Journal of Mathematics and Statistics 6 (1): 28-33, 2010
ISSN 1549-3644

An Improved Binary Method for Scalar Multiplication in Elliptic Curve Cryptography

[1]M.A. Mohamed, [1]M.R. Md Said, [1]K.A. Mohd Atan and [2]Z. Ahmad Zulkarnain
[1]Institute of Mathematical Research, Faculty of Computer Science and Information Technology,
[2]Faculty of Computer Science and Information Technology,
University Putra Malaysia, Serdang, Malaysia

Abstract: Problem statement: Until recently, many addition chain techniques constructed to support scalar multiplication operation have been proposed tailored to limited computational resources. In securing the efficiency of ECC point operation, the combinations of the two basic operations, point addition and doubling are mostly implemented. Using binary method, the operation of doubling depends solely on the length of binary representation itself, so the most probable way to reduce the total number of the whole operation is by reducing the number of addition operation. This limitation is quite problematic. **Approach:** In this study we proposed an improved binary method which reads input block by block basis. Instead of having to add one to current chain every time non zero digit appears, this method requires one addition for every non zero block. A mapping table is used to store all possible binary string and its decimal version. For every block, its decimal value is extracted from the table and this value will be added to the current chain. In return, it requires precomputations for all possible combination of input blocks. **Results:** The new method showed a significant reduction in the number of required additions and the magnitude of improvement varies according to the key size. **Conclusion:** The algorithm is suitable to be adapted into cryptographic system especially as the need for bigger key size is growing rapidly.

Key words: Scalar multiplication, elliptic curve, binary representation, addition chains

INTRODUCTION

Elliptic curve cryptography (Koblitz, 1987; Miller, 1985) was introduced in 1985. Elliptic Curve Cryptography (ECC) transforms a complex mathematical problem into an applicable computer algorithm. The scheme beats the capability provided by RSA, with a key length of 168 bits, it provides similar security height as RSA 1024 bits. One of the most important research areas in ECC is to improve scalar multiplication technique which aims at increasing efficiency.

In short, a point P on Elliptic Curve (EC) is transformed using a key k to another point Q using a scalar multiplication formula Q = kP. The cheapest operation of EC on computer system would be its point addition and point doubling. For any k, the calculation of kP is broken down into a series of additions and doublings.

Given an integer k, possibly starting from 1 (followed by 2), with allowable operations of addition and doubling of two previous terms to get a new one, our objective is to find the fastest way to reach k. From the computational complexity point of view, Downey *et al*. (1981) proved that the problem to find the smallest number of terms in the sequence is an NP problem which says that there is no known polynomial time algorithm to find an optimal solution. Huge interest was shown to produce a near optimal solution resulting from various techniques. Yao (1976), Zantema (1991) and Knuth (1981) discussed the resulting asymptotic values of addition chains.

This solution is customarily known as an addition chain (later addition subtraction chain) problem. The ascending terms from left to right in the addition chain is known as an addition sequence.

Definition 1: An addition chain for k is a sequence of positive integers of the form:

$$a_1 = 1, a_2,\ldots,a_s = k$$

such that $p \leq q < r$ where $a_r = a_p + a_q$. The length of the addition chain is equal to the number of element in the sequence other than the initial value, a_1.

Corresponding Author: M.A. Mohamad, Mohamed, Faculty of Computer Science and Information Technology, Institute of Mathematical Research, University Putra Malaysia, Serdang, 43400, Malaysia
Tel: +60-(0)3-8946-6579

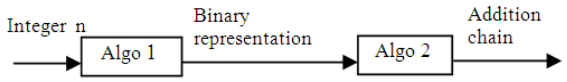

Fig. 1: Overview of addition chain technique

Definition 2: An addition subtraction chain for k is a sequence of positive integers of the form:

$$a_1 = \pm 1, a_2, \ldots, a_s = k$$

such that $p \leq q < r$ where $a_r = a_p \pm a_q$.

