n}{39w+80}}{\frac{(r-1)n}{w(r-1)+1}}=1-\frac{39(w(r-1)+1)}{(r-1)(39w+80)}$$ (*wr*NAF).

Figure 2 and table 2 summarize the comparison of the average Hamming weight of the CCS representation with the binary representation, NAF and complementary representation for n-bit scalar k and various window widths *w*.

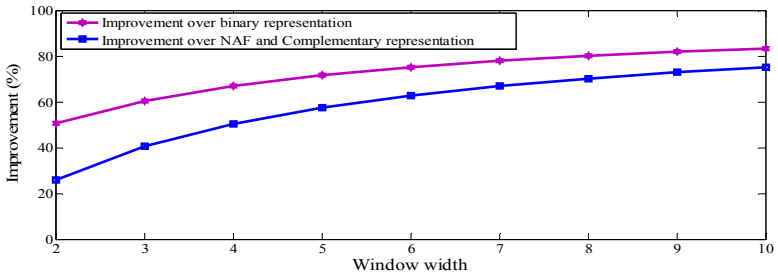

Figure 2: The average Hamming weight improvement over the binary representation, NAF and complementary representation for w=2-10.

Table 2: The comparative table for the average Hamming weight improvements over the binary representation, NAF and complementary representation

Representation	Average Hamming weight	Average Hamming weight improvement (%)								
		w=2	*w*=3	*w*=4	*w*=5	*w*=6	*w*=7	*w*=8	*w*=9	*w*=10
Binary	$\frac{n}{2}$	50.6	60.4	66.9	71.6	75.2	77.9	80.1	81.9	83.4
NAF and complementary	$\frac{n}{3}$	25.9	40.6	50.4	57. 5	62.7	66.9	70.2	72.9	75.1

As it is shown in figure 2 and table 2, the average Hamming weight of the CCS representation is reduced by about 50.6%-83.4% and 25.9%-75.1% in comparison with the binary representation and NAF and complementary representation respectively for *w*=2-10.

Moreover, figures 3-4 and table 3 summarize the comparison of the average Hamming weight of the CCS representation with the gNAF and *wr*NAF for various *w* and r.

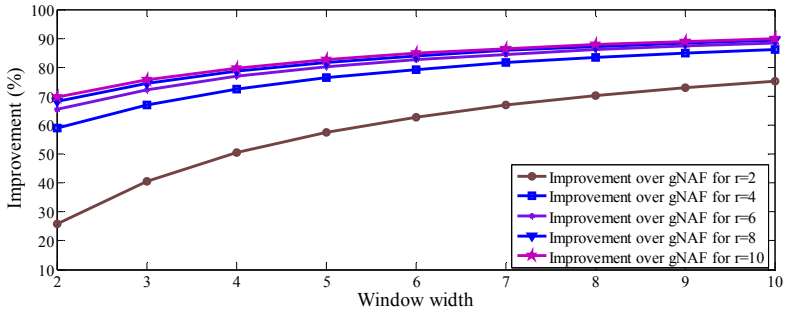

Figure 3: The average Hamming weight improvement over gNAF for w=2-10 and r=2-10

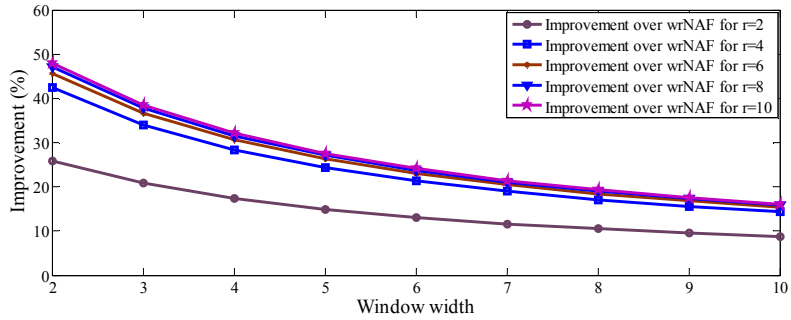

Figure 4: The average Hamming weight improvement over wrNAF for w=2-10 and r=2-10.

