Trends in Applied Sciences Research

ISSN 1819-3579

Academic
Journals Inc.

www.academicjournals.com

Trends in Applied Sciences Research 7 (3): 240-247, 2012
ISSN 1819-3579 / DOI: 10.3923/tasr.2012.240.247

A New CMM-NAF Modular Exponentiation Algorithm by using a New Modular Multiplication Algorithm

Abdalhossein Rezai and Parviz Keshavarzi
Electrical and Computer Engineering Faculty, Semnan University, Semnan, Iran

Corresponding Author: Abdalhossein Rezai, Electrical and Computer Engineering Faculty, Semnan University, 3513119111, Semnan, Iran Tel: +98-913-306081

ABSTRACT

This study presents a novel modular multiplication algorithm based on a new multiplier Non-Adjacent Form (NAF). In this new multiplier representation, the sliding window method is applied on the canonical recoded multiplier to reduce the average number of multiplier digits and thereby the average number of the required clock cycle for the modular multiplication algorithm. Moreover, a new and efficient modular exponentiation algorithm is presented based on this new modular multiplication algorithm, Common-Multiplicand-Multiplication (CMM) method and parallel structure. In this new CMM-NAF modular exponentiation algorithm, not only the common part of the modular multiplication is computed once rather than several times but also the modular multiplication and modular squaring operations are performed in parallel. Our analysis shows that the average number of required clock cycle in the proposed CMM-NAF modular exponentiation algorithm is reduced in comparison with other modular exponentiation algorithms considerably.

Key words: Public-key cryptography, security, fast modular exponentiation, Montgomery modular multiplication, non-adjacent form

INTRODUCTION

Recently, cryptography algorithms and protocols have been used to provide the information security (Wang, 2011; Sharma *et al*., 2006; Zaidan *et al*., 2010; Jaberi *et al*., 2012; Nikooghadam *et al*., 2008). Public-Key Cryptography (PKC) is an important component of the cryptography (Olorunfemi *et al*., 2007; Xia *et al*., 2010; Azeez *et al*., 2012; Salem *et al*., 2011). The modular exponentiation with large modulus is a crucial operation in many PKCs such as RSA and DSA (Rabah, 2006). This operation is implemented by repeating modular multiplication. So, the efficiency of many PKCs is determined by the efficiency of the modular multiplication algorithm and the number of required modular multiplication in the modular exponentiation algorithm (Nedjah and Murelle, 2009a; Wu, 2009a).

Montgomery modular multiplication algorithm (Montgomery, 1985) is an efficient algorithm for the modular multiplication because it avoids division by the modulus (Wu, 2009a; Nedjah and Murelle, 2009a; Xiangyan *et al*., 2009).

There are many research efforts to speed up the performance of the Montgomery modular multiplication algorithm such as high-radix design (Pinckney and Harris, 2008; Tawalbeh *et al*., 2005), scalable design (Shieh and Lin, 2010; Pinckney and Harris, 2008), parallel calculation quotient and partial result (Keshavarzi and Harrison, 1998) and signed-digit recoding (Koc and Hung, 1992; Philips and Burgess, 2004).

Moreover, there are many research efforts to reduce the number of modular multiplication such as sliding window method (Nedjah and Murelle, 2009a, b), signed-digit recoding (Wu, 2009a; Wu *et al*., 2008) and Common-Multiplicand-Multiplication (CMM) method (Ha and Moon, 1998; Wu, 2009a, b; Rezai and Keshavarzi, 2011).

This study presents a new modular multiplication algorithm based on the Non-Adjacent Form (NAF) and sliding window method. In this new algorithm, sliding window method is applied on the canonical recoded multiplier to reduce the average number of the multiplier digits and thereby, the average number of the required clock cycle (multiplication step) for the modular multiplication algorithm. Moreover, this study presents a novel modular exponentiation algorithm, called CMM-NAF modular exponentiation algorithm, by using this new modular multiplication, the CMM method and the parallel structure. Present analysis shows that the average number of required clock cycle in the CMM-NAF exponentiation algorithm is reduced in comparison with other CMM modular exponentiation algorithms considerably.

