I.J. Computer Network and Information Security, **2014, 10, 57-63**
Published Online September 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijcnis.2014.10.07

Hardware Realization of Fast Multi-Scalar Elliptic Curve Point Multiplication by Reducing the Hamming Weights Over GF(p)

Nagaraja Shylashree
Research Scholar, PESCE, Department of E & C Engineering, Mandya, 571401, India
Email: shylashashi@gmail.com

Venugopalachar Sridhar
PESCE, Department of E & C Engineering, Mandya, 571401, India
Email: venusridhar@yahoo.com

Abstract—We present a new hardware realization of fast elliptic curve Multi-Scalar Point Multiplication (MSPM) using the sum of products expansion of the scalars. In Elliptic curve point Multiplication latency depends on the number of one's (Hamming Weight) in the binary representation of the scalar multiplier. By reducing the effective number of one's in the multiplier, the multiplication speed is automatically increased. Therefore we describe a new method of effectively reducing the Hamming weight of the scalar multipliers thereby reduces the number of Point Adders when multi scalar multiplication is needed. The increase in speed achieved outweighs the hardware cost and complexity.

Index Terms—Sum of products expansion, Hamming Weight, multi-scalar point multiplication, triple-scalar point multiplication, elliptic curve point multiplication, elliptic curve point addition.

I. Introduction

Elliptic curve cryptography (ECC) was proposed by Koblitz [1] and Miller in 1985 [2]. In the Specifications for public key cryptography mentioned by IEEE 1363 standard [3] and Recommended by NIST [4]. Elliptic Curve point multiplication is used in Diffi-Hellman type key agreement [8], Elliptic Curve Digital Signature Algorithm [9] and Elgamal crypto systems [10-11]. Innumerable hardware solutions are already available for elliptic curve Point Multiplication, [12-15]. Multi-Scalar Point Multiplication (MSPM) is also used in several elliptic curve cryptosystems. MSPM is mainly used in group key generation protocols in a key generation center [5]. Double and triple scalar multiplications are generally used in group key generation and distribution. Here, we describe the design for double and triple scalar multiplication with increase in the overall multiplication speed. Our two new hardware architecture is very well suited for both FPGA and ASIC realizations because of modular design.

The organization of this paper is as follows: Double Scalar Multiplication defined in Section II. Sections III discuss about Triple Scalar Multiplication. Finally Section IV concludes the work.

II. Double Scalar Multiplication

Let K and L be two scalar numbers in GF(p). Our objective is to generate K*P and L*P using Elliptic Curve Point Multiplication (PM) simultaneously [6]. Conventional hardware realization is shown in Fig.1.Schematic diagram for the realization of conventional left to right binary method [5] of PM is shown in Fig.2.

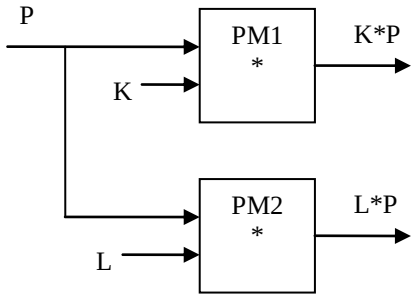

Fig.1. Conventional Point Multiplication

Elliptic Curve PM's are realized Point Doublers (PD's) and Point Adders [7]. In getting K*P, the number of PA's used is equal to the number of 1's in the binary representation of K. The number of 1's present in a binary number is called its Hamming weight. In scalar point multiplication using left to right method the number of point additions is equal to the number of 1's in the scalar multiplier [7].Therefore, by reducing the Hamming weight (number of 1's) of the scalar multiplier, the number of point additions are reduced. Hence higher speed is achieved. In a hardware set up, K and L are stored in the binary format. Let the size of K and L each be N bits. On the average, we can assume that the number

of 1's in an N bit binary digit, is 50 percent. That is, the number of 1's in K or L on the average, would be, 0.5*N. Then the total number of 1's from both K and L would be N. Therefore the number of PA's required for the conventional realization is,

$$n(PA, conventional) = N \quad (1)$$

In our new proposed scheme, the number of PA's required is reduced. This in turn increases the speed of Point Multiplication.

