ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

Modular Multiplication Algorithm in Cryptographic Processor: A Review and Future Directions

Poomagal C. T
Research Scholar, Department of Electronics and Communication Engineering,
Sri Venkateswara College of Engineering, Sriperumbudur, India
Email: ctpoomagall@gmail.com

Dr. Sathish Kumar G. A
Professor, Department of Electronics and communication Engineering,
Sri Venkateswara College of Engineering, Sriperumbudur, India
Email: sathish@svce.ac.in

Abstract: *The strength of public-key cryptography depends on the degree of difficulty of a private key to be determined from its corresponding public key. Such a Key can be generated from computationally high radix arithmetic operations like modular exponentiation with very large integer values. To perform this modular exponentiation, different multiplication algorithms like Montgomery algorithm, Karatsuba algorithm, VLM3 algorithm are used. In this Paper, those algorithms results are analyzed and their performance towards measure of algorithm complexity, critical path delay, area, time period and frequency are compared.*

Keyword: *High-speed and high-radix arithmetic; Modular exponentiation and multiplication; Montgomery algorithm; Karatsuba algorithm; VLM3 algorithm.*

1. INTRODUCTION

Cryptography has a lot of algorithms, which are hard to break and are having a great deal of computational effort. To achieve cryptography, encryption is the most effective way. So that to access the encrypted file and to decrypt the file, there must be the secret key. Based on the type of key, Encryption is segregated into symmetric encryption and asymmetric encryption or public-key encryption. Because of the computational complexity of asymmetric encryption, it is usually used only for less sized blocks of data, typically the transfer of a symmetric encryption key. This symmetric key is then used to encrypt the original message sequence to transmit. The symmetric encryption and decryption is based on simpler algorithms and is much faster.

2. MODULAR EXPONENTIATION

To perform the public key encryption, modular exponentiation is an important candidate used in the key generation and exchange process. The process of modular exponentiation renders a result, which is the remainder a division to a base raised to a power, by a positive integer. In general, modular exponentiation can be given by the value 'c' in Equation (1)

$$c = b^e \pmod m \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Raising b to the power e, and then reduce modulo m is a very unsophisticated and slow method because raising b to the power e can result in a really huge number that takes long computations for any high radix number and so this retards the speed of the algorithm .

Algorithm 1 Repeated classical modular multiplication Algorithm [4].

$Step1: INPUT \rightarrow a, b, m.$
$Step2: OUTPUT \rightarrow P$
$Step3: P = a.b$
$Step4: m \ll N$
$Step5: for\ i = N\ down\ to\ 0\ do$
$if (P \geq m) P = P - m$
$m = m \gg 1$
$endfor$
$Step6: return\ P$

To keep the intermediate result (mod m), which end up in long computations at each step, modular multiplication can be used, so that no need to ever hold numbers larger than the modulo. The repeated implementation of a classical modular multiplication algorithm in [4] will give efficient results in computing

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

modular exponents. The classical modular multiplication is given as Equation (2):

$$P = a.b\ (mod\ m) \ldots\ldots\ldots\ldots\ldots\ldots (2)$$

3. MULTIPLICATION ALGORITHM

3.1 Montgomery Multiplication

Montgomery multiplication is one of the methods for fast modular multiplication termed as Montgomery modular multiplication, by the American mathematician Peter L. Montgomery in 1985[5].

Algorithm 2 Combined Loop Operation of Montgomery Multiplication [MM] Algorithm [2].

$Step1: INPUT \rightarrow (k = (k_{t-1}, \ldots . k_1, k_0)_2\ with\ k_{t-1} = 1,$
$P = (x, y) \epsilon E(F_{2^m}).$
$Step2: OUTPUT \rightarrow k.P.$
$Step3: For\ i = t - 2\ to\ 0\ with\ k_i = 1$
$Point\ adition: P(X_1, Z_1) = P(X_1, Z_1) + Q(X_2, Z_2).$
$Point\ Doubling: Q(X_2, Z_2) = 2Q(X_2, Z_2).$
$Step4: If\ k_{i-1} = 1, then,$
$1: Z_1 = X_2.Z_1,$
$2: X_1 = X_1.Z_2, Z_2 = Z_2^2, T = Z_2^4,$
$3: X_2 = b.T + X_2^4, Z_2 = X_2^2,$
$4: Z_2 = X_2.Z_2,$
$5: X_1 = X_1.Z_1, T = (X_1 + Z_1)^2, Z_1 = T.Z,$
$6: X_1 = x.T + X_1.$
$Step5: Else\ if\ k_{i-1} = 0, then,$
$1: Z_2 = X_1.Z_2,$
$2: X_2 = X_2.Z_1,$
$3: X_2 = b.T + X_2^4, X_2 = X_2^2,$
$4: Z_2 = X_2.Z_2,$
$5: X_1 = X_1.Z_1, T = (X_1 + Z_1)^2, Z_1 = T.Z,$
$6: X_1 = x.T + X_1.$
$Step6: Conversion. X_3 = \frac{X_1}{Z_1};\ Y_3 = \left[\frac{x + X_1}{Z_1}\right][(X_1 + x.Z_1)(X_2 + x.Z_2) + (x^2 + y)(Z_1.Z_2)][xZ_1Z_2]^{-1} + Y.$