PRELIMINARIES

The Montgomery modular multiplication algorithm: Montgomery (1985) proposed a modular multiplication algorithm which speeds up the modular multiplication and modular exponentiation algorithm. The Montgomery modular multiplication algorithm replaces the trial division by the modulus with a simple right shift. Algorithm 1 shows the radix-2 Montgomery modular multiplication algorithm.

Algorithm 1: The radix-2 Montgomery modular multiplication algorithm(Mont(X,Y))

Input: X,Y,M;
Output: $S(n) = XY2^{-n} \bmod M$
1. S (0) = 0;
    For I = 0 to n-1 Do
2. $q_i = (S(I) + x_iY) \bmod 2$;
3. $S(I+1) = (S(I) + x_iY + q_iM)/2$;
4. If $S(n) \geq M$ Then S (n)-M

In this algorithm, the inputs are n-bit X, Y and M. The output is $S(n) = XY2^{-n} \bmod M$. This algorithm computes $S(n) = XY2^{-n} \bmod M$ in n-clock cycle. So, it is a time-consuming operation.

The modular exponentiation algorithm: As modular exponentiation consists of series of the modular multiplications, the performance of the modular multiplication is determined by the efficiency of the implementation of the modular multiplication (Wu, 2009a; Nedjah and Murelle, 2009a). The Right-to-Left (R2L) Montgomery modular exponentiation algorithm is shown in Algorithm 2:

Algorithm 2: The R2L Montgomery modular exponentiation algorithm

Input: A,E,R,N;
Output: $C = A^E \bmod N$;
1. S: = AR mod N, C: = R mod N;
    For I = 0 to k-1 Do
2. If ($e_i = 1$) Then {C = Mont (S, C), S = Mont (S, S)};
3. Else S = Mont (SS);
4. C: = Mont (C, 1);

In this algorithm, the inputs are A, E, $R = 2^n$ and N. The output is $C = A^E \bmod N$. In Algorithm 2, when the exponent bit is nonzero, both Mont (S, C) and Mont (S, S), are performed. Ha and Moon (1998) proposed the common part in Mont (S, C) and Mont (S, S) can be computed once rather than twice. There are many attempts (Ha and Moon, 1998; Wu *et al*., 2008; Wu, 2009a, b) to speed up the performance of the modular exponentiation algorithm based on this idea. One of the recent attempts is the parallel CMM-CSD Montgomery algorithm (Wu, 2009b) which is shown in Algorithm 3.

Algorithm 3: The parallel CMM-CSD Montgomery modular exponentiation algorithm

Input: M, E_{CSD}, N, R;
Output:$C = M^{E[1]}$[mod N; $D = M^{E[-1]} \bmod N$;
1. S = MR mod N, C = D = R mod N;
    For I = 0 to m Do
        Parallel begin
2. If ($e_i = 1$), Then C = Mont(S, C);
3. If ($e_i = -1$), Then D = Mont(S, D);
4. S = Mont(S, S);
        Parallel End;
    Parallel begin
5. C = Mont(C, 1);
6. D = Mont(D, 1);
    Parallel End;

In this algorithm, the inputs are M, E_{CSD}, N and $R = 2^n$ where, E_{CSD} is canonical recoded E. the outputs are $C = M^{E[1]} \bmod N$ and $D = M^{E[-1]} \bmod N$. In Algorithm 3, the modular squaring and modular multiplication operations are executed in parallel. Moreover, the registers C and D are used to store the operation results based on the positive digit and negative digits in exponent E_{CSD}, respectively. This algorithm uses CMM method to compute the common part of three multiplications just once.

THE PROPOSED CMM-NAF MODULAR EXPONENTIATION

In serial-parallel multiplication, partial result shifts one bit per iteration. Moreover, multiplication by zero bit results in zero but this multiplication by zero is performed and implemented per iteration. In this study, we proposed a new modular multiplication by recoding and then by partitioning the multiplier. This new modular multiplication performs multiplication by zero partition with any length in only one clock cycle instead of several clock cycles. The proposed modular multiplication algorithm is shown in Algorithm 4.