Table 3: The comparative table for the average Hamming weight improvements over the gNAF and wrNAF

Representation	Average Hamming weight	Average Hamming weight improvement (%)							
		r=2 *w=2*	*r=2* *w=10*	*r=4* *w=2*	*r=4* *w=10*	*r=8* *w=2*	*r=8* *w=10*	*r=10* *w=2*	*r=10* *w=10*
gNAF	$\frac{n(r-1)}{r+1}$	25.9	75.1	58.9	86.2	68.3	89.3	69.8	89.9
***wr*NAF**	$\frac{n(r-1)}{w(r-1)+1}$	26	8.7	42.4	14.3	47.1	15.8	47.9	16.1

As it is shown in figures 3-4 and table 3, the average Hamming weight of the CCS representation is reduced by about 25.9%-89.9% and 8.7%-47.9% in comparison with gNAF and *wr*NAF respectively for *w*=2-10 and *r*=2-10.

5.2 Evaluation of the proposed scalar multiplication

As the number of required point addition/subtraction operation in the scalar multiplication is determined by the Hamming weight of the scalar k, reducing the Hamming weight can speed up the scalar multiplication. So, the effect of the CCS representation on the scalar multiplication is considered in this subsection.

The average number of required point addition/subtraction operation in the scalar multiplication for various representations and operand size are computed and summarize in figure 5 and table 4.

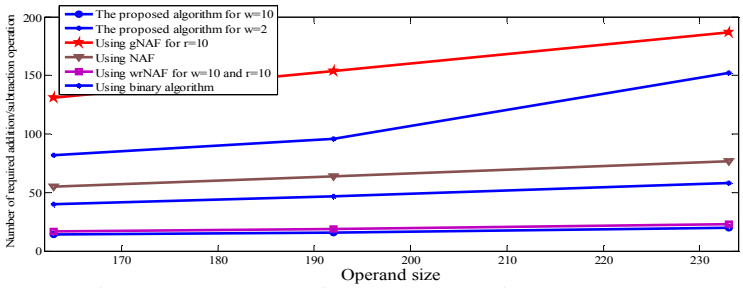

Figure 5: The comparison of the average number of required addition/subtraction operation in the proposed scalar multiplication algorithm and other scalar multiplication algorithms

Table 4: The comparative table for the average number of the required point addition/subtraction operation for various scalar representations

Scalar representation		Average Hamming weight	Average number of required point addition/subtraction operation		
			n=163	n=192	n=233
Binary		$\frac{n}{2}$	82	96	117
NAF and complementary		$\frac{n}{3}$	55	64	78
gNAF	r=2	$\frac{n(r-1)}{r+1}$	55	64	78
	r=10	$\frac{n(r-1)}{r+1}$	134	158	191
***wr*NAF**	r=2 *w*=2	$\frac{n(r-1)}{w(r-1)+1}$	55	64	78
	r=10 *w*=10	$\frac{n(r-1)}{w(r-1)+1}$	17	19	24
CCS	*w*=2	$\frac{39n}{39w+80}$	41	48	58
	w=10	$\frac{39n}{39w+80}$	14	16	20

As it is shown in figure 5 and table 4, the average number of required point addition/subtraction operation in the proposed scalar multiplication algorithm is reduced in comparison with the scalar multiplication algorithm which uses other representations. As described in section 2, each point addition requires 2S+2M+I (two finite field squaring (S), two finite field multiplication (M) and one finite field inverse (I) operation). So, the speed of the proposed scalar multiplication is increased compared to the scalar multiplication algorithm which uses other representation considerably.

6 Conclusion

As the integer representation has an important role in the computer arithmetic, major researches have been done in this area. This paper presents the CCS representation as a novel non-adjacent form (NAF) which uses the advantages of three recoding methods simultaneously: the complementary method, the canonical recoding method and the sliding window method. The main idea in this novel representation is the integer Hamming weight reduction by applying the canonical recoding method [9] and the sliding window method on the Chang et al.'s complementary recoding method [26]. Moreover, the Markov chain is used to analyze the average Hamming weight of the CCS representation. We proved that the average Hamming weight of the CCS representation is $\frac{39n}{39w+80}$ for n-bit integer with window width *w*. Using this representation in the scalar multiplication algorithm, the average number of the point addition/subtraction operation is reduced considerably. Our analysis shows that the average Hamming weight of the CCS representation is reduced at about 50.6%-83.4%, 25.9%-75.1%, 50.6%-90.7%, and 8.7%- 48.7% in comparison with the binary representation, NAF and complementary representation, gNAF and *wr*NAF respectively

for w=2-10 and r=2-10. Therefore, the proposed representation is particularly able to improve the speed of computing scalar multiplication and consequently ECC implementation for cryptography applications.