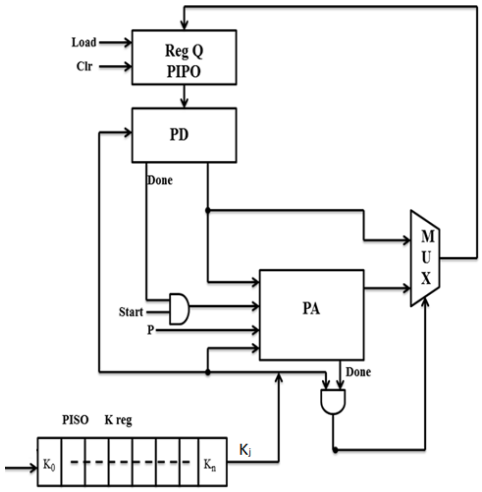

Fig.2. Hardware realization of conventional left to right binary method of PM

A. Basic Principle

Our new proposed scheme is based on a simple theorem involving binary representation of numbers. Let G and H be two binary numbers of equal sizes. Then the theorem gives the formula for the arithmetic sum of G and H as follows.

Theorem 1: Given two unsigned binary numbers G and H of same size, their arithmetic sum is given by,

$$G + H = (G \mid H) + (G \,\&\, H) \quad (2)$$

Here, $\mid$ is the bitwise OR operator and
& is the bitwise AND operator.
+ is the arithmetic addition operator.

Proof: Consider the truth table for arithmetic addition, Boolean OR and Boolean AND for two bits g and h as shown in **Table 5.1.**

Table.1. Relation between arithmetic sum and Boolean OR

g	h	g + h	g $\mid$ h OR	g & h AND	(g $\mid$ h)+ (g & h)
0	0	0	0	0	0
0	1	1	1	0	1
1	0	1	1	0	1
1	1	2	1	1	2

From the truth table, we see that,

$$g + h = (g \mid h) + (g \,\&\, h) \quad (3)$$

This relation holds good for all the bits of G and H. Therefore, (2) is proved.

Example 1. Let G = [1 0 0 1 1 0 0 1] = 153 and
H = [1 1 0 1 0 0 0 1] = 209.
Now G $\mid$ H = [1 1 0 1 1 0 0 1] = 217.
And G & H = [1 0 0 1 0 0 0 1] = 145.

From the above example, It can be verified that,

153 + 209 = 217 + 145.
i.e, G + H = (G $\mid$ H) + (G & H).

Lemma 1:

when (G & H) = 0 (all zeros), $G + H = (G \mid H)$ (4)

That is, the arithmetic sum and the bitwise Boolean OR of two unsigned binary numbers are same when their bitwise AND is all zeros. The proof of (4) is obtained from (2) by putting (G & H) =0. Another way of understanding this lemma is, when (G & H) =0, there is no carry from any bit position. Hence the arithmetic addition is same as bit wise OR operation.

Example 2. Let G = [1 0 0 1 1 0 0 1] = 153 and
H = [0 1 0 0 0 1 0 0] = 68.
Now G $\mid$ H = [1 1 0 1 1 1 0 1] = 221.
And G & H = [0 0 0 0 0 0 0 0] = 0.
Now, 153 + 68 = 221. That is, G + H = (G $\mid$ H).

B. Decomposition of K and L

In our new method, K and L are expanded, based on the well-known Boolean identity, as,

$$K = (K \,\&\, L) \mid (K \,\&\, \overline{L}) \quad (5)$$
$$L = (K \,\&\, L) \mid (\overline{K} \,\&\, L) \quad (6)$$

Since $(L \,\&\, \overline{L}) = 0$, $(K \,\&\, L) \,\&\, (K \,\&\, \overline{L}) = 0$. Then from Lemma 1,

$$(K \,\&\, L) \mid (K \,\&\, \overline{L}) = (K \,\&\, L) + (K \,\&\, \overline{L}) \quad (7)$$