The process from having an integer k transformed into an addition chain can be divided into two parts as in Fig. 1; the first one is to use an algorithm to convert k into some sort of binary representation, the second one is to use another algorithm to perform some operations to produce the minimal chain. Both processes are chosen to make sure the computation is efficient. In some cases, the two processes are combined into one method.

The evolution of efficient algorithm for both parts of operations come hand in hand. For those algorithms, the number of addition and doubling depends on the number of non zero elements in the representation. Referred as Hamming weight, the number of non zero elements in the representation must be reduced to the least in order to achieve optimal efficiency.

Integer representation:
Definition 3: A positive integer N can be expressed by a summation of its coefficient multiplied by its radix representation:

$$N = \sum_{i=0}^{n-1} b_i r^i = b_{n-1} r^{n-1} + \ldots + b_1 r + b_0$$

Where:
b = The coefficient
r = The radix for m-ary representation

Take 23 as an example, it can be expressed in radix 10 as $2.10^1+3.10^0$, or in radix 2 as $1.2^4+0.2^3+1.2^2+1.2^1+1.2^0$.

Binary number is the most suitable representation for computer systems as it able to work directly with such input. No extra efforts required for conversion as it understand 0 and 1 perfectly. Many other representations emerged as a result of weaknesses within binary representation, all of which bears the same objective to increase the efficiency. We will take a developmental look into those representations.

Unsigned binary digit: An unsigned binary digit is the simplest form of binary number to represent an integer N. The use of symbols 0 and 1 dated back to Leibniz in 17th century. This classical binary representation of length v for N can be represented as:

$$N = \sum_{i=0}^{n-1} b_i 2^i = b_{n-1} 2^{n-1} + \ldots + b_1 2 + b_0$$

where, $0 \leq b_i < 2$.

This representation allows coefficient set of {0, 1}. There is one to one correspondence between N and its binary representation. The hamming weight of this form can be determined by counting the number of its ones. On average it is expected to be $\frac{v}{2}$.

Signed binary digit: Out of unsigned binary digits, to reduce hamming weight, Booth (1951) introduced the idea of signed digit representation and turned previous expression into:

$$N = \sum_{i=0}^{n-1} b_i 2^i = b_{n-1} 2^{n-1} + \ldots + b_1 2 + b_0$$

where, $-2 < b_i < 2$.

The technique expands the coefficient to $\{0,\pm 1\}$. Using this representation, the original addition chain is transformed into addition subtraction chain as described in (Knuth, 1981). Unfortunately, this representation is not unique. Using the same example, 23 can be represented as $11\bar{0}0\bar{1}$ or $10\bar{1}0\bar{0}1$.

Non adjacent form: Signed binary digit is further improved to the so called Non Adjacent Form (NAF) (Reitwiesner, 1960) to consist of minimal weight. In addition it is proved that every integer has a unique form of this sort. The conception is to disallow two consecutive non zero bits, i.e., $b_i b_{i+1} = 0$. A simple hand calculation method for computing NAF using the formula 3N-N is shown in (Chang and Tsao-Wu, 1979). However, Reitwiesner (1960) and Mandelbaum (1967) exemplified a computer algorithmic code based on an iterative method to generate NAF from an unsigned binary as well as signed representation. An expression for N can be reduced to:

$$N = \sum_{i=0}^{n} b_i 2^i = b_n 2^n + b_{n-1} 2^{n-1} + \ldots + b_{r+1} 2^{r+1} + b_r 2^r + \ldots + b_1 2^1 + b_0$$

Where:

$$b_i = \begin{cases} +1,0 & i \geq n \\ \pm 1,0 & i < n \end{cases}$$

NAF representation adds an extra bit to the left most of the original representation. Such an iterative process can be described in steps as follows:

Step 1: We start at the digit $b_r = 1$. If $b_r.b_{r+1} = 1$, we search through b_{r+1}, b_{r+2} and so on until we find $b_{r+c} = 0$