This algorithm has a critical path delay of $T_A + \left[\log_2\left(\frac{d}{n}\right)\right]T_X$ or $T_{mux} + \log_2(n + k)\, T_X$.

Montgomery multiplication can be implemented by transforming the elements to be multiplied from one system to another and then the multiplication operation is carried out. In Elliptic Curve

Cryptography (ECC), the adequately used arithmetic operations are point addition and point doubling, which can be carried out by Montgomery multiplication method. It can be computed in either affine coordinates or projective coordinates. Finally, transforming the resulting coordinates output to the respective coordinates gives the classical modular product. Converting into Montgomery form, and computing a product using Montgomery multiplication is not much fast as computing the product in the integers and performing a modular reduction by division or Barrett reduction[6], [7].

Comparatively, when many products are required, the conversion to Montgomery form is an insignificant fraction of the computation, so the computation time is limited than the other methods.

The increased speed provided by Montgomery multiplication is essential to the cryptosystems like RSA, Diffie–Hellman key exchange and ECC, which are based on arithmetic operations with a large number [8]-[12]. By the usage of Montgomery form for multiplications, modular products can be calculated with less-expensive divisions. The proposed algorithm in [2] relies on high-performance design space due to its speed, sustainability to side channel attack and low resource usage. Montgomery Algorithm used in [2] is depicted for the projective coordinates and then converts back to affine coordinates as:

3.2 Karatsuba Algorithm

In 1960, Karatsuba algorithm is discovered by Anatoly Karatsuba and published in 1962[13], which is a fast multiplication algorithm and reduce the order of multiplications to at most $n^{\log_2 3} \approx n^{1.585}$ single-digit multiplications. The direct multiplication of m-bit multiplier is expensive, when m is large, for devices like FPGA. So Karatsuba multiplication algorithm can be used as the good solution and sub-quadratic complexity.

The basic step of Karatsuba's algorithm is the computing three different multiplications of smaller numbers, instead of calculating a product of two large numbers, plus some additions and digit shifts. The corresponding polynomial multiplications as follows in equation (3):

$$P = a.b = (2^l a_1 + a_0).(2^l b_1 + b_0) = \left(2^{2l}(a_1 b_1)\right) + 2^l(a_1 b_0 + a_0 b_1) + a_0 b_0 = 2^{2l} p2 + p1 + p0 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

Let a and b be two m-bit numbers and letl = ⌈m/2⌉. Initially, a and b are split into two equal-sized parts as a_0 and $a_0 b_0$ which are the l least significant bits and a_1 and b_1 are the most significant bits. The

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

value 2l denotes the base β of this system. Finally, multiplications of the half sized numbers a_1, a_0, b_1, and b_0 will resulting in the multiplication of the entire value of $a = 2^l(a_1 + a_0)$and $b= 2^l(b_1 + b_0)$.

