Light Weight Asymmetric Cryptographic Algorithm for Financial Transactions through Mobile Application

Rina Maria
B.E. Student
Dept. of Electronics andCommunication
Dayananda Sagar College of Engineering
Bengaluru, Karnataka, India

V. Anitha, PhD
Professor
Dept. of Electronics and Communication
Dayananda Sagar College of Engineering
Bengaluru, Karnataka, India

ABSTRACT

Nowadays world is seen through mobile internet in a fraction of seconds, which invaded the need for many mobile applications. Financial transactions through mobile phones has become primitive mode of transactions. All these mobile applications including financial transactions demands for better security due to pervasive environment. In our work, we propose light weight Elliptical Curve Cryptographic method suitable for mobile applications. Elliptic curve point multiplication or scalar multiplication is the operation of adding a point P on the elliptic curve to itself successively scalar number of times. This paper describes an algorithm for light weight computation of scalar multiplication for elliptic curves defined over binary fields using projective coordinate system eliminating the need to perform inversions as needed with computations involving affine coordinates. It effectively incorporates fast computation method for binary elliptic curves by making use of Exclusive-OR gates. The effectiveness of the algorithm is measured both by Matlab simulation and Field-Programmable Gate Array (FPGA) implementation. Hardware implementation using Verilog proves that the proposed algorithm consumes less resources in terms of delay and power.

General Terms

Vedic Multiplier

Keywords

Vedic multiplier, Urdhva Tirgyagbyham sutra, ECC, conventional array multiplier, FPGA implementation, delay, low power, low area consumption.

1. INTRODUCTION

Smart phones form a major medium for data communication, also they provide pervasive platform for all financial transactions which demand high security and fast transaction. Standard encryption techniques are used for mobile applications. Since, mobile applications demand fast and secure computation of key generation, Elliptic Curve Cryptography (ECC) is an appropriate technique used widely for public key encryption technique based on properties and functions of elliptic curves. It is widely used in cryptography due to fast computation and bounds for unfeasible attack [1].

Scalar multiplication is an important operation in elliptic curve cryptography. The speed of the multiplier, delay and the area occupied are some of the important constraints to be considered. Several works and hardware designs have been proposed and implemented to enhance the speed of scalar multiplication computations. In [2], and efficient algorithm to compute scalar multiplications is presented. The proposed algorithm is suitable for both hardware and software implementation. It makes use of a new system of coordinates, projective coordinates. Affine coordinate system requires two field inversions which is relatively expensive operation for hardware implementations. Use of projective coordinate system overcomes this drawback as it requires only one inversion at the end of the scalar multiplication process to change back to affine coordinates. The operations are done using projective coordinates. In this paper, we compute ECC using projective coordinator method. In addition, Vedic Urdhva-Tiryagbyham method is used for fast multiplication in the computation of ECC. This paper compares various algorithms with the proposed algorithm and concludes that proposed algorithm using projective coordinates computes faster than other algorithms and requires no precomputation.

The rest of this paper is organized as follows Section 2 gives an overview of the work related with Elliptic Curve Cryptosystems. Vedic mathematics sutras and usage are described in Section 3. Next, Section 4 presents our proposed methodology with algorithm. In Section 5, simulation environment and results are discussed. Finally, the last section concludes the paper.

2. ELLIPTIC CURVE CRYPTO-SYSTEMS

The basic principle of ECC protocols are described in [3]. It is a study which defines elliptic curves defined in binary fields. The elliptic curves which are defined over binary fields are given by the equation,

$$E: y^2+xy=x^3+ax^2+b \qquad 2.1$$

Where in Equation 2.1 x,y € GF(2^m), a,b are real numbers, a≠0 and b≠0.

Addition, multiplication, inversion and exponentiation computations in Galois field are described in the paper. It shows that elliptic crypto-schemes offer higher security per bit ratio compared to any other currently known public-key cryptosystems [3]. The survey of different techniques for implementing ECC at high speed are described in [4]. It gives criteria to choose a technique over the other and also gives efficiency-flexibility tradeoff. It suggests the use of projective coordinates to avoid costly inversions during implementation. Several principles to be considered by the designer are the choice of coordinates, multiplier architecture, and the speed of the architecture. It also describe about multi-scalar multiplications algorithms can also be designed using the principles and taking advantage of the full hardware available using small operands. New doubling method which is simpler to implement is proposed in [5] which explains about a new kind of projective coordinates. The new projective doubling algorithm requires three general field of multiplications, two multiplications by a fixed constant, and five squaring. It takes one less field multiplication than the previous doubling formula, giving a good improvement. In [5], the number of multiplications in different kind of projective coordinates can be seen and the methods work well even for elliptic curves chosen at random.

