An Optimal Algorithm for Left -to-Right Signed-Binary Representation of Four Integers

K.Pradheep Kumar

***Abstract*-**In elliptic curve cryptography, the problem of computing an integer k from two points on the ellipse namely kp, p the computation of ap+bq+cr+ds, is performed from Shamir's method. In this paper an algorithm is proposed for computing the minimum joint weight for the integers a, b, c, d, examining them from left to right.

***Keywords*—**ECC protocol, JSF algorithm, joint weight, optimality, ISB representation, SB representation, SB bit.

I. Introduction

IN [2], Joye and Yen and Shamir proposed an algorithm to compute the optimal weight from the signed binary representation of an integer.

In the elliptic curve problem, Shamir's method proposes a solution to compute the joint weight of two integers, using signed binary representation.

Shamir's method requires scanning the bits of an integer from right to left and results in a higher joint weight.

In this paper we propose a new algorithm for finding the optimal joint weight of four integers.

For two points kp, and p on an ellipse, to compute the integer k, explicitly, this algorithm simplifies the complexities of Shamir's method.

In our method, we use an algorithm, which scans the bits of an integer from left to right. By this method the joint weight of the integers is lesser than that computed by Shamir's method and is optimal.

The digital signature algorithm requires the computation of ap+bq+cr+ds, where a, b, c, d are integers and p, q, r,s are the points on an elliptic curve. In ECC protocols, we commonly use a method of computing kp. The joint weight is the number of non – zero columns when the binary expansion is written one below the other.

For example consider the following:

In the table, shown below a problem is discussed for the JSF recording, for which the entries in the columns are made.

The first entry in the row labeled " Double" is 0 (This is the starting point in the computation. The entries in other columns are based on the bits of a, b, c, d. If the bit of a in a column is –1, the entry corresponding to –p is added. Similar operations apply to b and Q. It is clear that the number of additions depend on the joint weight of a, b, c, d. By minimising the joint weight the operations will be faster.

Manuscript received March 14, 2005.

K. Pradheep Kumar is with the School of EEE, College of Engineering, Anna University, Chennai 600025, India (phone: 91-44-24354886; mobile: 919840204256; e-mail: pradheepkumark@yahoo.com).

II. Algorithm

This method is based on the following fact.

Let d be an l-bit binary number (dl-1, dl-2,……d1, d0).

Then d can be written as

d=dl-1*2^l-1+dl-2*2^l-2+….+d12+d0

Since 2^x=2^x+1-2^x, we can express d as follows

D=(dl-1-0) 2^l+(dl-2-dl-1)*(2^l-1) +…. d1*2^2 + (d0-d1) 2+(0-d0)

D can now be expressed as an L+1-bit number

D=((dl-1-0), (dl-2-dl-1), (do-d1), (0-do)). Each digit can be 0, 1, or –1. Using this observation we make the following recording. Let a, b, c, d be four integers given in the following binary format

A=(al-1, al-2,a1,a0)

B=(bl-1, bl-2, …., b1,b0)

C=(cl-1, cl-2, …..c1,c0)

D=(dl-1,dl-2,…d1,d0)

Algorithm 1 gives a method to compute the new signed binary representation

II.I Algorithm

1. Convert The binary expansion of a, b, c, and d an intermediate signed – binary (ISB) representation P1, P2, P3, P4

   P1=((al-1-0), (al-2-al-1),…,(a0-a1),(0-a0))
   P2=((bl-1-0), (bl-2-bl-1),….(b0-b1), (0-b0))
   P3=((cl-1-0), (cl-2-cl-1),…(c0-c1), (0-c0))
   P4=((dl-1-0),(dl-2,dl-1),…(do-d1),(0-d0))

2. Convert P1, P2, P3, P4 into Q1, Q2, Q3, Q4 by going from left to right and performing any of the following replacements. In the replacements shown below the lower columns are P2, P3, P4. If a replacement is not possible neglect a column and consider the next two or three columns