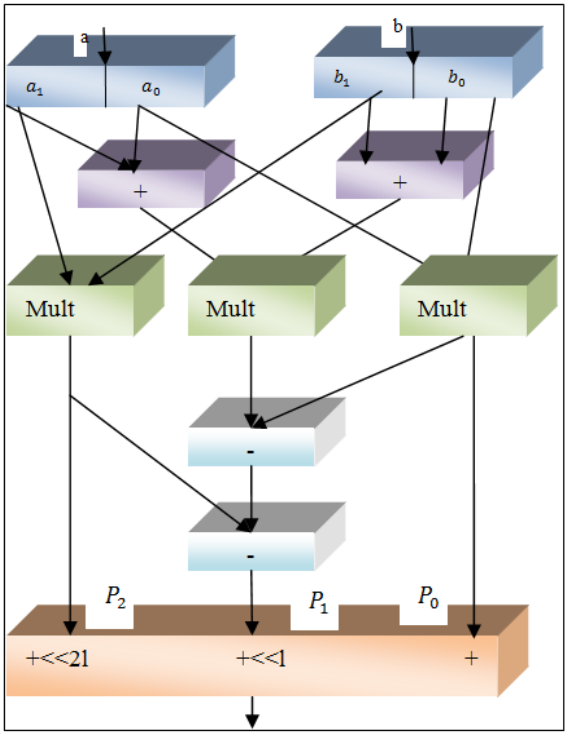

Figure1 Karatsuba Ofman Multiplication [KOM]

In P_1, the two multiplications and one addition are replaced by one multiplication, two additions, and two subtractions as $P_1 = (a_1 + a_0).(b_1 + b_0) - a_0b_0 - a_1b_1$, thus attain a global complexity of the order of $N\log_2(3)$. Similarly, the three sub-multipliers a_1b_1, a_0b_0 and $(a_1 + a_0).(b_1 + b_0)$are using the Karatsuba algorithm recursively. Pictorial representation of the process in Karatsuba ofman Multiplication [KOM] algorithm is depicted in figure1. Karatsuba's algorithm is asymptotically more efficient than the classical one, Other than for small-sized numbers. But the delay of this algorithm increases monotonically with iteration steps and the optimum area consumption is arrived at a threshold size in split operation. The modular multiplication used in the finite field elliptic curve can be formulated using Karatsuba algorithm, which causes a total delay of D_{tot} in equation (4):

$$D_{tot} = D_{MAC} + D_{add} + D_{MUX} \quad \ldots\ldots\ldots\ldots\ldots\ldots (4)$$

Where, D_{add} is extra addition's delay and $D_{MUX} = \log_k(2^s + s)$ for the multiplexer with 's' selection lines and k input bit. The delay generated by MAC in [3] is also calculated by the Karatsuba multiplication algorithm by four stages viz splitting stage(D_{sp}), classical multiplication stage(D_{CM}), aligning stage(D_{al}) and merged reduction & addition stage(D_{mod}). The operands $(a_1 + a_0).(b_1 + b_0)$ require r-step splitting of 2^r input bits. Therefore, $(D_{sp})=\log_k 2^r$for k-inputs and $(D_{CM}) = \log_k 2\tau$ for threshold size of τ. The aligning stage has r- steps, so $(D_{al}) = r$ and $(D_{MOD}) = \log_k t$ for t variables in longest chain. Hence,

$$D_{MAC} = D_{sp} + D_{CM} + D_{al} + D_{MOD}$$

3.3 VLM3 (Variable length Montgomery modular multiplication) Algorithm

One more new algorithm for the modular multiplication proposed in [1] referred as VLM3 algorithm implements the modular multiplication operation by final addition and comparison block along with MBS, CSA registers, shift registers, MUX, XOR and NAND gates.

Algorithm 2 VLM3 Algorithm. [1]

$Step1: INPUT \rightarrow X_{SD}, Y, M;$
$Step2: OUTPUT \rightarrow S = X.Y.2^{-n} mod\ M$
$Step3: S_C(0) = 0, S_S(0) = 0;$
$Step4: For\ i = 0\ to\ J$
$Step5: a^{(i)} = f^{(i)}$
$Step6: If\ f^{(i)} = 3, then [P_C(i), P_S(i)]$
$= S_C(i) + S_S(i),$
$a^{(i)} = a^{(i)} - 1;$
$Else\ If\ k_i = 0, then [P_C(i), P_S(i)]$
$= S_C(i) + S_S(i) + 2^{a^{(i)}}.Y;$
$Else\ [P_C(i), P_S(i)] = S_C(i) + S_S(i) - 2^{a^{(i)}}.Y;$
$Step7: q_i = [P_C(i), P_S(i)]_{k\ldots0}\left(2^{a^{(i)+1}}\right.$
$\left. - M^{-1}_{k\ldots0}\right) mod\ 2^{a^{(i)+1}}$
$Step8: [S_C(i+1) + S_S(i+1)]$
$= [P_C(i), P_S(i) + q^{(i)}M] / 2^{a^{(i)+1}}$
$Step9: End\ for;$
$Step10: S(n) = [S_C(J+1) + S_S(J+1)]$
$Step11: S(n) \geq M, then\ S(n) = s(n) - M$
$Step12: Return\ S(n).$

In the normal Montgomery multiplication algorithm, each digit of multiplier and multiplicand are represented in radix-powers of 2, hence the proposed method in [1] relies on high radix partial multiplication to binary modular multiplication using some newly proposed expansion techniques to the multiplier.