3. VEDIC MATHEMATICS

Enormous amount of research is being done in Vedic (ancient) mathematics which comprises of 16 sutras (formulae) and 13 sub-sutras (sub-formulae) for different operations. Urdhva Tiryagbyham is the sutra used for general multiplication. The main advantage of this sutra is that it reduces multi-bit multiplications down to one-bit. In [6], a low power Vedic multiplier design is proposed. It is implemented using VHDL, using Modelsim and ISE for simulation. A comparison of various multipliers like Array multiplier, Booth multiplier with Vedic multiplier concludes that Vedic multiplier has lesser delay and power dissipation than other multipliers. 32x32 bit Vedic multiplier is implemented on Spartan XC3S500-5-FG320 in [7]. It makes use of 16x16 multiplier blocks in an optimized way to give better delay, power and hardware requirements than conventional multipliers. The high speed multiplier proposed in this paper exhibits improved speed. In [8], 8x8 Vedic multiplier is implemented using 4x4 multiplier which in turn is implemented using 2x2 multiplier blocks. It is implemented using Verilog HDL on Xilinx FPGA Spartan 3 board. It compares the slices of FPGA occupied by 2x2, 4x4 and 8x8 Vedic multiplier and it can be concluded that these multipliers are more efficient than Array and Booth multipliers. Vedic multipliers are used in low power techniques [9] and high speed computations [10]. In [11] FPGA implementation of Urdhva-Tiryagbyham based multiplier is implemented using VHDL. Efficient multiplication is achieved using Urdhva-Tiryagbyham method and its importance is described in with design and implementation in papers [12-15].

4. PROPOSED METHODOLOGY

We in our proposed work, mainly focus on projective coordinator method. Projective coordinator method is helpful in easing the computation of inversion as field inversion in binary fields is an expensive process. Projective coordinates can be used to represent point O (point at infinity) by simply setting the third coordinate (Z) as 0 instead of division by 0 to indicate infinity (as in affine coordinates). Use of these coordinates makes representation of points easier, for example, a point x_1 could be represented as X_1/Z_1 in projective coordinate system or simply, (X_1, Z1). Computations are easier with these coordinates as it does not involve division.

In [2], the Montgomery algorithm using projective coordinates as modified is given below.

Consider $P_1 = (X_1, Y_1, Z_1)$, $P_2 = (X_2, Y_2, Z_2)$ and $Q = (X_3, Y_3, Z_3)$ where P_1 and P_2 are two points on the elliptic curve in terms projective coordinates X,Y and Z, and Q is the resulting point of scalar multiplication(affine coordinates) lying on the elliptic curve considered.

Input : k>0, P

Output: Q=kP

```
1.  Set $k=(k_{l-1},\dots k_2,k_1,k_0)_2$
2.  Set $X_1=x$, $Z_1=1$, $X_2=x^4+b$, $Z_2=x^2$
3.  For i from (l-2) to 0
        If $k_i=1$,
            $Madd(X_1,Z_1,X_2,Z_2)$, $Mdouble(X_2,Z_2)$
        else
            $Madd(X_2,Z_2,X_1,Z_1)$, $Mdouble(X_1,Z_1)$
4.  Return $Q=Mxy(X_1,Y_1,X_2,Y_2)$.
```

Madd and Mdouble is computed as,

If $P_1=P_2$, $X_3= X_1^4 + bZ_1^4$

$Z_3= Z_1^2X_1^2$

If $P_1\neq P_2$, $Z_3= (X_1Z_2 + X_2Z_1)^2$

$X_3= xZ_3 + (X_1Z_2) (X_2Z_1)$

According to this algorithm, points P_1 and P_2 are pre-computed using the affine coordinates of a point P and 2P(obtained by point doubling of P). Based on whether the i^th bit of k is 0 or 1, point addition(Madd) and point doubling(Mdouble) is computed. This continues till i becomes 0, as a result of which point P_1 obtained in the process is the output, converted back to affine coordinates, resulting in Q. For conversion of point Q back to affine coordinates, Mxy is computed using the following equations.

$x_3= X_1/Z_1$

$y_3= (x + X_1/Z_1) \{ (X_1 + xZ_1) (X_2 + xZ_2) + (x^2 + y) (Z_1Z_2) \} (xZ_1Z_2)^{-1} + y$

The field multiplications have increased than those using affine coordinates, but the field inversion is reduced to only one in the final conversion back to affine coordinates. The flow chart of the whole process of scalar multiplication is as shown below.