Note that if u=1 then ~u= -1

C1 c2 c3 -> e1 e2 e3
D1 d2 d3 f1 f2 f3
A1 a2 a3 g1 g2 g3
B1 b2 b3 h1 h2 h3
Is a replacement, then so is
B1 b2 b3 h1 h2 h3
A1 a2 a3 -> g1 g2 g3
D1 d2 d3 f1 f2 f3
C1 c2 c3 e1 e2 e3

The two steps of algorithm 1 can be combined together. We may illustrate this with an algorithm as shown below:

II.I.I. Algorithm

Input: L-bit binary number expansion of a, b, c, d

A=(al-1, al-2,..a1,a0)
B=(bl-1, bl-2,….b1,b0)
C=(cl-2, cl-1,…..c1,c0)
D=(dl-1,dl-2,….d1,d0)
Output:(L+1) bit signed binary representation of a, b, c ,d given by u0, u1,u2,u3
U0=(u0,l, u0,l-1,….u0,1, u0,0)
U1=(u1,l, u1,l-1,….u1,1, u1,1)
U2=(u2,l,u2,l-1,….u2,1,u2,0)
U3=(u2,l,u3,l-1,….u3,1,u3,0)

1. u0,l <- al-1-0
2. u1,l <- bl-1 –0
3. u2,l <- cl-1 –0
4. u3,l <- dl-1 –0
5. for j form l-1 down to 0

5.1If j> 0 then
5.1.1 uo,j <- aj-1–aj
5.1.2 u1,j <-bj- 1–bj
5.1.3 u2,j< -cj- 1-cj
5.1.4 u3,j <-dj-1-dj
5.2else
5.2.1 u0,0 <-0-a0
5.2.2 u1,0 <- 0-b0
5.2.3 u2,0 <- 0-c0
5.2.4 u3,0 <- 0 – d0
5.3for I from 0 to 3 do
5.3.1 If ui,j+1 * ui,j= -1 and (u1-I, j+1=0 or u1-I,j+1*u1-I,j= -1) then
5.3.1.1 ui,j+1 <-0
5.3.1.2 ui,j <- - ui,j
5.3.1.3 next I
5.4If j<l-1 then
5.4.1 for I in 0 to 3 do
5.4.1.1 If ui,j+2*ui,j = -1 and ui,j+1=0 and u1-I,j+2=0 and u1-I,j+1 !=0 and u1-I,j=0 then
5.4.1.1.1 ui,j +2 <-0
5.4.1.1.2 ui,j+1 <- - ui,j
5.4.1.1.3 ui,j <- -ui,j
5.4.1.2 else if ui,j+2*ui,j= -1 and ui,j+1=0 and u1-I,j=0 and u1-I,j+2*u1-I,j+1= -1 then
5.4.1.2.1 ui,j+1 <- ui,j+2
5.4.1.2.2 ui,j <- ui,j+2
5.4.1.2.3 ui,j+2<-0
5.4.1.2.4 u1-I,j+1 <- -u1-I,j+1
5.4.1.2.5 u1-I,j+2 <- 0
next I
next j

Algorithm 2 can be combined with Shamir's method for computing aP+bQ+cR+dS, thereby we eliminate the necessity for storing Q1 and Q2

Example 1. Suppose a=2 b=8 c=9 d=10. The binary expansion of a, b, c, d is given by

A=0010
B=1000
C=1001
D=1010

The joint weight is 4

Applying step1 to obtain the ISB

P1=0 0 1 ~1 0
P2=1 ~1 0 0 0
P3=1 ~1 0 1 ~1
P4=1 ~1 1 ~1 ~1

The joint weight is 5

Now we need to apply step2 of the algorithm

Q1= 0000
Q2= 0101
Q3= 0101
Q4= 0101

Now the joint weight is 2, By the JSF method, the joint weight is 4, which is lesser. also in SB representation the same JSF is obtained