Step 2: Apply the following substitution to the partition

Step 3: Repeat Step 1 and Step 2 until the last bit:

$$\sum_{i=r}^{r+c} b_i 2^i = 2^{r+c} - 2^r$$

Algorithm 1, known as Reitwiesner right-to-left binary method works out as such, given N = 23, the relevant unsigned binary follows this conversion sequence, $10111 \rightarrow 1100\bar{1} \rightarrow 10\bar{1}00\bar{1}$. Jedwab and Mitchell (1989) also came out with an alternative iterative method called Weight Minimization Algorithm (WMA) to compute NAF and theorems for the sparseness and uniqueness of the output:

A1. INPUT: $b = \{b_{n-1},\ldots,b_0\}$
A2. $c_0 = 0$; $b_{n+1} = 0$; $b_n = 0$
A3. FOR i from 0 to length(b) step up by 1 DO
A4. $c_{i+1} = \text{floor}[(c_i + b_i + b_{i+1})/2]$
A5. $b'_i = c_i + b_i - 2c_{i+1}$
A6. OUTPUT: $(b'_n, b'_{n-1},\ldots,b'_0)$

Algorithm 1:

Reitwiesner right-to-left method: Morain and Olivos (1990) showed that the expected non zero length of NAF can be rounded to $\frac{v}{3}$. A left-to-right version for computing NAF is given by (Joye and Yen, 2000). There are other alternative to NAF called Mutual Opposite Form (MOF) (Okeya *et al*., 2004) but for the purpose of our comparison analysis, NAF is likely to be sufficient since both NAF and MOF has the same non zero density.

MATERIALS AND METHODS

Addition chain: The question is how addition technique makes use of different number representation to produce an optimal chain? The technique either processes the binary representation bit by bit basis or in a partition. Here we examine some of the well known methods ever crafted.

Naive: Let k = 5, to calculate Q = 5P, naïve method proceed as P+P+P+P+P, 4 times. In total k-1 additions are performed. This method is consuming maximum time possible and is considered as no algorithm.

Binary method: Knuth (1981) gives a good history of this particular technique. The idea is based on "square and multiply" technique for efficient exponentiation of power to be used in RSA. Even so, without having to change the conceptual essence, by adapting "square and multiply" to "double and add", the technique can be used to improve elliptic curve point calculation.

A left-to-right binary method is one of its two variations which scans binary input from the most significant bit right through to the least significant bit. The most significant bit is chosen to be digit 1 from the left most of binary representation which means that we need to skip any unnecessary 0 before meeting 1:

A1. INPUT: $k = \{k_{n-1},..,k_0\}$
A2. y = 1, z = 1
A3. FOR i from length(k)-1 to 0 step-down by 1 DO
A4. y = y + y
A5. IF $k_i = 1$, THEN
A6. y = y + z
A7. OUTPUT: (y)

Algorithm 2:

Binary left-to-right: Algorithm 2 shows a procedure to simulate left-to-right method. Using k = 59, the generated addition chain is 1, 2, 3, 6, 7, 14, 28, 29, 58, 59. On average, this method requires v number of doublings and $\frac{v}{2}$ number of additions. The total number of operations is given by $\frac{3}{2}v$. This method only operates on the current value of y hence only a single storage field is required. There is an alternative binary method called right-to-left method which scans input from the least significant bit. Compare to left-to-right method, this method requires one extra operation which is an addition by unity. Another drawback is that it also needs extra memory to store the value from previous doubling 2^{n-1}.

Addition subtraction method: This method was introduced by Morain and Olivos (1990) aiming at reducing the number of addition operations by reducing non zero density. It exploits the fact that to calculate the inverse point on the elliptic curve is at insignificant cost. Subtraction is reduced to addition of negative point of the curve. Having NAF as an input, the original

binary method is modified to handle a negative coefficient, to the following Algorithm 3:

A1. INPUT: k = {k_n,k_{n-1},..,k_0}
A2. y = 1, z = 1
A3. FOR i from length(k)-1 to 0 step-down by 1 DO
A4. y = y + y
A5. IF k_i =1, THEN
A6. y = y + z
A7. IF k_i = -1, THEN
A8. y = y – z
A9. OUTPUT: (y)

Algorithm 3:
Add-sub left-to-right: Again using k = 59, the generated addition chain is 1, 2, 4, 8, 16, 15, 30, 60, 59. This chain is one shorter than the chain from binary method. On average, this method requires v number of doublings and $\frac{v}{3}$ number of additions. The total number of operations is given by $\frac{4}{3}v$. Similar to binary method, this method is also available for right to left computation.

