International Journal of the Physical Sciences Vol. 5(8), pp. 1290-1295, 4 August, 2010
Available online at http://www.academicjournals.org/ijps
ISSN 1992 - 1950 ©2010 Academic Journals

Full Length Research Paper

Improving multiplication and reminder using implementation based on word and index

Malihe Danesh, Hossein Shirgahi* and Najmeh Danesh

Young Researchers Club, Islamic Azad University, Jouybar Branch, Jouybar, Iran.

Accepted 12 August, 2010

Asynchronous cryptography is one of the most widely used cryptographic algorithms. Much research has been done to find other applications of this cryptography method. Modular exponentiation is one of the primitive operations in these algorithms. Since we need to perform multiplication, square and modular division, most aspects of these studies were based on optimizing each of these algorithms. Since access to faster software or hardware techniques was the objective of these studies, after studying optimized implementation methods of multiplication and division, we tried to take a step toward the application of methods of software implementation of some examples to decrease implementation time and increase the efficiency of algorithms of this type in this paper.

Key words: Reduction division, asynchronous cryptography, modular multiplication, Karatsuba-Ofman multiplication.

INTRODUCTION

Modular exponentiation is one of the important and primitive operations performed in the algorithms of asynchronous cryptography. This operation is used in most algorithms of asynchronous cryptography such as RSA, diffie-Hellman key exchange, ElGamal signature design and DSS algorithm. Different software and hardware techniques were used for better implementation considering the importance of modular exponentiation. What is important for the software implementation of the algorithms of modular exponentiation is modular multiplication which is performed in each exponentiation algorithm. We need to implement a modular multiplication to calculate M^e (mod n). Three ways of calculating modular multiplication are they are represented by R: =a.b (mod n) and a, b and n are k bit integers are as follows:

1. Multiplication and reduction: In this method at first the multiplication of a and b is performed separately and then reduction by dividing the product by n.
2. Blackley method: The stages of multiplication and reduction are combined in this method (Blakley, 1983).
3. Montgomery method: This algorithm represents residue class to modulus n and uses residue account (Montgomery, 1985).

In the three methods mentioned above, the operations of modular reduction and multiplication are needed. Considering the importance of the operations of cryptography and decryption, much effort was done in accelerating the algorithms of modular reduction and multiplication and in decreasing their required time especially in big numbers. Next section deals with some of these important algorithms. Then we present our recommended methods on multiplication and remainder algorithms to increase the efficiency of some of these algorithms and compare the results of their implementation with previous methods. Finally we summarize the obtained results.

RELEVANT WORK

Several methods were offered for improving remainder and multiplication algorithms. Each method aims to increase the speed of these algorithms by presenting software and hardware implementations (Barrett, 1986;

*Corresponding author. E-mail: hossein.shirgahi@gmail.com.

```
Function KORMA(a,b)
 $t_0$ = KORMA($a_0$ , $b_0$)
 $t_2$ = KORMA($a_1$ , $b_1$)
 $u_0$ = KORMA($a_1+a_0$ ,
$b_1+b_0$)
 $t_1 = u_0 - t_0 - t_2$
return ( $2^{2h} t_2 + 2^h t_2 + t_0$)
```

Figure 1. Karatsuba-Ofman algirthm.

Laurichesse and Blain, 1991). If k represents number of bits of mentioned numbers, then standard multiplication algorithm with time order O(k^2) in (Sloan, 1985), Karatsubu-Ofman with time order O($K^{1.58}$) in (Knuth, 1981; Bewick, 1994; Nedjah and Mourelle, 2003) and multiplication algorithm based on FFT with time order O(klogk) in (Shindler, 1997; Zuras, 1993; Dordevic et al. 2002) are among the commonest and most efficient which are used in hardware implementation. Karatsuba-Ofman multiplication algorithm is the best option for hardware implementation.

Unreductible and reduction division are among the most efficient reduction algorithms which are used in hardware and software implementation and has the time order O(kn) , which k is the number of bits (Booth, 1951; Brickell, 1983; Brickell et al. 1992). Since the work done in this paper is based on the software implementation of Karatsuba-Ofman algorithm in multiplication and reduction division algorithm in modulation, will explained briefly.

Karatsuba-Ofman

This algorithm uses a method for performing multiplication which needs less O(k^2) in bit action. The method was proposed by two mathematicians named Karatsuba and Ofman in 1962. The method is explained briefly here. At first the two numbers of a and b are divided equally:

a: $2^h a_1+a_0$, b: $2^h b_1+b_0$

Multiplication algorithm splits them into parts of b_1 b_0 a_1 a_0. We need a three k-bit multiplication and adding the results to calculate the multiplication of a by b. Figure 1 pseudocode represents the operations existing in this algorithm clearly.