III. Proof of Optimality

Here, in this section we prove the optimality of the algorithm for four integers. As a first step the signed binary representation of two integers with joint weight w0. Here we obtain an Intermediate Signed Binary representation (ISB). In step2 of the algorithm the joint weight w1 is computed from the ISB. In other words the joint weight of the 4 integers resulting from this method is always less than or equal to the joint weight of the original signed binary representation. For the four integers there are several signed binary representations and the ISB representations. For this purpose we go in for the JSF representation. This proves that Algorithm 1 produces an output optimal weight. We then prove the existence of uniqueness of the ISB representation.

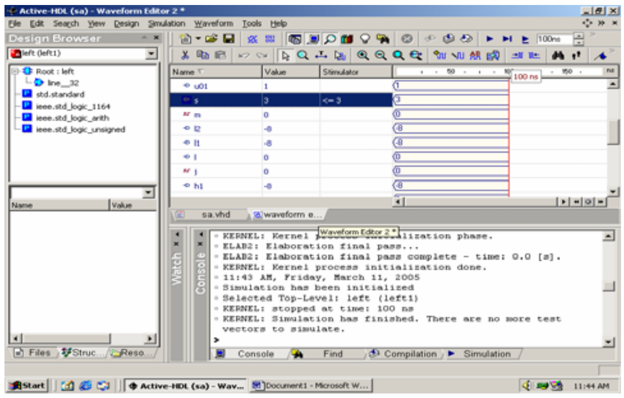

Fig 3.1 Wave Output

III.I. Optimality

A SB representation of a pair of two integers (4 integers) has a joint weight w0 and then from the next step of the algorithm1, it scans the ISB representation of the pair of two integers (4 integers), the scanning is done from left to right only once.

In [3], the ISB representation of an integer j>=0 exists and is unique

The existence of the above algorithm may be proved from the inverse of Algorithm 3 and vice versa. It may be also noted that both these algorithms are deterministic algorithms. Here the algorithm implies that a bijection is provided between the binary expansion and the ISB representation

III.I.I THEOREM I

If some bit is 0 in the original SB representation. And the algorithm 5 converts it to x i.e., xE {1, -1), then in the ISB representation, there exists x~ to its right.

Proof

Recall that R1 and R3a are only replacements that convert that convert 0 to x

i.e., x E{1,-1}. Now we consider R2, which may be applied after applying R1. Suppose the leftmost bit is non zero of a sequence S is x, then the bit remains unchanged.

TABLE III JOINT WEIGHTS AND TIMINGS BY THE JSF ALGORITHM AND OUR ALGORITHM

Evaluation of Efficiency									
Password bits	Joint weight from JSF(o1)	Joint Weight from Left to Right Algorithm(o2)	Time-JSF algortihm(t1)ms	Time-Left to Right algorithm(t2)ms	saving ms(t2-t1)ms	Time saved / JSF time as %	Efficiency JSF	Efficiency -left to right	Increase in Eff as %
80	40.8	43.8	17.9	13.5	4.4	24.58	51.00	54.75	7.35
120	60.6	64.7	27.3	19.7	7.6	27.84	50.50	53.92	6.77
163	82	84.2	34.9	25.3	9.6	27.51	50.31	51.66	2.68
233	116.3	118.9	50	40.6	9.4	18.8	49.91	51.03	2.24
250	125.5	127.9	52.9	41.9	11	20.79	50.20	51.16	1.91
283	142	145.3	58.9	47.5	11.4	19.35	50.18	51.34	2.32
350	175.6	178.3	72.4	58.9	13.5	18.65	50.17	50.94	1.54
409	205	207.4	83.2	65.8	17.4	20.91	50.12	50.71	1.17
450	225.3	228.2	92.6	71.3	21.3	23	50.07	50.71	1.29
571	285.9	287.9	114.8	92.6	22.2	19.34	50.07	50.42	0.70
Average						**22.08**			**2.80**