In this algorithm, X_{SD} is the expansion of the multi-

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

plier X using canonical recoding as done in [1], Y is multiplicand and M is Modulus. Using the expansion proposed in [1], the carry and sum component of the product P is denoted as

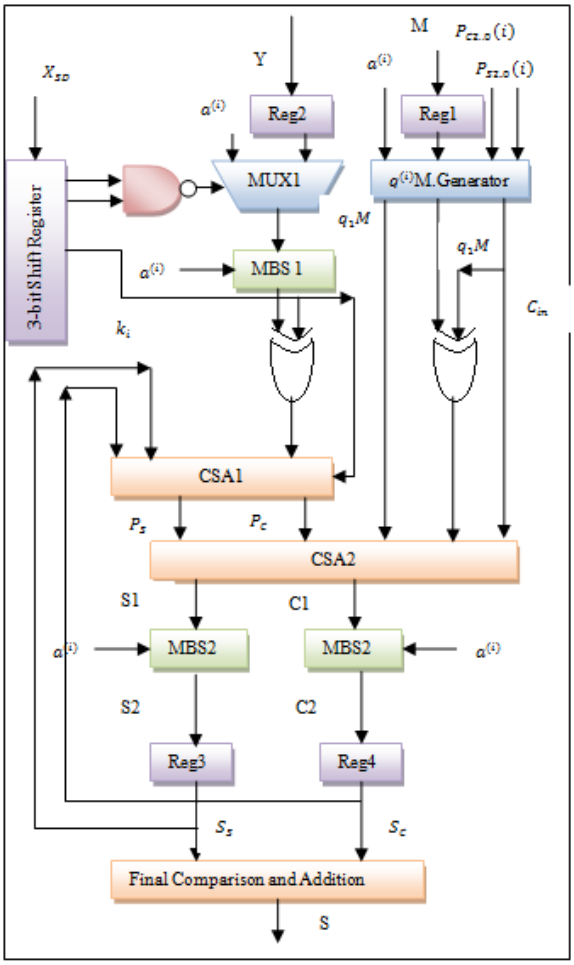

Figure 2 VLM3 Algorithm [1]

$$[P_C(i), P_s(i)] = S_C(i) + S_s(i) + 2^{a^{(i)}}.Y$$
$$[P_C(i), P_s(i)] = S_C(i) + S_s(i) - 2^{a^{(i)}}.Y$$

In this circuit, the selection input of Multiplexer 1 is based on the input $f^{(i)}$, (i,e) if sel=0 then mux output is 0 for MBS otherwise Y to the MBS1 input. Another signal K_i is used to determine the operation of CSA. Finally, CSA1 generates the output at each clock cycle as $S_C(i) + S_s(i) + X(i).Y$. Similarly CSA2 has $[P_C(i), P_s(i) + q^{(i)}M$. And the value a(i) to MBS2 shifts the inputs of MBS2. The $q^{(i)}M$ generator module in the VLM3 algorithm data path contains two MBS, an LUT and a full adder to generate the modulus value, which is shown below in figure2.

3.4 COMPARISON OF DISCUSSED ALGORITHMS

The comparison table of parameters such as critical path delay, area consumed, time period and frequency for the above discussed-algorithm is depicted below.

TABLE 1 COMPARISON OF THE ALGORITHM IN [2], [3] AND [1].

Parameters	Montgomery Algorithm[2]	Karatsuba algorithm[3]	VLM3 Algorithm[1]
Critical Path Delay	$T_{mux} + \log_2(n + k)\, T_x$	$D_{tot} = D_{MAC} + D_{add} + D_{MUX}$	$3T_{FA} + T_{XOR}$
Area (Slices)	1089	3041	6105
Frequency (MHz)	296	294	400
Time Period (ns)	3.37	3.4	2.5