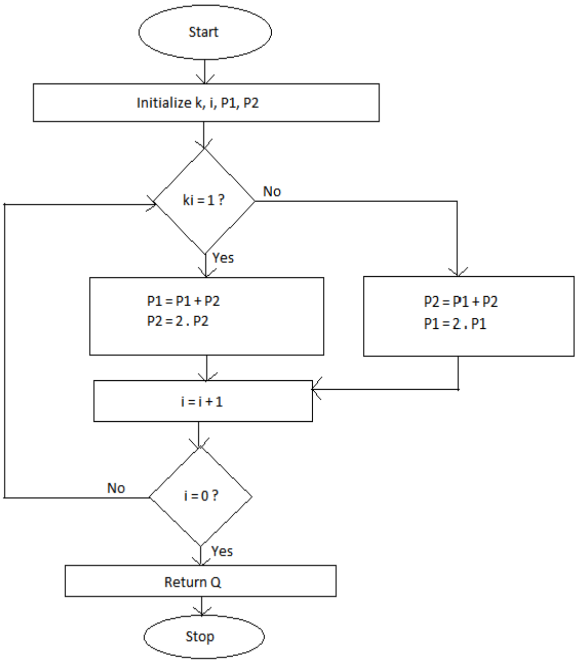


Figure 1: Scalar Multiplication

In this paper, field multiplications are computed using Vedic mathematics sutras. The main advantage of this sutra is that it reduces multi-bit multiplications down to one-bit.

As mentioned earlier, this paper uses Urdhva Tiryagbyham, a Vedic mathematics sutra for reducing the complexity of multiplications. Urdhva Tiryagbyham method means ―vertically and crosswise. Vertically means straight above multiplication and crosswise means diagonal multiplication and taking their sum. This sutra can be implemented starting from right hand side (LSB) or from left hand side (MSB). One-bit multiplication results in partial products in each step, which are modulo-2 added i.e., XOR operation is performed on the partial products and one-bit result is obtained. The procedure is illustrated as below.The multiplication of a_3 a_2 a_1 a_0 and b_3 b_2 b_1 b_0 is shown as follows.

Step 1: a_3 a_2 a_1 a_0 ↑ $r_7 = a_0 b_0$

b_3 b_2 b_1 b_0

Step 2: a_3 a_2 a_1 a_0 ╳ $r_6 = a_1 b_0 + a_0 b_1$

b_3 b_2 b_1 b_0

Step 3: a_3 a_2 a_1 a_0 $r_5 = a_0 b_2 + a_1 b_1 + a_2 b_0$

b_3 b_2 b_1 b_0

Step 4: a_3 a_2 a_1 a_0 $r_4 = a_0 b_3 + a_1 b_2 + a_2 b_1 + a_3 b_0$

b_3 b_2 b_1 b_0

Step 5: a_3 a_2 a_1 a_0 $r_3 = a_1 b_3 + a_2 b_2 + a_3 b_1$

b_3 b_2 b_1 b_0

Step 6: a_3 a_2 a_1 a_0 ╳ $r_2 = a_2 b_3 + a_3 b_2$

b_3 b_2 b_1 b_0

Step 7: a_3 a_2 a_1 a_0 ↑ $r_1 = a_3 b_3$

b_3 b_2 b_1 b_0

Result obtained from this method is r_1 r_2 r_3 r_4 r_5 r_6 r_7. Further masking with the irreducible polynomial of the field gives the final result of multiplication.

5. SIMULATION RESULTS AND ANALYSIS

Simulation Environment: The method to evaluate the efficiency of the algorithm is achieved in two methods. In first method we simulated the output using projective coordinate using Matlab. Later in the second method, the implementation was carried out using Xilinx ISE and Xilinx ISim on a Xilinx Virtex 5 FPGA using Verilog and netlists is generated. FPGA Specifications as given in Table1. XPower Analyzer is a tool used for power analysis post-implemented place and routed designs. The proposed method is used to implement 4x4 and 8×8 Vedic multiplier, coded in Verilog and simulated using Xilinx I Simsimulator and synthesized using Xilinx XST.