REFERENCES

[1] Miller,V., 1985. Uses of elliptic curves in cryptography. In proceedings of Advance in Cryptology (CRYPTO), Lect. Notes Comp. Sci. 218, Springer, pp. 417–428.

[2] Koblitz, N., 1987. Elliptic curve cryptosystem. Math. of comput., 48: 203-209.

[3] Dan, Y., Zou, X., Liu, Z., Han, Y., and Yi, L., 2009. High-performance hardware architecture of elliptic curve cryptography processor over GF(2^{163}). J. Zhejiang Univ. Sci. A, 10(2): 301-310.

[4] Osmani, A., 2012. Design and evaluation of new intelligent sensor placement algorithm to improve converage problem in wireless sensor networks, J. Basic. Appl. Sci. Res., 2(2):1431-1440.

[5] Blake, I., Seroussi, G., and Smart, N., 1999. Elliptic Curves in Cryptography. Cambridge University Press, UK.

[6] Avanzi R., Heuberger C., and Prodinger H., 2011. Redundant τ-adic expansions I: non-adjacent digit sets and their applications to scalar multiplication. Des. Cods. Cryptogr., 58(2): 173-202.

[7] Takagi, T., Jr, D., Yen, S. and Wu, B., 2006. Radix-r non-adjacent form and its application to pairing-based cryptosystem. IEICE. Trans. Fundam., E89-A(1): 115-122.

[8] Qin, B., Li, M, Kong, F. and Li, D., 2009. New left-to-right minimal weight signed-digit radix-r representation. Comput. Electr. Eng., 35(1): 150-158.

[9] Reitwiesner, G., 1960. Binary arithmetic. Adv. Comput., 1: 231-308.

[10] Morain, F., and olivos, J., 1990. Speeding up the computations on an elliptic curve using addition-subtraction chains. RAIRO theor. inform. appl., 24: 531-543.

[11] Arno, S., Wheeler, F., 1993, Signed digit representations of minimal Hamming weight. IEEE Trans. Comput., 42(8): 1007–1010.

[12] Joye, M., and Yen S., 2000. Optimal left-to-right binary signed-digit recoding. IEEE Trans. comput., 49(7): 740-748.

[13] Clark, W, Liang J., 1973. On arithmetic weight for a general radix representation of integers. IEEE trans. Inform. Theor., 19(3): 823–826.

[14] Joye, M., and Yen, S.,2002. New minimal modified radix-r representation with applications to smart cards. In proceedings of 5th international workshop on practice and theory in public key cryptosystems, (PKC02), Lect. Notes Comp. Sci.2274, Springer, pp.375-378.

[15] Kong, F., Yu, J., Cai, Z., and Li, D., 2006. Left-to-right generalized nonadjacent form recoding for elliptic curve cryptosystems. In proceedings of the IEEE. international conference on hybrid information technology (ICHIT2006), pp. 299-303.

[16] Muir J., 2007. A simple left-to-right algorithm for minimal weight signed radix-r representations. IEEE trans. Inform. Theor.53(3): 1234–1241.

[17] Miyaji A., Ono, T., and Chen, H., 1997. Efficient elliptic curve exponentiation. In proceedings of the international conference on Information and Communications Security (ICICS'97), Lect. Notes Comp. Sci. 1334, Springer, pp. 282-291.

[18] De Win, E., Mister,S. Preneel, B., and Winer, M., 1998. On the performance of signature schemes based on elliptic curves. In proceedings of the international symposium on Algorithmic number theory (ANTS), Lect. Notes Comp. Sci. 1423, Springer, pp. 252-266.