Proposed method: We proposed an enhanced binary method which we can have some control in the number of point addition operations. The first step is to partition the binary representation into blocks of equal size. Instead of non zero density, the number of additions varies according to the number of non zero blocks. If there is an odd block at the end, sufficient padding bit will be appended to the left of this block prior extracting its decimal value. The number of block is the rounded ratio between the length and the block size. During precomputation, a Table 1 is generated to hold mappings between 2^s combinations of {0,1} and its decimal counterparts where s is the size of the block.

Each decimal value in the Table 1 shown must be precomputed and in our case of s = 4, 14 precomputations are required to compute 2, 3,..,15. By setting s = 1, no precomputation hence no table is required and this procedure is similar to an ordinary binary method.

Table 1: Mapping table for s = 4

Binary	Decimal	Binary	Decimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	10
0011	3	1011	11
0100	4	1100	12
0101	5	1101	13
0110	6	1110	14
0111	7	1111	15

The algorithm may take as its input an unsigned binary as well as NAF. At running time, the program reads input on block by block basis specified by the block size until no more block is available. It operates from left to right. Every time a new block comes in, the content is compared to “Binary” column from Table 1, which in turns is mapped to a decimal value under the “Decimal” column as for the return. If this is our very first block, only addition is performed to the current chain otherwise s times doublings followed by one addition need to be performed:

A1. INPUT: k={k_{n-1},..,k_0}
A2. y = 1, z = 0
A3. q = v÷s, r = v mod s
A4. IF r>0 THEN
A5. k = {0..0}$^{s-r}$+ k
//Append {0..0}$^{s-r}$ padding bits to k
A6. q = q + 1
A7. FOR i from q*s-1to 0 step-down by s DO
A8. z = match(k[i, i-1, i-2, i-3])
//Function match return the decimal value z
A9. FOR j from 0 to s-1 step-up by 1 DO
y = y + y
A10. IF z > 0 THEN
A11. y = y + z
A12. OUTPUT: (y)

Algorithm 4:
Proposed method: The whole idea of our method described above is implemented as in Algorithm 4. It consists of more lines but the complexity is no different to others. The loop is executed at most v times.

RESULTS

Table 2 shows the different number of needed addition and doubling operations, tested key size parameter of different values which were generated at random, as well as against other previous methods. As a result, our proposed method significantly reduced the total number of operations with respect to other methods and this in turn shall minimize the required computation time.

DISCUSSION

Considering an unsigned binary input, the maximum number of addition operations is approximated to $\frac{v}{s}$. Each time on encountering a block of all zero, this value will be reduced by one. As expected, the maximum number of doubling operations is given by v.

Table 2: Experimental comparison between proposed method and other existing methods

Method				Proposed method		
		Binary method	Add-sub method	s = 3	s = 4	s = 5
Precomputation		0	0	6	14	30
Key size = 160	ADD	88	52	54	40	32
	DBL	159	160	157	156	155
	Total	247	212	217	210	217
Key size = 384	ADD	202	117	128	96	77
	DBL	383	384	381	380	379
	Total	585	501	515	490	486
Key size = 512	ADD	265	168	171	128	103
	DBL	511	512	509	508	507
	Total	776	680	686	650	640
Key size = 1024	ADD	530	350	342	256	205
	DBL	1023	1024	1021	1020	1019
	Total	1553	1374	1369	1290	1254

This method is not intended to reduce the number of this operation. Regardless of precomputations, the total number of operations is given by $v\left(1+\frac{1}{s}\right)$.