Algorithm 4: The proposed modular multiplication algorithm

Input: X, Y, M;
Output: $P: = XY2^{-n} \bmod M$;
1. P = 0;
{recoding phase}
2. Compute D by performed canonical recoding on X;
    parallel begin
        {partitioning phase}
3. Building Π(D) using algorithm 5;

Algorithm 4: Continued

4. Let m = # Π(D);
{pre-computation phase}
5. Compute and store V_iY
Parallel End
{multiplication phase}
6. For I = 0 to m-1 Do
7. P: = P +V_iY;
8. q: = $P_0M'_0$ mod 2^{li};
9. P: = (P+qM)/2^{li};
10. If (P>M) Then P = P-M;

In this algorithm, the inputs are n-bit integer X, Y and M. The output is $P = XY2^{-n}$ mod M. m is the number of partitions in the multiplier, l_i is the length (i.e., the number of digits) of ith partition and V_i is the ith partition value.

In the recoding phase of this new algorithm, the canonical recoding is performed on the multiplier. In the partitioning phase, the partitioning is performed on the resulted signed-digit multiplier. The partitioning strategy instrumented in this algorithm scans the multiplier from the least significant digit to the most significant digit according to Algorithm 5. In this strategy, the zero windows are allowed to have an arbitrary length but the maximum length of the nonzero windows should be the exacted value of d digit.

Algorithm 5: The partitioning algorithm

Input: D, d
Output: Π(D)
1. ZP: Check the incoming single digit;
2. If it is zero, Then stay in ZP
3. Else go to NZP;
4. NZP: Stay in NZP until all d digits are collected;
5. Check the incoming single digit;
6. If it is zero, Then go to ZP
7. Else go to NZP;

In the pre-computation phase of Algorithm 4, the least significant digit of the nonzero partition is either 1 or $\bar{1}$ which implies that the nonzero partition value is always an odd number. So, we require only pre-computation of V_iY for odd value of V_i. It should be noted that, the pre-computation phase and the partition phase are performed independently in parallel. This also speeds up the modular multiplication.

The multiplication phase of Algorithm 4 is performed m times. Recall that, m denotes the number of partitioning in the signed-digit multiplier. In each iteration of the multiplication phase of algorithm 5, l_i bits of the multiplier and n-bit multiplicand are processed.

We also proposed using this new modular multiplication algorithm in order to speeding up the CMM-CSD Montgomery exponentiation algorithm (Wu, 2009b) as shown in Algorithm 6.

Algorithm 6: The proposed CMM-NAF modular exponentiation algorithm

Input: M, E_{CSD}, N, R;
Output: $C = M^{E[1]} \mod N$; $D = M^{E[-1]} \mod N$;

Algorithm 6: Continued

```
Parallel begin
1. S: = M.R mod N, C: = D: = R mod N;
2. Compute Z by performed steps 2-5 of algorithm 4 on S after performing m0.R;
   Parallel End
      For I = 0 to m Do
         parallel begin
3.          If (ei = 1) Then C: = Z.C mod N; /* perform steps 6-10 of algorithm 4 for positive signed-digit */
4.          If (ei = -1) Then D: = Z.D mod N; /* perform steps 6-10 of algorithm 4 for negative signed-digit */
5.          S: = Z.S mod N; /* perform steps 6-10 of algorithm 4 */
6.          Compute Z by performed steps 2-4 of algorithm 4 on S after performing Z0.S;
      Parallel end;
   Parallel begin
7.    C: = C.1 mod N;          /* perform algorithm 4 */
8.    D: = D.1 mod N;          /* perform algorithm 4 */
    Parallel end;
```

In this algorithm, the inputs are M, E_{CSD}, N and $R = 2^n$ where, E_{CSD} is canonical recoded E. the outputs are $C = M^{E[1]} \bmod N$; $D = M^{E[1]} \bmod N$. In this algorithm, for $e_i = 1$, both Z.C mod N and Z.S mod N are computed in parallel. Similarly, for $e_i = \bar{1}$, both Z.D mod N and Z.S mod N are computed in parallel. In addition by using CMM method, the common part of two operations is computed once rather than twice.