From (7) and (5),

$$K = (K \,\&\, L) + (K \,\&\, \overline{L}) \quad (8)$$

Similarly, L can be expanded as,

$$L = (K \,\&\, L) + (\overline{K} \,\&\, L) \quad (9)$$

Therefore K*P can be expressed in the light of Eq. (8) as,

$$K*P = ((K \,\&\, L) + (K \,\&\, \overline{L}))*P$$

$$= (K \,\&\, L)*P + (K \,\&\, \overline{L})*P \quad (10)$$

Similarly, L*P can be expressed as,

$$L*P = (K \,\&\, L)*P + (\overline{K} \,\&\, L)*P \quad (11)$$

*C. Hardware Realization of K*P and L*P*

In (10), (K & L) and (K & $\overline{L}$) are scalar integers having a size of N bits each in binary format. Therefore two elliptic curve Point Multipliers and a Point Adder can realize (10) as shown in Fig. 3. Thus K*P is the output of PA_1. Now consider (11). In Fig. 3, (K & L)*P from PM_1 and ($\overline{K}$ & L)*P from PM_3 are added in PA_2 to get L*P. The bitwise operations for getting (K & L), (K & $\overline{L}$) and ($\overline{K}$ & L) are done in the Digital Logic Block. The new Double-scalar Point Multiplication scheme shown in Fig. 3. reduces the total number of Point Additions.

D. Calculation of the percentage saving in the number of PA's in the new scheme as in Fig.5.3

Consider the general scenario, where the number of 1's in K and L is randomly distributed as follows.

$$\text{prob(i th bit of K =1)} = p \quad (12)$$
$$\text{prob(i th bit of L =1)} = p \quad (13)$$

where $0 \le p \le 1$
Then, 0
prob (i th bit of K = 1 and i th bit of L =1) = $p * p$

$$= p^2 \quad (14)$$

$$\text{Hence, prob(i th bit of (K \& L) =1)} = p^2 \quad (15)$$

Therefore,

$$\text{the number of 1's in (K \& L)} = p^2*N \quad (16)$$

similarly,

$$\text{the number of 1's in (K \& } \overline{L}\text{)} = p*(1-p)*N \quad (17)$$

$$\text{the number of 1's in (} \overline{K} \text{ \& L)} = (1-p)*p*N \quad (18)$$

Therefore, the total number of PA's inside the three PM's is $p^2*N + p*(1-p)*N + (1-p)*p*N$. Counting the two external Adders PA1 and PA2, the total number of Additions in the new scheme designated by n(PA, new scheme) is,

$$n(PA, \text{new scheme}) = p^2*N + p*(1-p)*N + (1-p)*p*N + 2 \quad (19)$$

For the conventional multiplier shown in Fig. 1, The total number of 1's given by

$$n(PA, \text{conventional}) = p * N + p * N = 2*p * N. \quad (20)$$

Therefore, saving in the number of PA's = n (PA, conventional) – n (PA, new scheme)

$$= (2*p * N) - (p^2*N + p*(1-p)*N + (1-p)*p*N + 2)$$
$$= N* p^2 - 2. \quad (21)$$

Therefore, the percentage saving is

$$(N*p^2-2)/ (2 * p * N) \quad (22)$$

When N is large -2 can be neglected in the numerator. Therefore, the approximate percentage saving is p/2. In general, assuming p = 0.5 and assuming a 32 bit scalar, N =32. Then using (22), percentage saving would be 18.75%.

When N is large minus 2 can be neglected in the numerator of (22). Therefore, the approximate percentage saving is p/2. If p = 0.5, the percentage saving is 25%.

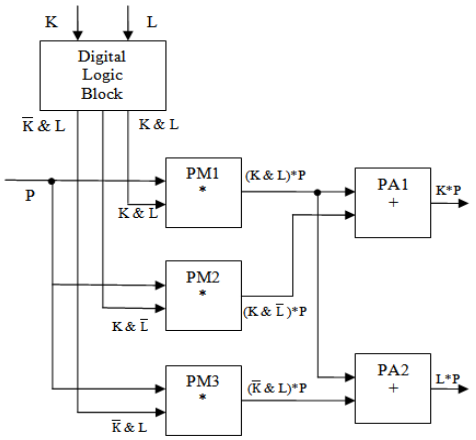

Fig.3. New Double-scalar Point Multiplication (DSPM) Scheme

In Fig.3. K and L are stored in N bit Parallel registers and serve as input to the digital logic block. The Outputs K * P and L * P are taken from PA1 and PA2. The digital logic block generates K& L, $\overline{K}$ & L and K & $\overline{L}$. Here, N and P are selected according to the NIST p-192 standard [4]. Algorithm 1 computes for the Double Scalar point multiplication (DSPM).