[19] Solinas, J., 2000. Efficient arithmetic on Koblitz curves. Des. Cods. Cryptogr., 19(2-3): 195-249.

[20] Okaya K., Schmidt-Samoa K., Spahn, C., and Takagi, T., 2004. Signed binary representations revisited. In proceedings of the international conference onthe Advance in cryptology (CRYPTO2004), Lect. Notes Comp. Sci. 3152, Springer, pp. 123-139.

[21] Avanzi, R., 2005. A note on the signed sliding window integer recoding and its left-to-right analogue. selected areas cryptography, Lect. Notes Comp. Sci. 3357, Springer, pp.130-143.

[22] Khabbazian M., Gulliver, T., and Bhargava, V., 2005. A new minimal average weight representation for left-to-right point multiplication method. IEEE. Trans.comput., 54(11): 1454-1459.

[23] Muir J., and Stinson, D., 2005. New minimal weight representation for left-to-right window methods. Lect. Notes Comp. Sci. 3376, Springer, pp. 366-383.

[24] Backenius, E., Sall, E., and Gustafsson, O., 2006. Bidirectional conversion to minimum signed-digit representation. In proceedings of the IEEE.international symposium on the circuits and system, Island of Kos, pp. 21-24,

[25] Takagi T., Yen S., and Wa, B., 2004. Radix-r non-adjacent form. In proceedings of the international conference on the information security (ISC'2004), Lect. Notes Comp. Sci. 3225, Springer, pp. 99-110.

[26] Chang, C., Kuo, Y., and Lin, C., 2003. Fast algorithms for common multiplicand multiplication and exponentiation by performing complements. In proceedings of the 17th IEEE international conference on advanced information networking and applications (AINA 2003), pp. 807-811.

[27] Balasubramaniam P., and Karthikeyan, E., 2007. Elliptic curve scalar multiplication algorithm using complementary recoding," Appl. Math. comput., 190(1): 51-56.

[28] Balasubramaniam, P., and Karthikeyan E., 2007. Fast simultaneous scalar multiplication. Appl. Math. comput., 192(2): 399-404.

[29] Wang, B., Zhang, H., Wang, Z., and Wang, Y., 2007. Speeding up scalar multiplication using a new signed binary representation for integers. In proceedings of the international conference on the Multimedia content analysis and mining, Lect. Notes Comp. Sci. 4577, Springer, pp.277-285.

[30] Huang X., Shahand, P., and Sharma D., 2010. Minimizing Hamming weight based on 1's complement of binary numbers over GF(2^m). In proceedings of the IEEE. 12^{th} international conference on the advanced communication technology (ICACT 2010), pp. 1226-1230.

[31] Booth, A., 1951. A signed binary multiplication technique. J. Mech. Appl. Math., 4: 236-240.

[32] Egeciglu, O., and Koc ,C., 1994. Exponentiation using canonical recoding. Theor. Comput. Sci., 129(2): 407-417.

[33] Rodriguez-Henriquez, F., Saqib, N., Diaz-perez, A. and Koc, C., 2006. Cryptographic Algorithms on Reconfigurable Hardware. Springer, USA.

[34] Rezai, A., and Keshavarzi, P., 2011. High-performance Modular Exponentiation Algorithm by Using a New Modified Modular Multiplication Algorithm and Common-Multiplicand-Multiplication Method. In proceedings of the IEEE international World Congress on Internet Security (WorldCIS 2011), pp. 192-197.

[35] Rezai, A. and Keshavarzi, P., 2012. A new CMM-NAF modular exponentiation algorithm by using a new modular multiplication algorithm. Trends applied sci. res., 7(3):240-247.

[36] Dormale, G., and Quisquater, J., 2007. Area and time trade-offs for iterative modular division over GF(2^m): novel algorithm and implementations on FPGA. Int. J. electron., 94(5): 515-529.

[37] Rezai, A. and Keshavarzi, P., 2011. High-performance implementation approach of elliptic curve cryptosystem for wireless network applications. In proceedings of the IEEE international conference on consumer electronics, communications and networks (CECNet 2011), pp.1323-1327.

[38] Menzes, A., 1993. Elliptic curve public key cryptosystems. Springer verlog, USA.