In step 1 of this algorithm, S is computed by using Algorithm 4. In step 2, Z is computed by executing steps 2-5 of Algorithm 4 on S after computing m_0. R. In steps 3 and 4 of Algorithm 6, $M^{E[1]}$ and $M^{E[\bar{1}]}$ are computed based on the value of the e_i. These values are computed by executing steps 6-10 of Algorithm 4. In step 5 of Algorithm 6, the partial result S is computed by performing steps 6-10 of Algorithm 4. In step 6 of this algorithm Z is computed by performed steps 2-4 of Algorithm 4. This step is computed after computing Z_0.S. In this algorithm, the exponentiation operation $M^{E_{CSD}}$ is depicted as $M^{E_{CSD}} = C \times D^{-1}$.

EVALUATION

In the proposed modified modular multiplication algorithm, sliding window method is performed on the canonical recoded multiplier. So, the Hamming weight of the multiplier is 3n/3d+4. Therefore, based on the computational analyses of Ha and Moon (1998), for n-bit modulus, both operations Z.C mod N, and Z.D mod N require:

$$(\frac{6n}{3d+4}-2)(2n+1)+2(2n+1)=(\frac{6n}{3d+4})(2n+1)$$

Multiplication steps (clock cycles). Moreover, the operation Z.S mod N requires multiplication steps:

$$(\frac{6n}{3d+4}-2)n+2(2n+1)=\frac{6n^2}{3d+4}+2n+2$$

In the proposed CMM-NAF modular exponentiation for $e_i = 1$, both Z.C mod N and Z.S mod N are performed in parallel. Similarly, for $e_i = \bar{1}$, both Z.D mod N and Z.S mod N are performed in parallel.

In addition, the radix-2 signed-digit exponent is used. Thus, the probability of digits "0", "1" and "-1" is 2/3, 1/6 and 1/6, respectively. Therefore, the proposed modular exponentiation algorithm for k-bit exponent takes following multiplication steps:

$$\frac{2}{3}k[(\frac{6n}{3d+4})(2n+1)]+\frac{1}{3}k[(\frac{6n}{3d+4})(2n+1)]=k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})$$

However, the Dusse and Kaliski's exponentiation algorithm (Dusse and Kaliski, 1990), the Ha-Moon's improved Montgomery exponentiation algorithm (Ha and Moon, 1998), the CMM-MSD algorithm (Wu *et al*., 2008), the Wu's improved CMM-MSD exponentiation algorithm (Wu, 2009a) and the Wu's parallel CMM-CSD exponentiation algorithm (Wu, 2009b) require 1.5k ($2n^2$+n), 0.5k($5n^2$+4n), 0.5k($2n^2$+2n+0.75), 1.833k(n^2-n-2) and 0.5k($2n^2$+2n+5.33) multiplication steps, respectively. So, the proposed modular exponentiation algorithm reduces the required multiplication steps (clock cycles) in comparison with other similar work as follow:

$$1-\frac{k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})}{1.5k(2n^2+n)}\approx 1-\frac{4}{3d+4}$$ (Dusse and Kaliski, 1990)

$$1-\frac{k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})}{0.5k(5n^2+4n)}\approx 1-\frac{12}{2.5(3d+4)}$$ (Ha and Moon, 1998)

$$1-\frac{k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})}{1.833k(n^2-n-2)}\approx 1-\frac{12}{1.833(3d+4)}$$ (Wu, 2009a)

$$1-\frac{k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})}{0.5k(2n^2+2n+0.75)}\approx 1-\frac{12}{3d+4}$$ (Wu *et al*., 2008)

$$1-\frac{k(\frac{12n^2}{3d+4}+\frac{6n}{3d+4})}{0.5k(2n^2+2n+5.33)}\approx 1-\frac{12}{3d+4}$$ (Wu, 2009b).

Figure 1 summarizes these multiplication steps (clock cycles) improvement for the proposed CMM-CSD modular exponentiation algorithm in comparison with exponentiation algorithm by Dusse and Kaliski (1990), Ha and Moon (1998), Wu (2009a,b) and Wu *et al*. (2008) for various window widths.