Table1: FPGA Specifications

Family	Virtex5
Device	XC5VLX110T
Package	FF323
Speed	-2

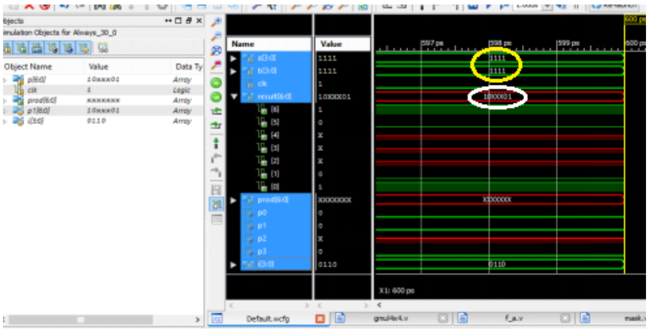


Figure 2: 4x4 Vedic Multiplier

Figure2 shows the intermediate result of multiplication. The numbers circled in yellow are the inputs, intermediate result being highlighted in white. The final result is highlighted in redin Figure 3.

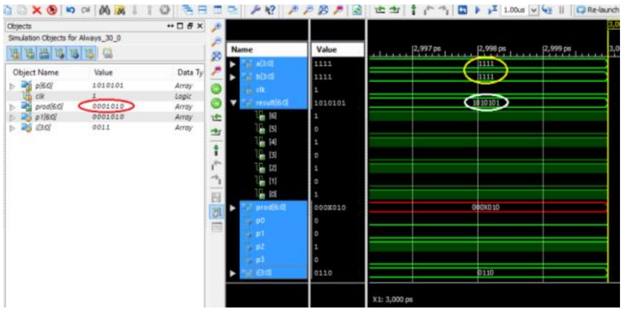


Figure 3: 4x4 vedic multiplier

Figures 2 and 3 show the results for 4x4 vedic multiplier. As we can see in Figure 3, the result of multiplication is 1010101 but it is masked by the field polynomial to obtain the final result of 15x15 as 10.

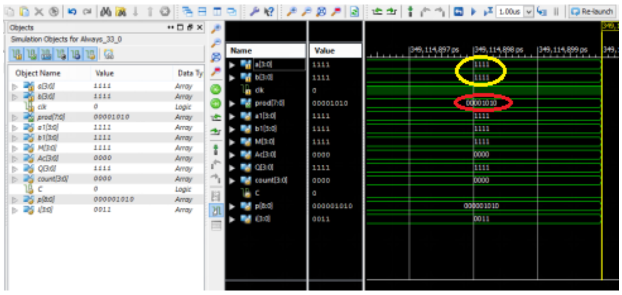
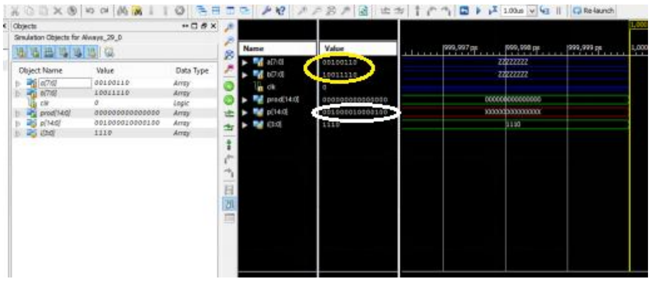


Figure 4: 4x4 array multiplier using shift-and-add algorithm

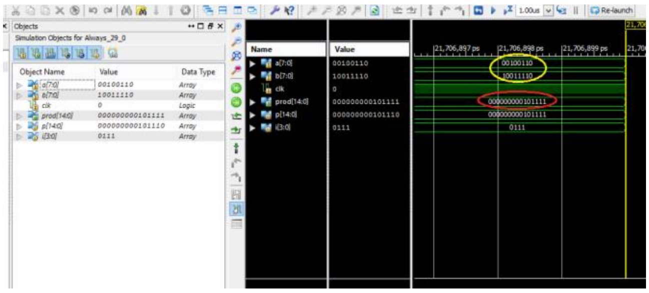
Figure 4 shows the simulation results of an array multiplier using shift and add algorithm. The final result 10 obtained after masking the result of multiplication with the field polynomial is highlighted in red. Multiplication of two 8-bit inputs a and b results in a 15-bit output which is masked with the irreducible polynomial of the given field (here, GF(2^8) considered and the irreducible polynomial chosen is

$x^8+x^4+x^3+x+1$). 8x8 Vedic multiplier is simulated and results are shown in Figure 5 and 6.