1. Algorithm for Double scalar point multiplication

Inputs: Scalar K, L and Elliptic curve point P
Outputs: K * P and L * P

1. Using the digital logic block with inputs K and L, get K& L, K&$\overline{L}$ and $\overline{K}$ & L.
2. Apply P and K&L to PM1 to get (K& L) *P
   Apply P and K&$\overline{L}$ to PM2 to get (K& $\overline{L}$) *P
   Apply P and $\overline{K}$&L to PM3 to get ($\overline{K}$& L) *P
3. Add (K& L) *P and (K& $\overline{L}$) *P using PA1 to get K * P
4. Add (K& L) *P and ($\overline{K}$& L) *P using PA2 to get L * P

III. Triple Scalar Multiplication

The principle described in section II can be extended to triple scalar Point Multiplication. Here the aim is to get K*P, L*P and M*P where K, L and M are three scalars of size N bits each. Conventional method uses three PM's as shown in Fig. 4. Take the probability that a bit in K, L or M is 1, as q. Then the the number of ones in K, L or M is q*N. Therefore, the total number of one's is 3*q*N. Hence the number of PA's is,

$$n(PA, triplePM, conventional) = 3 * q * N \quad (23)$$

Using our new scheme, the number of PA's can be reduced and there by the speed can be increased.

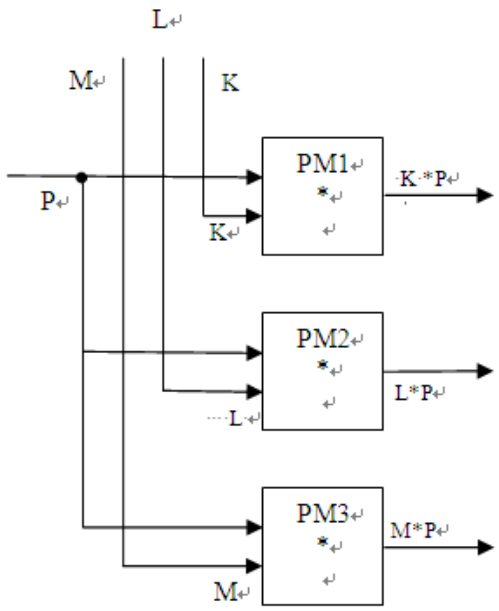

Fig. 4.Triple-scalar Point Multiplication, Conventional method

A. Triple-scalar multiplication, new scheme

The three scalars K, L and M are expanded using the Karnaugh maps. K is expanded as shown in Table 2.

Table 2. Karnaugh map for K

	$\overline{L}\ \overline{M}$	$\overline{L}$ M	L M	L $\overline{M}$
$\overline{K}$	0	0	0	0
K	1	1	1	1

The cells for K are shaded in gray. From **Table 5.2**, we can expand K in sum of product form as,

$$K = KLM + K\overline{L}\,M + K\overline{L}\,\overline{M} + KL\overline{M} \quad (24)$$

Karnaugh map for L is shown in Table 3.

Table .3. Karnaugh map for L

	$\overline{L}\ \overline{M}$	$\overline{L}$ M	L M	L $\overline{M}$
$\overline{K}$	0	0	1	1
K	0	0	1	1

From Table.3,

$$L = KLM + \overline{K}L\,M + \overline{K}\,L\overline{M} + KL\overline{M} \quad (25)$$

Similarly, Karnaugh map for M is shown in Table 4.

Table 4. Karnaugh map for M

	$\overline{L}\ \overline{M}$	$\overline{L}$ M	L M	L $\overline{M}$
$\overline{K}$	0	1	1	0
K	0	1	1	0

From Table 4,

$$M = KLM + \overline{K}L\,M + \overline{K}\,\overline{L}M + + K\overline{L}M \quad (26)$$

In (24), (25) and (26), KLM means (K & L & M) which is the bitwise ANDed K, L and M. Similarly other expressions like $K\overline{L}\,M$, *etc.* represent the bitwise AND operations of those variables. The + symbol represents bitwise OR operations as well as arithmetic additions depending on the context, because according to Lemma 1, both are same.