The performance time is 3t (k/2) + βk, where βk represents the operations of addition, multiplication and shift in this algorithm. Solving this it was found out that the complexity of this algorithm is O($k^{\log_2 3}$) = O($k^{1.58}$).

Karatsuba-Ofman algorithm is faster in complexity than standard one. There is a load thanks to the recursive identity of the algorithm. Since access to faster speed is the objective, we plan to increase the efficiency of the above algorithm by decreasing the computational load.

Reduction division algorithm

Following the multiplication, there is the step of division by which the remainder is computed. Since we need the value of remainder we can simplify the steps of division algorithm to speed up the process. The division step can be performed by one of division algorithms considering the dividend t, divisor n and remainder R so that they can use shift and subtraction n from t alternatively until the remainder k is obtained in the range $0 \le R \le n$. Moreover, a negative remainder may be obtained after subtraction. Reduction division algorithm is an example of the algorithms used while confronting a negative remainder.

In the existing procedure in pseudocode, Figure 1 balances the operands t and n on the left at first. Since t, a 2k-bit number and n, a k-bit number, balancing on the left causes them to be shifted to left that means computation starts with 2^kn.

We consider R_i as the remainder in ith step of algorithm. We consider R equals t(R_0 = t). Then the shifted value of n is subtracted from t to obtain R1. If R1 is a positive number or zero, we go to the next step; otherwise the remainder is retrieved to its previous value.

Reduction division algorithm performs k subtractions to reduce 2k-bit t to k-bit n. This causes algorithm, especially for large integers to take long. As its clear, the complexity of this algorithm is O (kn).

METHODS

Proposed method in multiplication

Here we intend to use a method of software imple-mentation of Karatsubofman algorithm to improve its efficiency in multiplication. In fact we plan to decrease the effect of recursive load in the algorithm to increase its speed and gain access to better operation time in large integers. Our proposed method is based on decreasing the number of recursive routines in which we used the property based on word instead of bit. All operations are performed based on bit and in each step the bit length of the number is decreased to half and the recursive procedure is recalled.

Our proposed method on Karatsub-Ofman algorithm is that we converted numbers to words with the same length instead of bit division and halving operation is performed word by word so that in each step the number of desired words is halved and this trend continues to reach a word in both numbers. In other words we evaluate the input parameters of the recursive algorithm of KORMA based on word length instead of bit length and perform division based on word. As a result, the number of recursive procedures decreases until the algorithm ends. Also, this method was implemented with words of different lengths. The results obtained are shown next section. If the length of the desired word is a multiple of bit length of a and b, we add some zeros to the above

Table 1. Time of using word based method in multiplying two N bit numbers.

Time(ms)	Bit based	Byte based	Short word based	Word based
N=32	1734	21.2	4.9	0.3
N=64	6542	97.1	22.2	4.8
N=128	25967	396.2	100.1	26.3
N=256	105621	1628.4	430.7	116.2
N=512	364333	6633.3	1621.5	412.7
N=1024	1136652	27877.2	5843.4	1393.5

```
Input: t,n
Output: R= t mod n
    1.  $R_0$=t
    2.  n=$2^k$n
    3.  for i=1 to k
    4.      $R_i$:=$R_{i-1}$-n
    5.      if $R_i$<0 then $R_i$:=$R_{i-1}$
    6.      n:=n/2
    7.  return $R_k$
```

Figure 2. Reduction division algorithm.

numbers according to the bit length and apply the algorithm.

To compare the proposed method with the one based on bit, we performed the software implementation of the multiplication algorithm and implemented it in a system with specifications 512 MB RAM, CPU Pentium (III), 800 MHz using C++.

THE PROPOSED METHOD IN REMAINDER

In this section, we tried to increase efficiency by speeding up the reduction division algorithm followed by a decrease in the operation time. For this reason, we used the property based on word in this method and used a word by word reduction instead of bit by bit reduction. This is how the proposed method works: at first we converted the dividend and divisor into words with fixed length and applied the algorithm to them. In the second step in the algorithm of Figure 2 we shift the divisor words by k instead of bit shifting. Then we used word by word reduction in iteration ring instead of bit by bit reduction.

Finally we shift the divisor words one unit into right side. As a result the number of reduction steps is based on the difference of the words of the two numbers which is much less than the differenced in their bits. Another improvement on this algorithm, we used an index instead of transfer which had a great effect on the decrease in the time needed for implementing the algorithm. Next section deals with the results obtained from the synchronous implementation of these two methods on the efficiency of the reduction division algorithm.