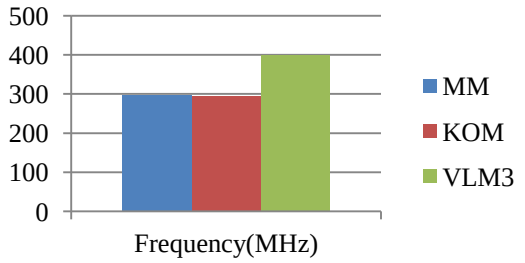

Figure 3 Frequency Comparison Chart

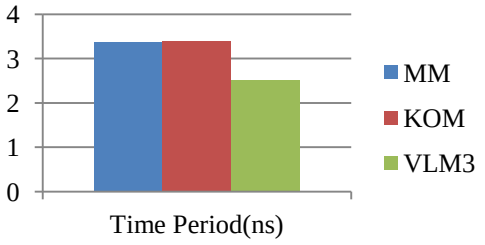

Figure 4 Comparison of Time period in [1], [2] and [3]

In the aspects of time and frequency, VLM3 algorithm shows the better results of more frequency and lowest time period among the three algorithms compared in this paper, which can be observed from the Figure 3 and Figure 4. In terms of area, VLM3 consumes comparatively more area than Karatsuba ofman Multiplication [KOM] algorithm and Montgomery

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

multiplication [MM] algorithm, which can be clearly observed from the below chart for Area plot, Figure 5.

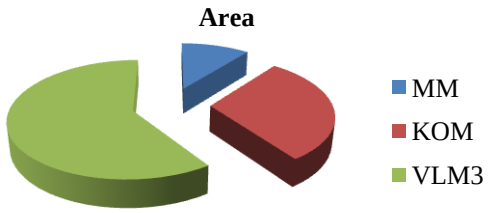

Figure 5 Area Plot for the three different algorithms in [1], [2] and [3]

4. OPEN ISSUES AND FUTURE RESEARCH DIRECTIONS

The analysis of reason for more area consumption by VLM3 algorithm than the algorithm in [2] and [3] and adapting new approaches to reduce area for VLM3 algorithm is one of the directions to optimize the compared VLM3 algorithm implementation. It can be suggesting that the look up table (LUT) is used as part of generator can be replaced with virtual memory concept, so the area consumption can be greatly reduced for VLM3 algorithm and so the concern about space complexity of the this algorithm is optimized. Additionally, the encoding methodology used in the paper [1] is canonical recoding algorithm, which has some drawbacks with respect to the length of the sequence such as bits of multiplier generated sequentially, number of add/subtracts and length of shifts is variable, so comparatively it may be difficult to implement and it would be efficient if other encoding techniques which has the uniform shifts of 2 bits each.

After all this, the time complexity analysis for an algorithm helps to analyses how fast the algorithm is working to get the final results, so normally the Big-O-Notation is used to express the time complexity. In this VLM3 algorithm, assuming that *j*=N, the time complexity can be expressed as two cases depends on the conditions, for condition 1: *O(N)=10+15N*, if $f^{(i)} = 3$ and condition 2: *O(N)=10+16N*, if $k_i = 0$, from this condition 1 is the best case and the other condition is the worst case analysis.

The optimization of system clock is a challenging task to design with avoidance of clock jitter. The source of the clock jitter as well as the amount of tolerable jitter has not realized for the VLM3 algorithm system design and it is an open issue. Similarly, scheduling schemes with pipelining stages can be employed to this discussed algorithm in [1]. So that some more significance can be placed to efficiently and effectively coordinate the jitter and scheduling and this leads to the effectual administration of time complexity. The implementation of the scheduling with proper clocking scheme to the VLM3 algorithm system design is an open topic. Hence the feasible scheduling with proper memory management is an important future development of VLM3 algorithm design implementation along with the space complexity and time complexity analysis.

On implementing the scheme of scheduling to this algorithm in [1], the demand on memory can be changed. In the paper [1], it has been using four registers and a 3-bit shift register, the chances are there that this convention can be changed according to the feasibility of rescheduling and what method of scheduling is going to be employed. So the feasibility of memory manipulation in an effective way is an open issue of the scheme, because the practical implementation has various challenges to meet the memory requirements. Once the implementation of the architecture is done, then from the real time physical constraints, side channel attack can be analyzed and corresponding measures of side channel resistance can be taken to improve the efficiency of the processor.

5. CONCLUSION

This paper overlay the recent exertion and inspection of the work in implementation of multiplication operation for the cryptographic applications. To make computation easier in the cryptographic processor, modular multiplication plays a vital role in the processes. A short background to the modular multiplication is presented by different methods are following, Montgomery multiplication algorithm, Karatsuba algorithm and VLM3 algorithm are discussed in this paper and their performance is analyzed for different parameters such as area, frequency and critical path delay.