From (24),

$$K*P = (KLM)*P + (K\overline{L}\,M)*P + (K\overline{L}\,\overline{M})*P + (KL\overline{M})*P \quad (27)$$

Here, + operator represents Point Addition. Similar to Eq. (26), from Eqs. (24) and (25),

$$L*P = (KLM)*P + (\overline{K}LM)*P + (\overline{K}L\overline{M})*P + (KL\overline{M})*P \quad (28)$$

$$M*P = (KLM)*P + (\overline{K}LM)*P + (\overline{K}\,\overline{L}M)*P + (K\overline{L}M)*P \quad (29)$$

In (27), (28) and (29), we have 7 distinct minterms as the multipliers for P. Therefore, 7 PM's are required. To find the number of external adders, we write (27), (28) and (29) as,

$$K*P = ((KLM)*P + (KL\overline{M})*P) + ((K\overline{L}\,M)*P + (K\overline{L}\,\overline{M})*P) = (KL)*P + (K\overline{L})*P \quad (30)$$

Here, $((KLM)*P + (KL\overline{M})*P) = (KL)*P$ (31)

and $((K\overline{L}\,M)*P + (K\overline{L}\,\overline{M})*P) = (K\overline{L})*P$ (32)

Each Equation among (31), (32) and (30), requires one adder to realize it. Hence 3 adders are needed to realize K*P as shown in Fig. 5. In Fig. 5, PA1 realizes (31), PA2 realizes (32) and PA3 realizes (30). PA4 realizes ($\overline{K}$L)*P. Adder PA7 accepts (KL)*P, ($\overline{K}$L)*P and gives out L*P. Similarly, PA5, PA6 and PA8 give M*P. thus a total 8 external PA's are required.

The inputs to the adders are obtained using 7 PM's as shown in Fig.6. The minterm inputs to PM's are obtained using a digital logic block whose inputs are L, M and K.

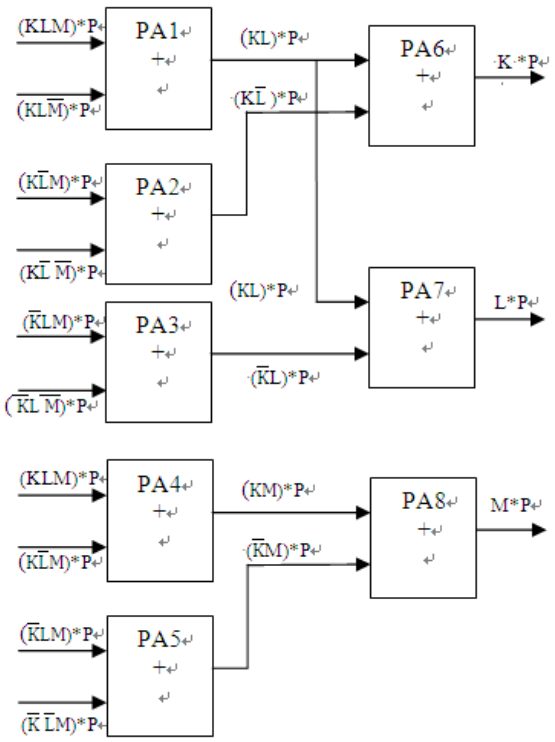

Fig. 5. Triple-scalar Point Multiplication, new method

In Fig. 6 K, L and M are stored in N bit Parallel registers and serve as input to the digital logic block. The digital logic block generates (K L M) *P, (K L $\overline{M}$) * P , (K $\overline{L}$ M) * P, (K $\overline{L}$ $\overline{M}$) * P ,($\overline{K}$ L M) * P, ($\overline{K}$ L $\overline{M}$) * P, and ($\overline{K}$ $\overline{L}$ M) * P. These outputs are given to PA1, PA2, PA3, PA4 and PA5 as shown in Fig.5. The Outputs K * P, L * P and M * P are taken from PA6, PA7 and PA8.Here, N and P are selected according to the NIST p-192 standard [4]. Algorithm 2 computes for the triple scalar point multiplication.