RESULTS

Multiplication

The results are shown in Table 1. The lines of the table represent number of bits of the desired integers and columns represent length of words, so that the first column which is based on bit, equals the main Karatsub-Ofman algorithm and other columns are based on bite, short word and word respectively. The values in the table show the time needed for the multiplication of the two numbers in second. This method had a suitable effect on the operation time of the above algorithm. In other words the increase in word size in numbers with a fixed bit length decreased the operation time of the algorithm remarkably. This decrease is more remarkable in numbers with bit length over 250 bits.

In Figure 3, we studied the effect of increase in word size with numbers with fixed bit length that time in word based method in multiplying two N bit numbers is better than other methods. Since the time changes of program implementation is very much for numbers with different bit length especially for larger integers, we used logarithmic representation in the vertical axis. The efficiency improves in an exponential way as the number of bits increases which shows a considerable improvement of the proposed method. The improvement in the implementation based on 32 bit words is at least 10^2 which are shown in Figure 4.

Remainder

The results obtained from studying the time of performing software implementation of the above proposed method on reduction division algorithm are shown in Table 2. We used the same system in the previous section. As you can see in Table 2 the operation time of the algorithm has decreased compared with the original algorithm. The amount of increase had more changes as the word length increases. The effect based on word on the numbers with bit length is more remarkable.

The diagram of the results is shown in Figure 5. The efficiency of the proposed method improves as the number of bits of words increases that time is based on

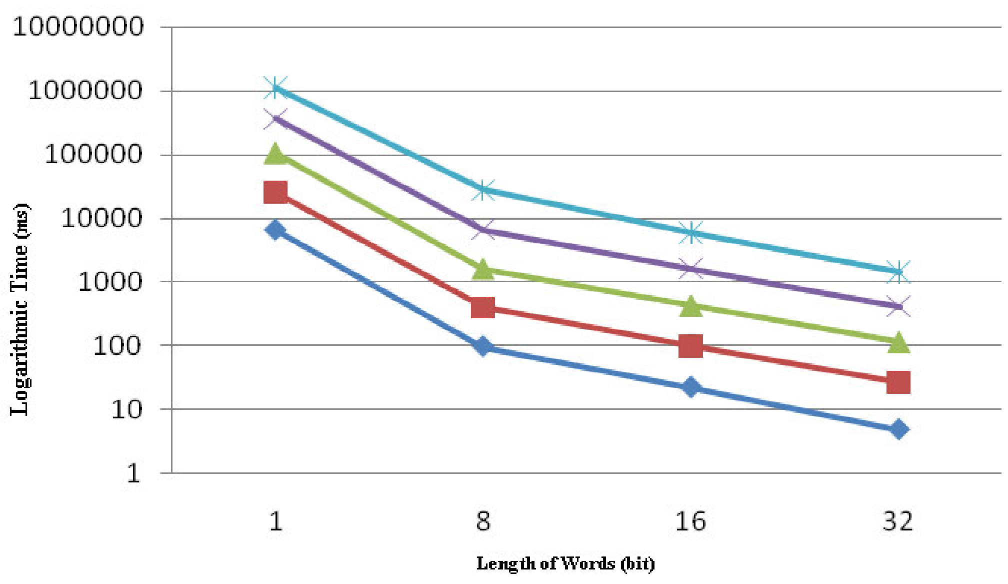

Figure 3. The effect of increasing word size in numbers with a fixed length bit.