The results show that this new CMM-NAF exponentiation algorithm reduces the average number of required clock cycles by about 69.2-88.2, 63.1-85.9, 7.7-64.7, 49.6-80.7 and 7.7-64.7% in comparison with Dusse and Kaliski (1990), Ha and Moon (1998), Wu *et al*. (2008) and Wu (2009a, b), respectively for d = 3-10. In addition, the algorithm complexity of the proposed CMM-NAF modular exponentiation algorithm is reduced in comparison with Wu (2009a) and Rezai and Keshavarzi (2011).

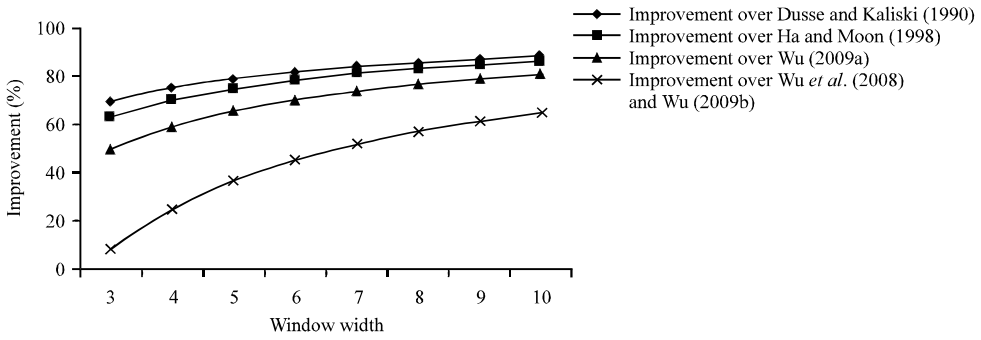

Fig. 1: The clok cycle (multiplication step) improvement of the proposed CMM-NAF modular exponentaition algorithm

CONCLUSIONS

This study presents a novel modular exponentiation algorithm based on a novel modular multiplication, called CMM-NAF modular exponentiation algorithm. The CMM-NAF modular exponentiation uses other techniques such as CMM method and parallel processing. In the proposed modular multiplication algorithm, the sliding window method is performed on the signed-digit multiplier to reduce the average number of multiplier digits and thereby, to reduce the average number of the required clock cycles for the modular multiplication algorithm. Using the CMM method in the CMM-NAF modular exponentiation algorithm, the common part of the modular multiplication is computed once rather than several times. Moreover, by using the parallel processing, the speed of the proposed modular exponentiation increases considerably. The results show that the average number of required clock cycles (multiplication steps) in the proposed CMM-NAF exponentiation algorithm is reduced by 69.2-88.2, 63.1-85.9, 7.7-64.7, 49.6-80.7 and 7.7-64.7% in comparison with Dusse and Kaliski (1990), Ha and Moon (1998), Wu *et al*. (2008) and Wu (2009a, b), respectively for d = 3-10.

REFERENCES

Azeez, N.A., A.P. Abidoye, K.K. Agbele and A.O. Adesina, 2012. A dependable model for attaining maximum authentication security procedure in a grid based environment. Trends Appl. Sci. Res., 7: 78-86.

Dusse, S.R. and B.S. Kaliski, 1990. A Cryptographic Library for the Motorola DSP 56000. In: Advance in Cryptology, Damgard, I.B. (Eds.). Springer-Verlag, Berlin, Germany, pp: 230-244.

Ha, J.C. and S.J. Moon, 1998. A common-multiplicand method to the Montgomery algorithm for speeding up exponentiation. Inf. Proc. Lett., 66: 105-107.

Jaberi, A., R. Ayanzadeh and A.S.Z. Mousavi, 2012. Two-layer cellular automata based cryptography. Trends Applied Sci. Res., 785: 68-77.

Keshavarzi, P. and C. Harrison, 1998. A new modular multiplication algorithm for VLSI implementation of public-key cryptography. Proceedings of the 1st International Symposium on Communication Systems and Digital Signal Processing, April 6-8, 1998, Sheffield Hallam University Press Learning Centre, UK, pp: 516-519.