B. Calculation of saving in the number of Adders

Here, q is the probability that a bit in K, L or M is 1. Then the probability that a bit is 1 in the bitwise ANDed 3 variable minterm like, LKM is q*q*q = q^3. Therefore, the number of ones in the term KLM is q^3*N. Similarly for other terms, the number of 1's is as follows. the number of ones in the term KL$\overline{M}$ is,

$$q*q*(1–q)*N = q^2*(1–q)*N. \quad (33)$$

the number of ones in the term K$\overline{L}$ M is,

$$q*(1–q)*q*N = q^2*(1–q)*N. \quad (34)$$

the number of ones in the term $\overline{K}$L M is,

$$(1–q)*q*q*N= q^2*(1–q)*N. \quad (35)$$

the number of ones in the term K$\overline{L}$ $\overline{M}$ is,

$$q*(1–q)*(1–q)*N = q*(1–q)^2*N. \quad (36)$$

the number of ones in the term $\overline{K}$L$\overline{M}$ is,

$$(1–q)*q*(1–q)*N = q*(1–q)^2*N \quad (37)$$

the number of ones in the term $\overline{K}$ $\overline{L}$M is,

$$(1–q)*(1–q)*q*N = q*(1–q)^2*N. \quad (38)$$

Therefore, the total number ones in the 7 terms generated by the digital Logic Block of Fig. 6, is,

Total number of one's = $(q^3+3*q^2*(1–q) + 3*q*(1–q)^2)*N$

$$= (q^3–3*q^2+3*q)*N \quad (39)$$

Therefore the number of PA's inside the digital block is,

$$(q^3–3*q^2+3*q)*N. \quad (40)$$

Adding the additional external 8 PA's, the total number of PA's is,

$$n(PA,triplePM, new scheme) =(q^3–3*q^2+3*q)*N + 8 \quad (41)$$

From Eq. (23) and (41),

Saving in PA's = $3*q*N– ((q^3–3*q^2+3*q)*N + 8)$

$$= (3*q^2– q^3)*N – 8 \quad (42)$$

$$\% \text{ Saving} = \frac{(3*q^2 – q^3)*N–8}{1.5*N} * 100 \quad (43)$$

In general, for q = 0.5, the percentage saving is given by,

$$\% \text{ saving} = \frac{0.625*N–8}{1.5*N} * 100 \quad (44)$$

Neglecting 8, compared to 0.625*N,

$$\% \text{ saving} = \frac{0.625*N}{1.5*N} * 100 = 41.33\ \% \quad (45)$$

2. Algorithm for Triple scalar point multiplication

Inputs: Scalar K, L, M and Elliptic curve point P
Outputs: K * P , L * P and M * P

1. Using the digital logic block with inputs K, L and M, get (KLM), (KL$\overline{M}$), (K$\overline{L}$ M), (K$\overline{L}\overline{M}$), ($\overline{K}$L$\overline{M}$), ($\overline{K}$ LM) and ($\overline{K}$ $\overline{L}$ M)
2. Apply P and KLM to PM1 to get (KLM) *P
   Apply P and K L $\overline{M}$ to PM2 to get (KL$\overline{M}$)*P
   Apply P and K $\overline{L}$ M to PM3 to get (K$\overline{L}$ M) *P
   Apply P and K $\overline{L}$ $\overline{M}$ to PM4 to get (K$\overline{L}\overline{M}$) *P
   Apply P and $\overline{K}$L$\overline{M}$ to PM5 to get ($\overline{K}$L$\overline{M}$) *P
   Apply P and $\overline{K}$ L M to PM6 to get ($\overline{K}$ LM) *P
   Apply P and $\overline{K}$ $\overline{L}$ M to PM7 to get ($\overline{K}$ $\overline{L}$ M)*P
3. Add (KLM) *P and (KL $\overline{M}$) *P using PA1 to get KL* P
   Add (K$\overline{L}$M) *P and (K$\overline{L}$ $\overline{M}$) *P using PA2to get K$\overline{L}$* P
   Add ($\overline{K}$LM) *P and ($\overline{K}$L$\overline{M}$) *P using PA3 to get $\overline{K}$ L* P
   Add (KLM) *P and (K$\overline{L}$M) *P using PA4 to get KM* P
   Add ($\overline{K}$LM) *P and ($\overline{K}$ $\overline{L}$ $\overline{M}$) *P using PA5 to get $\overline{K}$ M* P
4. Add KL*P and K$\overline{L}$ *P using PA6 to get K * P.
   Add KL*P and $\overline{K}$ L *P using PA7 to get L * P.
   Add KM *P and $\overline{K}$ M *P using PA8 to get M * P