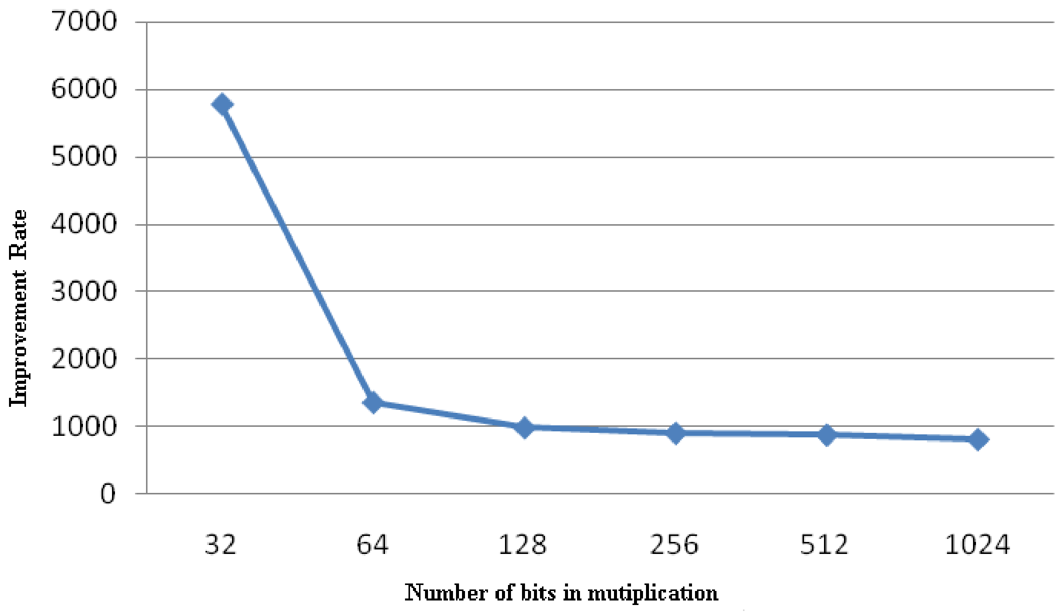

Figure 4. Improvement rate in proposed method per 32-bit word multiplication.

Table 2. Time of method based on word and using Index in calculated the remaining.

Based on word	Based on short word	Based on bite	Based on bit	Time(ms)
0.0212	0.027	0.036	0.185	N=32
0.0475	0.0725	0.1125	0.69	N=64
0.1512	0.2425	0.4	2.67	N=128
0.3931	0.83	1.48	7.15	N=256
1.104	3.08	5.4	21.4	N=512
3.3377	7.2680	17.494	67.223	N=1024

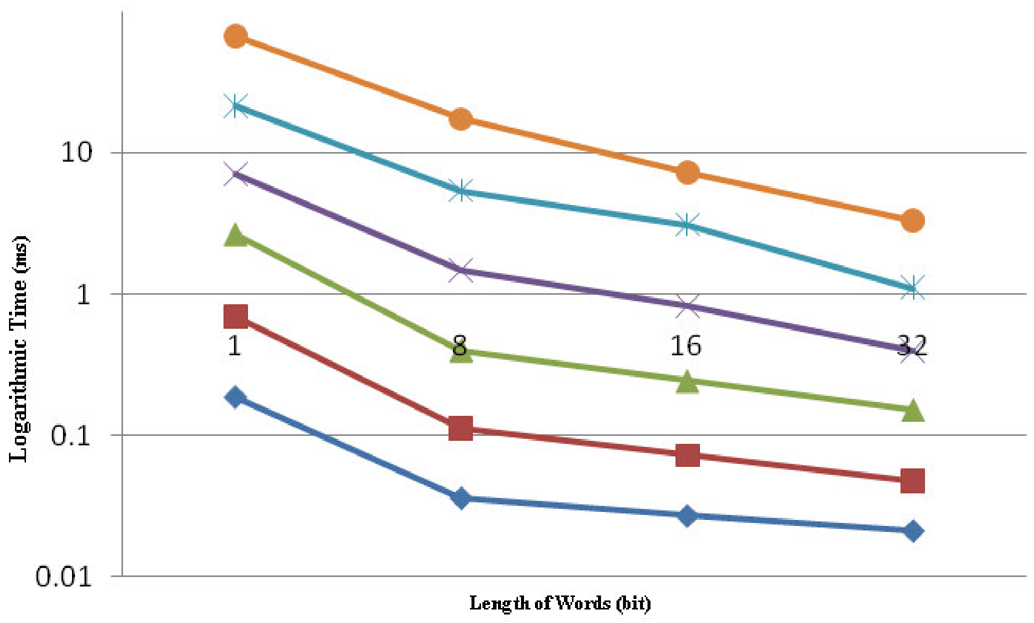

Figure 5. Effect of increasing word size and using index on numbers with fixed length bit.