Object Name	Value	Data Type
a[7:0]	00100110	Array
b[7:0]	10011110	Array
clk	0	Logic
prod[14:0]	000000000000000	Array
p[14:0]	001000010000100	Array
i[3:0]	1110	Array

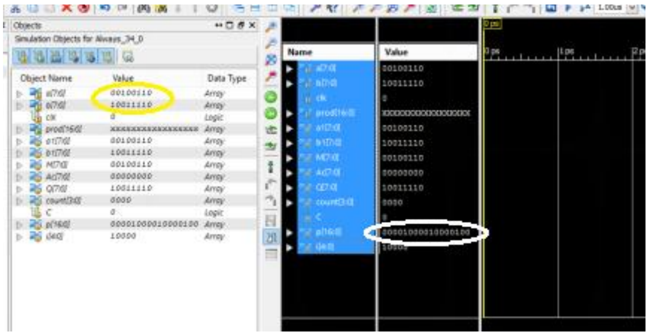
Figure 5: 8x8 Vedic multiplier without Masking



Object Name	Value	Data Type
a[7:0]	00100110	Array
b[7:0]	10011110	Array
clk	0	Logic
prod[14:0]	000000000101111	Array
p[14:0]	000000000101110	Array
i[3:0]	0111	Array

Figure 6: 8×8 Vedic Multiplier with Masking

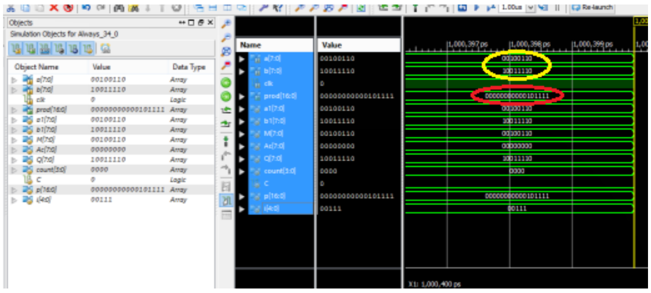
Figure 5 shows the result of multiplication of 38 and 158 as 6004(without masking) highlighted in white. Figure 6 shows the final result(highlighted in red) of 38x158 as 47(after masking).8x8 multiplier implemented using shift and add algorithm is simulated and results are as shown in Figure 7 and 8.



Object Name	Value	Data Type
a[7:0]	00100110	Array
b[7:0]	10011110	Array
clk	0	Logic
prod[16:0]	xxxxxxxxxxxxxxxxx	Array
a1[7:0]	00100110	Array
b1[7:0]	10011110	Array
M[7:0]	00100110	Array
Ac[7:0]	00000000	Array
Q[7:0]	10011110	Array
count[3:0]	0000	Array
C	0	Logic
p[16:0]	00001000010000100	Array
i[4:0]	10000	Array

Figure 7: 8x8 array multiplier using shift and add algorithm

Figure 7 shows the result of 38 and 158 as 6004 (without masking) highlighted in white. By masking with the field polynomial, the final result is as shown in Figure8.



Object Name	Value	Data Type
a[7:0]	00100110	Array
b[7:0]	10011110	Array
clk	0	Logic
prod[16:0]	00000000000101111	Array
a1[7:0]	00100110	Array
b1[7:0]	10011110	Array
M[7:0]	00100110	Array
Ac[7:0]	00000000	Array
Q[7:0]	10011110	Array
count[3:0]	0000	Array
C	0	Logic
p[16:0]	00000000000101111	Array
i[4:0]	00111	Array

Figure 8: 8x8 array multiplier using shift and add algorithm

In Figure 8, 8x8 array multiplier using shift and add algorithm is depicted. Final result (after masking) of 38x158 as 47 highlighted in red. In addition comparison of different parameters for ECC using conventional array multiplier and proposed Urdhva Tiryagbyham multiplier method is shown in Table2. From the results it is clearly shown that among both multipliers in terms of FPGA resources used (Slices), Urdhva Tiryagbyham multiplier utilizes lesser resources for the same n-bit multiplication than Array multipliers. The main advantage of this sutra is that it reduces multi-bit multiplications down to one-bit. Therefore, it can be effectively used in hardware and/or software implementation of elliptic curve cryptography defined over GF (2^m), along with the use of projective coordinate system.