IV. Conclusions

A new hardware realization of Elliptic Curve Double-scalar Point Multiplication (DSPM) and Triple scalar point multiplication (TSPM) by reducing the Hamming weights is described. The module uses available Point Addition and Point Doubling Modules. The number of Point Additions is reduced because of the sum of products (SOP) expansion of the scalars. In case of double scalar multiplication, for 32 bit scalars, N=32 and the % saving of adders is 18.75 % than conventional multiplier. Similarly, our triple scalar multiplication, for 32 bits scalars, N=32 and the % saving of adders is 25% than conventional multiplier. Therefore our proposed methods are faster compared to the conventional methods even it uses more number of Point Multipliers. The principle described can be extended to more than 3 input multi-scalar multiplications.

In future, Fast Elliptic Curve Point Multiplication using Balanced Ternary Representation and Precomputation over GF(p) can be investigated.

Acknowledgment

The author would like to thank the Chairman Dr. R N Shetty, Director Dr. H N Shivashankar, Principal Dr. M K Venkatesha of RNS Institute of Technology for their constant support and encouragement and also to thank her professor N Bhaskara Rao, for his guidance and helpful comments in the study.

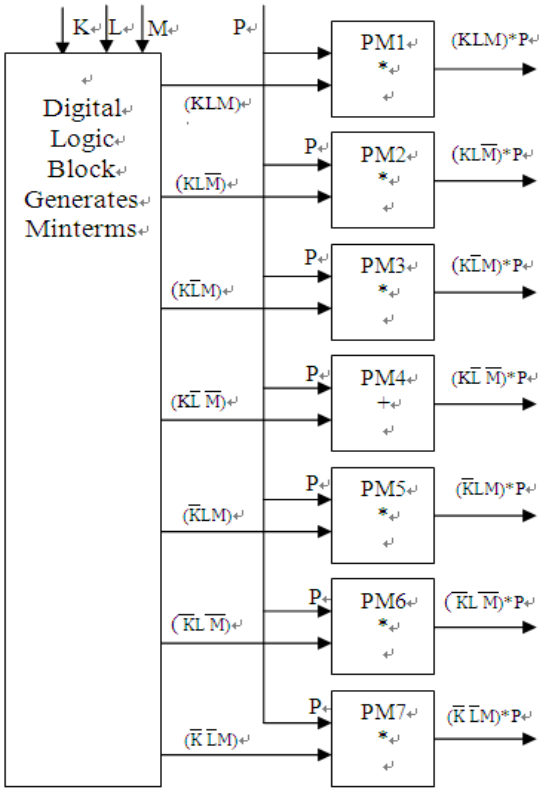

Fig.6. New Triple-scalar Point Multiplication Scheme

References

[1] N. Koblitz, "Elliptic curve cryptosystems", Mathematics of Computation, Vol.48, pp 203-209, 1987.
[2] V. Miller, "Uses of Elliptic Curve in Cryptography", Advances in Cryptology Crypto'85, LNCS, Vol. 218, pp. 417–426, 1986.
[3] IEEE 1363 standard specifications for public-key cryptography, 1363, Jan 2000.
[4] NIST, Recommended Elliptic Curves for Federal Government Use, May 1999 (http://csrc.nist.gov/encryption).
[5] K.Muthumayil, Dr.V.Rajamani, Dr.S.Manikandan and M.Buvana, "A Group Key Agreement Protocol based on stability and power using Elliptic curve cryptography" IEEE International Conference on Emerging Trends in Electrical and Computer Technology, pp.1051 - 1056, 2011.
[6] Raveen R. Goundar, Ken-ichi Shiota, and Masahiko Toyonaga, "A Novel Method for Elliptic Curve Multi-Scalar Multiplication", World Academy of Science, Engineering and Technology, Vol 33, pp 832-836, 2009.
[7] Darrel Hankerson, Alfred Menezes and Scott Vanstone, "Guide to Elliptic Curve Cryptography" Springer, 2004.