Koc, C.K. and C.Y. Hung, 1992. Adaptive mary segmentation and canonical recoding algorithms for multiplication of large binary numbers. Comput. Math. Appl., 24: 3-12.

Montgomery, P.L., 1985. Modular multiplication without trial division. Math. Comput., 44: 519-521.

Nedjah, N. and L.M. Mourelle, 2009a. A hardware/software co-design versus hardware-only implementation of modular exponentiation using the sliding-window method. J. Circuts, Syst. Comput., 18: 295-310.

Nedjah, N. and L.M. Mouller, 2009b. High-performance hardware of the sliding-window method for parallel computation of modular exponentiations. Int. J. Para. Prog., 37: 537-555.

Nikooghadam, M., M.R. Bonyadi, E. Malekian and A. Zakerolhosseini, 2008. A protocol for digital signature based on the elliptic curve discrete logarithm problem. J. Applied Sci., 8: 1919-1925.

Olorunfemi, T.O.S., B.K. Alese, S.O. Falaki and O. Fajuyigbe, 2007. Implementation of elliptic curve digital signature algorithms. J. Software Eng., 1: 1-12.

Philips, B. and N. Burgess, 2004. Minimal weight digit set conversions. IEEE Trans. Comput., 53: 666-677.

Pinckney, N. and D. Harris, 2008. Parallelized radix-4 scalable montgomery multipliers. J. Integr. Circuits Syst., 3: 39-45.

Rabah, K., 2006. Implementing secure RSA cryptosystems using your own cryptographic JCE provider. J. Applied Sci., 6: 482-510.

Rezai, A. and P. Keshavarzi, 2011. High-performance modular exponentiation algorithm by using a new modified modular multiplication algorithm and common-multiplicand-multiplication method. Proceedings of the World Congress on Internet Security, February 21-23, 2011, London, United Kingdom, pp: 192-197.

Salem, Y., M. Abomhara, O.O. Khalifa, A.A. Zaidan and B.B. Zaidan, 2011. A review on multimedia communications cryptography. Res. J. Inform. Technol., 3: 146-152.

Sharma, S., R.C. Jain and S. Bhadauria, 2006. A power efficient encryption algorithm for multimedia data in mobile Ad hoc network. Trends Applied Sci. Res., 1: 416-425.

Shieh, M. and W. Lin, 2010. Word-based Montgomery modular multiplication algorithm for low-latency scalable architectures. IEEE Trans. Comput., 59: 1145-1151.

Tawalbeh, L.A., A.F. Tenca and C.K. Koc, 2005. A radix-4 scalable design. IEEE. Potentials, 24: 16-18.

Wang, Y., 2011. Unconditional security of cryptosystem: A review and outlook. Trends Applied Sci. Res., 6: 554-562.

Wu, C.L., D.C. Lou and T.J. Chang, 2008. An efficient montgomery exponentiation algorithm for public-key cryptosystem Proceedings of the International Conference on Intelligence and Security Information, June 17-20, 2008, Taipei, Taiwan, pp: 284-285.

Wu, C., 2009a. Fast parallel montgomery binary exponentiation algorithm using canonical signed-digit recoding technique. Proceedings of the 9th International Conference on Algorithms and Architectures for Parallel Processing, June 8-11, 2009, Taipei, Taiwan, pp: 428-438.

Wu, C.L., 2009b. An efficient common-multiplicand-multiplication method to the Montgomery algorithm for speeding up exponentiation. Inf. Sci., 179: 410-421.

Xia, Y., L. Kuang and M. Zhu, 2010. A hierarchical access control scheme in cloud using HHECC. Inform. Technol. J., 9: 1598-1606.

Xiangyan, F., Z. Jiahang, X. Tinggang and Y. Youguang, 2009. The researcher and implement of high-speed modular multiplication algorithm basing on parallel pipelining. Proceedings of the Asia-Pacific Conference on Information Processing, July 18-19, 2009, China, pp: 398-403.

Zaidan, A.A., B.B. Zaidan, A.K. Al-Frajat and H.A. Jalab, 2010. An overview: Theoretical and mathematical perspectives for advance encryption standard/rijndael. J. Applied Sci., 10: 2161-2167.