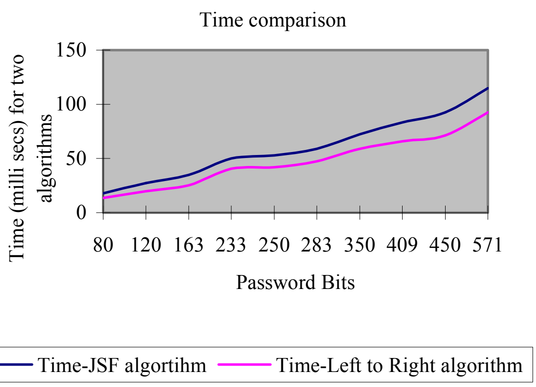

Fig 3.2 Plot to show Time Comparison of Algorithms

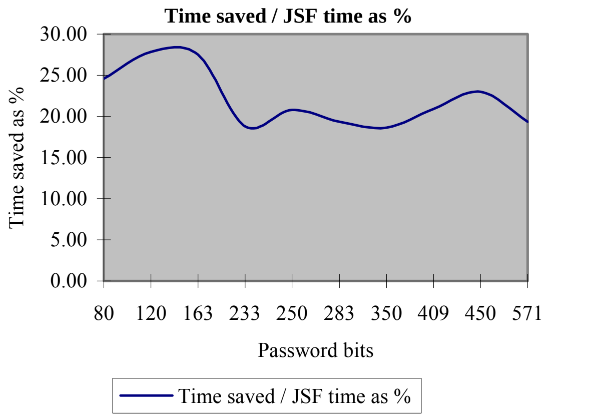

Fig 3.3 Plot to show time saving in Left to Right algorithm

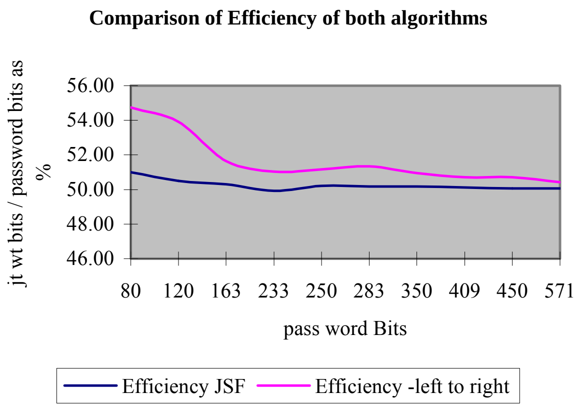

Fig 3.4 Plot to compare the efficiency of both algorithms

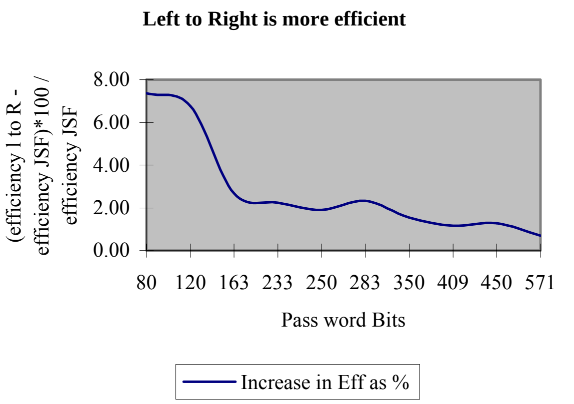

Fig 3.5 Plot to show that left to right our algorithm is more efficient

IV. Results and Conclusion

Our method scans the SB representation at most three columns at a time from left to right, thus to compute ap+bq+cr+ds, the memory required is only three SB bits for each integer. While using the JSF algorithm, the whole representation needs to be stored in memory before computing ap+bq+cr+ds. Thus, the memory required is (L+1) SB bits for each integer. Concerning hardware overhead our method is superior to the JSF algorithm.