[8] Idrissi, Y.E.H.E., N. Zahid and M. Jedra, "Security analysis of 3GPP (LTE)-WLAN interworking and a new local authentication method based on EAP-AKA". IEEE International Conference on Future Generation Communication Technology, pp. 137-142, 2012.
[9] Shivkumar,S. and G.Umamaheswari, "Certificate authority schemes using elliptic curve cryptography, RSA and their variants-simulation using ns2". American Journal of Applied Sciences, vol 11, pp. 171-179, 2014.
[10] Jie. L.K. and H. Kamarulhaili, "Polynomial interpolation in the elliptic curve Cryptosystem", Journal of Math. Stat., vol 7, pp.326-331, 2011.
[11] Ismail, E.S. and M.S. Hijazi, 2012. Development of a new elliptic curve cryptosystem with factoring problem. American Journal of Applied Sciences, vol 9, pp.1443-1447, 2012.
[12] Ghosh S., M. Alam, I.S. Gupta and D.R. Chowdhury, "A Robust GF(p) parallel arithmetic unit for public key cryptography", IEEE 10th International conference on Euromicro Digital System Design Architectures, Methods and Tools, pp.109-115, 2007.
[13] G. Orlando, C. Paar, "A Scalable GF(p), Elliptic CurveProcessor Architecture for Programmable Hardware", in: Cryptographic Hardware and Embedded Systems (CHES), LNCS 2162, pp.348-363, 2001.
[14] De Dormole, G.M. and J.J. Quisquater, "High-speed hardware implementations of elliptic curve cryptography A Survey", Journal of System Architecture, vol 53, pp. 72-84, 2007.
[15] Sandeep S.V, HameemShanavas.I, Nallusamy.V, Brindha.M, "Hardware Implementation of Elliptic Curve Cryptography over Binary Field", IJCNIS Vol. 4, PP.1-7, 2012.

Authors' Profiles

Nagaraja Shylashree born in 1981, Coondapur. Received DE & CE in 1999, B.E. degree in Electronics and Communication from VTU, Karnataka, India in 2006. She received M.Tech. degree in VLSI Design and Embedded Systems from VTU, Karnataka, India in 2008. She is currently a research scholar(part-time) in PESCE, Mandya, Karnataka, India and is also an Assistant Professor in ECE Department, RNSIT, Karnataka, India. She has around 9 years of teaching experience. She has published twelve (12) research papers in various international journals and conferences. In addition, she is an author of Lab manuals for the branch of Electronics and Communication Engineering. She is member for IEEE and life member for ISTE and IETE.

Venugopalachar Sridhar born in 1958, Mysore. B.E degree in Electronics and Communication in PESCE, Mandya, Mysore and M.E in Jadavpur University, Calcutta.He has been awarded the Ph.D degree in IIT Delhi and also been awarded Post doc in Malaysia. He is currently The Principal of P.E.S. College of Engineering, Mandya, Karnataka, India. He was the Registrar (Evaluation) in VTU, India. He has around 29 years of teaching experience.
His research interests include Cryptography, VLSI and Embedded Systems, and Bio-Medical Engineering. He has published more than 50 research papers in various international journals and conferences. He is member for IEEE and life member for ISTE and IETE.

Manuscript received 14.03.2014; Revised 04.05.2014; accepted 06.06.2014.

How to cite this paper: Nagaraja Shylashree, Venugopalachar Sridhar,"Hardware Realization of Fast Multi-Scalar Elliptic Curve Point Multiplication by Reducing the Hamming Weights Over GF(p)", IJCNIS, vol.6, no.10, pp.57-63, 2014. DOI: 10.5815/ijcnis.2014.10.07