From Table 2, the total number of operation is calculated by adding ADD and DBL fields, in case of our proposed method, the number of precomputation is also included. Specifically for our method, it was tested for block of size 3-5. It can be seen that the number of reduction in operations is proportionate to the key size. For instance, at s = 4, with key size = 160, the ratio of reduction between proposed method and addition-subtraction method is 0.9905, with key size = 384, it is 0.9780, with key size = 512, it is 0.9615 and with key size = 1024, it is reduced to 0.9368. Also, among different block sizes, using key size of 160 as suggested by the current standard, the most optimal chain is obtained when using s = 4.

A little less precomputations can be obtained by using NAF input. At s = 4, instead of 14, only 9 precomputations are required to compute 2, 3,..,10 and their respective inverse which is considered at no extra cost. Also, NAF suppose to have more zero density and possibly more zero blocks which could further reduce the number of additions.

CONCLUSION

In this study, we studied a possible improvement on the original binary method. At the expense of some precomputations, we showed how to obtain a shorter addition chain through avoiding having to do addition operation every time non zero digits appears by limiting it to one for each block. The method was experimented for different key size and the result shows that the optimal block size varies with the key size.

If this method were to be adapted into the standard, the protocol should be implemented such a way that, the block size is negotiable between the communicating parties during the initial phase and it should well be suited to the key size required by the current standard in order to obtain the best result.

REFERENCES

Booth, A.D., 1951. A signed binary multiplication technique. Q. J. Mech. Applied Math., 4: 236-240. http://bwrc.eecs.berkeley.edu/classes/icdesign/ee241_s01/PAPERS/archive/booth51.pdf

Chang, S.H. and N. Tsao-Wu, 1979. On the evaluation of minimum distance of binary arithmetic cyclic codes. IEEE Trans. Inform. Theor., 15: 628-631. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?isnumber=22647&arnumber=1054346&count=30&index=9

Downey, P., B. Leong and R. Sethi, 1981. Computing sequences with addition chains. SIAM J. Comput., 10: 638-646. http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=SMJCAT000010000003000638000001&idtype=cvips&gifs=yes

Jedwab, J. and C.J. Mitchell, 1989. Minimum weight modified signed-digit representations and fast exponentiation. Elect. Lett., 25: 1171-1173. DOI: 10.1049/el:19890785

Joye, M. and S.M. Yen, 2000. Optimal left-to-right binary signed-digit recoding. IEEE Trans. Comput., 49: 740-748. DOI: 10.1109/12.863044

Knuth, D.E., 1981. The Art of Computer Programming. Vol. 2. Seminumeral Algorithms. 2nd Edn., Addison-Wesley.

Koblitz, N., 1987. Elliptic curve cryptosystems. Math. Comput., 48: 203-209. http://www.jstor.org/pss/2007884

Mandelbaum, D., 1967. Arithmetic codes with large distance. IEEE Trans. Inform. Theor., 13: 237-242. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1054015

Miller, V.S., 1985. Use of Elliptic Curves in Cryptography. Springer-Verlag, New York, ISBN: 0-387-16463-4, pp: 417-426.

Morain, F. and J. Olivos, 1990. Speeding up the computations on an elliptic curve using addition-subtraction chains. Theor. Inform. Appli., 24: 531-543. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.347

Okeya, K., K. Schmidt-Samoa, C. Spahn and T. Takagi, 2004. Signed binary representations revisited. Lecturer Notes Comput. Sci., 3152: 123-139. DOI: 10.1007/b99099

Reitwiesner, G.W., 1960. Binary arithmetic. Adv. Comput., 1: 231-308. http://citeseer.ist.psu.edu/context/825067/0

Yao, A.C.C., 1976. On the evaluation of powers. SIAM J. Comput., 5: 100-103. http://scitation.aip.org/getabs/servlet/GetabsServlet?prog=normal&id=SMJCAT000005000001000100000001&idtype=cvips&gifs=yes

Zantema, H., 1991. Minimizing sums of addition chains. J. Algorithms, 12: 281-307. http://portal.acm.org/citation.cfm?id=105962