Table 2: Comparison of conventional array multiplier and proposed Urdhva Tiryagbyham multiplier

Parameters	Conventional Array multiplier		Proposed Urdhva Tiryagbyhamvedic multiplier	
	4x4	8x8	4x4	8x8
No. of Slice registers used	8	16	8	16
Total available	69,120	69,120	69,120	69,120
Utilization (%)	1	1	1	1
No. of Slice LUTs used	8	40	8	35
Total available	69,120	69,120	69,120	69,120
Utilization (%)	1	1	1	1
No. of occupied Slices	4	16	3	12
Total available	17,280	17,280	17,280	17,280
Utilization (%)	1	1	1	1
Maximum frequency(MHz)	733.568	733.568	733.568	733.568
Delay (ns)	0.682	0.682	0.682	0.682
Worst case slack (ns)	1.521	1.530	1.28	1.572
Power Consumption Signal Power in Watts	0.00117	0.00121	0.00082	0.00104

6. CONCLUSION

Nowadays, mobile phone usage is part of all financial communications which demands fast and secure transactions. The proposed light weight asymmetric algorithm is promising in two aspects. Firstly, it utilizes less slices which reduces cost, delay and complexity. Secondly, it reduces the power consumption by 3.5% where it has a greater impact on temperature and longevity of the mobile phones. Thus our proposed light weight cryptographic algorithm can be applied for all mobile transactions that demands higher speed and more security. This work can be further extended by increasing the number of input bits with other Vedic sutras to reduce the delay in computation.

7. REFERENCES

[1] N. Koblitz, “Elliptic curve cryptosystems”, Mathematics of computation, 1987.

[2] J López, R Dahab, “Fast multiplication on elliptic curves over GF (2^m) without precomputation”, *Cryptographic Hardware and Embedded Systems*, 1999.

[3] K Rabah, “Elliptic curve cryptography over binary finite field GF (2^m)”, *Information Technology Journal*, 2006.

[4] GM de Dormale, JJ Quisquater, “High-speed hardware implementations of elliptic curve cryptography: A survey”, *Journal of systems architecture*, 2007.

[5] J López, R Dahab, “Improved algorithms for elliptic curve arithmetic in GF (2^n)”, *International Workshop on Selected Areas in Cryptography*, 1998.

[6] Hankerson D., López Hernandez J., Menezes A., “Software Implementation of Elliptic Curve Cryptography over Binary Fields”, *International Workshop on Cryptographic Hardware and Embedded Systems*, 2000.

[7] K Okeya, K Sakura, “Efficient elliptic curve cryptosystems from a scalar multiplication algorithm with recovery of the y-coordinate on a Montgomery-form elliptic curve”, *Cryptographic Hardware and Embedded Systems*, 2001.

[8] A Kanhe, SK Das, AK Singh, “Design and implementation of low power multiplier using vedic multiplication technique”, *International Journal of Computer Science and Communication*, January-June 2012.

[9] GG Kumar, V Charishma, “Design of high speed vedic multiplier using vedic mathematics techniques”, *International Journal of Scientific and Research Publications*, March 2012.

[10] M Poornima, SK Patil, SKP Shivukumar, Shridhar K P, Sanjay H, “Implementation of multiplier using vedic algorithm”, International Journal of Innovative Technology and Exploring Engineering, May 2013.

[11] Siba Pradhan, Ritsinghda Das, ”VLSI Implementation of Vedic Multiplier Using Urdhva-Tiryakbhaym Sutra in VHDL Environment:A Novelty”, *IOSR Journal of VLSI and Signal Processing(IOSR-JVSP)* Volume 5, Issue 1,ver.III(Jan-Feb-2015).

[12] Vishikha Sharma, Aniket Kumar, “Design, Implementation & Performance of Vedic Multiplier for Different Bit Lengths”, *International Journal of Innovative Research in Computer and Communication Engineering*, Vol. 5, Issue 4, April 2017.

[13] Kedar N. Palata,Vinobha K. Nadar,Jatin S. Jethawa,Tushar J. Surwadkar,Rajan S. Deshmukh, “Implementation of an Efficient Multiplier based on Vedic Mathematics”, *International Research Journal of Engineering and Technology (IRJET)*,Volume: 04, Issue: 04, Apr -2017.

[14] P Gulati, H Yadav, MK Taleja, “Implementation of an efficient multiplier using the Vedic multiplication algorithm”,*International Conference on Computing, Communication and Automation (ICCCA2016), April 2016.*

[15] Prof. Mrs. Y.D. Kapse, Miss. Pooja R. Sarangpure, Miss. Komal M. Lokhande, “Review on a Compressor Design and Implementation of Multiplier using Vedic Mathematics”, *International Journal of Advanced Research in Computer and Communication Engineering*, Vol. 6, Issue 2, February 2017.