Among the discussed algorithms, though VLM3 consumes more area than others, but it achieves in less time and with reasonable frequency compared to Montgomery algorithm and Karatsuba Algorithm. The open research issues considering the design of scheduling and memory management have been discussed to further possible research investigations. This will give a new access in particular state of art in the field of multiplier design for cryptographic processor. This exertion will hopefully lead the researchers and reader to implement in the possible future scheduling procedure to VLM3 algorithm.

REFERENCES

[1] Rezai, Abdalhossein., Keshavarzi, Parviz. (2015). High-Throughput Modular Multiplication and Exponentiation Algorithms Using Multibit-Scan– Multibit-Shift Technique, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 23(9):1710-1719.

ISSN: 2456 - 3935
International Journal of Advances in Computer and Electronics Engineering
Volume: 02 Issue: 02, February 2017, pp. 28 – 33

[2] Benaissa, Mohammed., Zia-Uddin-Ahamed Khan. (2015). Throughput/Area-efficient ECC Processor Using Montgomery Point Multiplication on FPGA, *IEEE Transactions on Circuits and Systems—II: Express Briefs* 62(11):1223-1232.
[3] Lijuan Li., Shuguo Li. (2016). High-Performance Pipelined Architecture of Elliptic Curve Scalar Multiplication Over GF(2m*), IEEE Transactions On Very Large Scale Integration (VLSI) Systems* 24(4):1223-1232.
[4] Laszlo Hars. (2003). Long Modular Multiplication for Cryptographic Applications, B.S. Kaliski Jr. et al. (Eds.): *Springer-Verlag* Berlin Heidelberg: 57–70.
[5] Montgomery , P. L. (1985). Modular multiplication without trial division, *Math.Comput* 44(170):519–521.
[6] Barrett , Paul. (1986). Implementing the Rivest Shamir and Adleman Public Key Encryption Algorithm on a Standard Digital Signal Processor, chapter in Advances in Cryptology — *Proceedings of CRYPTO' 86* 263:311-323.
[7] Knezevic, Miroslav. (2010). Faster Interleaved Modular Multiplication Based on Barrett and Montgomery Reduction Methods, *IEEE Transactions on Computers* 59(12):1715 – 1721.
[8] Tilborg, Henk C.A. van., Jajodia , Sushil. Encyclopedia of Cryptography and security"ISBN 978-1-4419-5905-8, *Springer Book*.
[9] Kong, Yinan., Phillips, Braden. (2010). Revisiting Sum of Residues Modular Multiplication, *Journal of Electrical and Computer Engineering* 2010 (657076):9.
[10] Shieh, M.-D., Chen, J.-H., Wu, H.-H., Lin, W.-C. (2008). A new modular exponentiation architecture for efficient design of RSA cryptosystem, *IEEE Transactions in Very Large Scale Integration. (VLSI) System* 16(9):1151–1161.
[11] Koblitz , N., Menezes, A., Vanstone, S. (2000). The state of elliptic curve cryptography, Des. Codes Cryptogaphy 19(2/3):173–193.
[12] Hankerson, R., Menezes, A., Vanstone, S. (2004). Guide to Elliptic Curve Cryptography. New York, NY, USA: *Springer-Verlag*.
[13] David , Jean Pierre., Kalach, Kassem., Tittley, Nicolas. (2007). Hardware Complexity of Modular Multiplication and Exponentiation, *IEEE Transactions on Computers* 56(10):1308-1319.
[14] Karatsuba , A., Ofman, Yu. (1963). Multiplication of Many-Digital Numbers by Automatic Computers, Proceedings of the USSR Academy of Sciences 145: 293–294. *Translation in the academic journal Physics-Doklady*, 7: 595–596, 1963.

Authors Biography

Poomagal C.T. is currently pursuing Ph.D in Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Sriperumbudur, TamilNadu, India. She received her M.E degree from Sri Venkateswara College of Engineering, Sriperumbudur, TamilNadu, India. Her current research interests include VLSI architectures for cryptography algorithms and VLSI Signal Processing Algorithms.

Sathish Kumar G.A. is Professor of Department of Electronics and Communication Engineering, Sri Venkateswara College of Engineering, Sriperumbudur, TamilNadu, India. He obtained his M.E degree from PSG College of Technology, Coimbatore, India. He has completed PhD in Anna University, Chennai, India. His research interest is Network Security, Image Processing, and VLSI Signal Processing Algorithms.