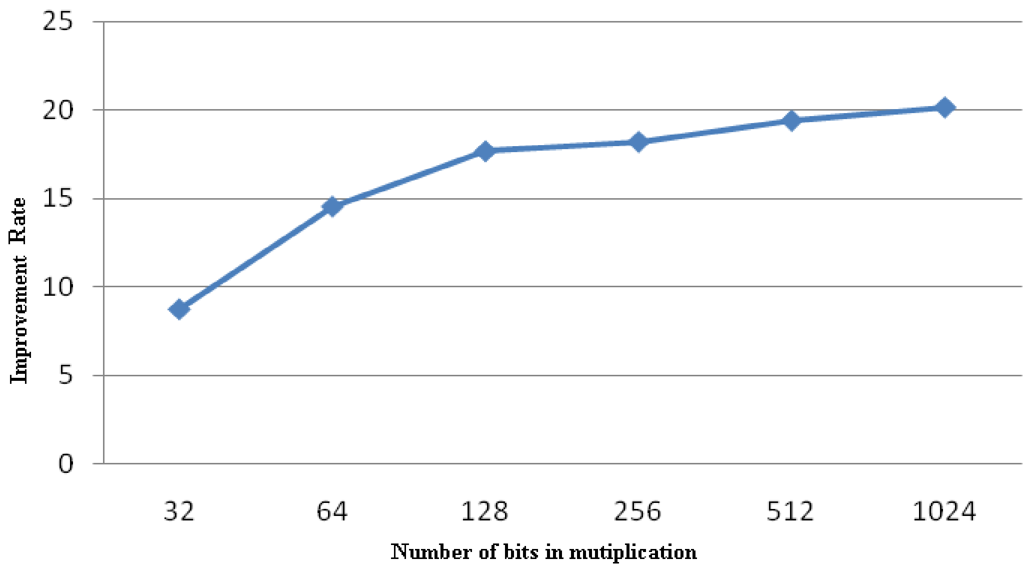

Figure 6. Improvement rate in proposed method per 32-bit word in remaining operation.

word method and using index in calculating the remaining is efficient. This is more obvious in 32 bit words which are shown in Figure 6. The efficiency is more than the original algorithm in numbers with any bit length.

Conclusion

After studying examples of the most efficient methods of remainder and multiplication operations used in asynchronous cryptography algorithms, actions were taken to improve the efficiency of some of them. Two methods based on word and index was used.

The results shows the operation time of Karatsubofman algorithms in 32 bit words decreased at least 10^{-3} times as compared to traditional methods and the operation time of reduction division algorithms for computing the remainder decreased at least with coefficient 0.1 in the proportion of older algorithms. In fact when the bit length of the word increased in multiplication the increased improvement was exponential and at least 10 times more.

REFERENCES

Barrett P (1986). Implementating the Rivest, Shamir and Aldham public-key encryption algorithm on standard digital signal processor, Proceedings of CRYPTO'86, Lecture Notes Comput. Sci. Springer-Verlag, 263: 311-323.

Bewick GW (1994). Fast multiplication algorithms and implementation, Ph. D. Thesis, Department of Electrical Engineering, Stanford University, United States of America.

Blakley GR (1983). A Computer Algorithm for the Product AB Modulo M, IEEE Trans. Comput., 32(5): 497-500.

Booth A (1951). A signed binary multiplication technique, Quarterly J. Mechanics Appl. Maths. pp. 236-240.

Brickell EF (1983). A Fast Modular multiplication Algorithm with Application to Two key Cryptography, in Advances in Cryptology, Proc. Crypto, New York, '86: 51-60.

Brickell EF, Gordon DM, McCurley KS, Wilson DB (1992). Fast Exponentiation with Precomputation, Proc. Eurocrypt, Springer, 92(658): 200-207.

Dordevic G, Unkasevie T, Markovic M (2002). Optimization of Modular Reduction Procedure in RSA Algorithm Implementation on Assembler of TMS320C54x Signal Processors, Proc. 14th Int. Conf. Digital Signal Processing, pp. 811-814.

Knuth DE (1981). The art of computer programming: seminumerical algorithms, vol 2, 2nd Edition, Addison-Wesley, Reading, Mass.

Laurichesse D, Blain L (1991). Optimizing implementation of RSA cryptosystem, comput. Secure, May, 10(3): 263-267.

Montgomery PL (1985). Modular multiplication without trial division, Math. Comput., 44: 519-521.

Nedjah N, Mourelle LM (2003). Hardware simulation model suitable for recursive computations: Karatsuba-Ofman's multiplication algorithm, Proceedings of ACS/IEEE International Conference on Computer Systems and Applications, Tunis, Tunisia, July.

Shindler V (1997). High-speed RSA hardware based on low-power piplined logic, Ph. D. Thesis, Institut für Angewandte Informations-verarbeitungund Kommunikationstechnologie, Technishe Universität Graz, January.

Sloan KR (1985). Comments on 'A Computer Algorithm for Calculating the Product AB Modulo M', IEEE Trans. Comput., March, C-34(3): 290-292.

Zuras D (1993). On squaring and multiplying large integers, In Proceedings of International Symposium on Computer Arithmetic, IEEE Comput. Society Press, pp. 260-271.