Our method is implemented in hardware with the bits of a, b, c, d as inputs to compute the efficiency of 21% in optimisation of simulation time. Here each row of the table is obtained by generating randomly a million pairs of numbers. The second column is obtained by simulating the JSF algorithm.

From the Table 2 values we see that the joint weights obtained are approximately L/2. Here a left to right method is presented for a pair of two integers (4 integers), this may be extended for a higher number of integers.

The above graphs and table show that left to right algorithm saves time upto 22% and is 2.79 % more efficient compared to JSF algorithm. Left to right algorithm remains non vulnerable unless the locus of the curve is detected. However this will be very difficult and more time consuming.

References

[1] J.A. Solanais "*Low weight binary representations for pairs of integers*" Technical report CORR 2001-41, center for applied cryptographic research, Univ of Waterloo, Canada, 2001.

[2] M.Joye and S. sen "*Optimal left to right binary signed digit recording*" IEEE Transactions computers vol 49 no. 7, pp 740-748 July 2000.

[3] J. Lopez and R.Dahab, "*An overview of Elliptic curve cryptography*" technical report, Inst of computing state Univ of Campinas Brazil, May 2000.

[4] A.Menezes, *Elliptic curve public cryptographic systems* Kluwer Academic, 1993.

[5] I. Blake, G.Seroussi and N.Smart, *Elliptic curves in cryptography* London Math soc, lecture notes series 265, Cambridge University press, 1999.

[6] T.Elgamal, "*A public key cryptosystem and signature scheme based on discrete logarithms*", IEEE Transactions, Information Theory vol 31, pp 469-472, 1985.

[7] J.Proos, "*Joint Sparse forms and generating zero columns when combing*", Technical Report CORR 2003-23, center for Applied cryptographic research, Univ of Waterloo, Canada, 2003.

[8] A.D. Booth, " *A signed binary multiplication Technique*", Quarterly J. Mechanical applications Math, vol 4 , no.2, pp 236-240, 1951.

[9] Nat'1 Inst of standards and Technology, "*Digital signature standard*", FIPS publication.

[10] P.J Grabener, C. Heuberger, H.Prodigner, and J.Thuswaldner, "*Analysis of Linear combination algorithms in cryptography*", preprint.

TABLE I JSF ALGORITHM REPRESENTATION TO COMPUTE THE JOINT WEIGHT

	Integer							
	A	1	0	-1	0	-1	0	1
	B	0	-1	1	-1	0	1	0
	C	1	0	-1	1	1	-1	-1
	D	-1	1	0	0	-1	0	1
	Double	0	4P	8P+ 2Q+ 2R+ 2S	16P+ 4Q+ 4R+ 4S	32P+ 8Q+ 8R+ 8S	64P+ 16Q+ 16R+ 16S	128P+ 32Q + 32R+ 32S
Points	+P	P						129P+ 32Q+ 32R+ 32S
	-P							
	+Q		4P+ Q					
	-Q							
	+R			8P+ 2Q+ 3R+ 2S				129P+ 32Q + 33R+ 32S
	-R					32P+ 8Q+ 7R+ 8S		
	+S		4P+ Q+ S					
	-S					32P+ 8Q+ 9R+ 7S	64P+ 16Q+ 16R+ 15S	

TABLE II
REPLACEMENT TABLE

A1	0	U	→	0	U		0	0	→	0	0
	V	-v	→	0	V	Or	V	-v	→	0	V
	0	W	→	0	W		0	0	→	0	0
	X	-x	→	0	X		X	-x	→	0	X
A2	U	-u	→	0	U						
	V	-v	→	0	V						
	W	-w	→	0	W						
	X	-x	→	0	X						
A3	U	0	-u	→	0	U	U				
	0	V	0	→	0	V	0				
	W	0	-w	→	0	W	W				
	0	X	0	→	0	X	0				
A4	U	0	-u	→	0	U	U				
	V	-v	0	→	0	V	0				
	W	0	-w	→	0	W	W				
	X	-x	0	→	0	X	0				

