Efficient Montgomery Modular Multiplication Algorithm Using Complement and Partition Techniques

Chia-Long Wu
Department of Aviation & Communication Electronics, Chinese Air Force Institute of Technology, Kaohsiung 820, Taiwan, R.O.C.
E-mail: chialongwu@seed.net.tw

Der-Chyuan Lou* and Te-Jen Chang*
***Department of Electrical Engineering, Chung Cheng Institute of Technology, National Defense University, Tahsi, Taoyuan 33509, Taiwan, R.O.C.**
E-mail: dclou@ccit.edu.tw

Abstract

High-speed, area-efficient, and low-power Montgomery modular multipliers for RSA algorithm have been developed for digital signature and user authentication in high-speed network systems and smart cards. In this paper, we present an efficient algorithm for modular multiplication by performing complements on the multiplicand and partitioning the key size of the multiplier. By applying the modular arithmetic and complement technique to the Montgomery algorithm, the proposed algorithm can efficiently evaluate modular exponentiation for RSA cryptosystem. The computational complexity of the proposed algorithm is $\frac{k}{4}+\frac{1}{2}$ multiplications, which is less than $\frac{k}{2}+1$ multiplications in Lee-Jeong-Kwon algorithm, where k is the bit-length of the multiplicand and the multiplier.

Keywords: Cryptography, modular arithmetic, key partition, public-key cryptosystem, modular multiplication.

摘要

針對 RSA 公開金鑰演算法演進而言，正朝向發展出快速、有效的使用面積、以及低功率的蒙哥馬利模乘法器，可應用在安全交易系統中智慧卡的數位簽章與使用者認證作業上。本論文中，我們提出一個有效地可節省模乘法數的演算法，本演算法是在被乘數利用補數方法，乘數利用分割金鑰大小的方法。運由模數運算以及補數方法在蒙哥馬利演算法上，則針對 RSA 密碼系統而言，所提出的方法可以有效地解決模指數的運算。我們提出的時間複雜度為 $\frac{k}{4}+\frac{1}{2}$ 乘法量，Lee、Jeong 與 Kwon 三位學者所提出的演算法則需要 $\frac{k}{2}+1$ 乘法量，其中 k 為被乘數與乘數的位元長度。

關鍵詞：密碼學、模運算、金鑰分割、公開金鑰密碼系統、模乘法。

1. Introduction

The computation of the modular exponentiation is a fundamental and important arithmetic operation in many scientific investigations, especially in the area of cryptology. The RSA (Rivest Shamir Adleman) [23] is one of the most widely used public-key cryptosystems. The encryption function is based on modular exponentiation with key size of 1024 or 2048 bits, and the modular multiplication is one of the major computation methods in cryptography systems [18]. The performance of the modular multiplication is the core arithmetic of the RSA cryptosystem comparing with other cryptography systems (e.g., international data encryption algorithm [12, 24], Diffie-Hellman key exchange [28]). The paper presents a faster technique for modular multiplication based on performing the complements on the multiplicand and partitioning the key size of the multiplier. The performance of the proposed algorithm is nearly 2 times faster than Lee-Jeong-Kwon algorithm [29].

The rest of this paper is organized as follows. Some characteristics of the modular arithmetic [9] and the complement technique are described in Section 2. In Section 3, we introduce the general integer multiplications, the Montgomery algorithm, the Jeong-Burleson algorithm, and the Lee-Jeong-Kwon algorithm. The proposed algorithm is described in Section 4. Finally, we put our conclusions for the proposed algorithm and state some future works.

2. Preliminaries

2.1 Modular Arithmetic

There are several moduli m_1, m_2, …, m_r that contain no common factors, and to work indirectly with residues u mod m_1, u mod m_2, …, u mod m_r, instead of directly with the number u.

We define the following notations:

$u_1 \equiv u \bmod m_1$, $u_2 \equiv u \bmod m_2$, …, $u_r \equiv u \bmod m_r$;

$v_1 \equiv v \bmod m_1$, $v_2 \equiv v \bmod m_2$, …,

$$v_r \equiv v \bmod m_r . \quad (1)$$

This is a consequence of the "Chinese Remainder Theorem" stated below. We may therefore regard (u_1, u_2, …, u_r) as a new type of internal computer representation, a "modular representation" of the integer u. We can easily compute (u_1, u_2, …, u_r) and (v_1, v_2, …, v_r) in Equation (1) from an integer numbers u and v by means of division.

The characteristics of a modular representation like addition, subtraction, and multiplication as follows.

$$(u_1,u_2,...,u_r)+(v_1,v_2,...,v_r)=((u_1+v_1) \bmod m_1,...,(u_r+v_r) \bmod m_r) \quad (2)$$

$$(u_1,u_2,...,u_r)-(v_1,v_2,...,v_r)=((u_1-v_1) \bmod m_1,...,(u_r-v_r) \bmod m_r) \quad (3)$$

$$(u_1,u_2,...,u_r)*(v_1,v_2,...,v_r)=((u_1*v_1) \bmod m_1,...,(u_r*v_r) \bmod m_r) \quad (4)$$

According to the above equations, we can derive the following equation:

$(u v) \bmod m_j = (u \bmod m_j)(v \bmod m_j) \bmod m_j$

for each modulus m_j [3, 10, 11, 14, 17, 26].

The processes of addition, subtraction, and multiplication using Equation (1), Equation (2), Equation (3), and Equation (4) are called residue arithmetic or modular arithmetic.

2.2 Complement Technique

Let A be an integer with the binary representation, indicated as $A_{k-1}A_{k-2}...A_0$, with the leftmost bit is the most significant one. It is easy to know that the following equation holds [4]:

$$A=(10...0)_{(k+1)bits}-\overline{A}-1, \quad (5)$$

where $\overline{A}=\overline{A_{k-1}}\,\overline{A_{k-2}}...\overline{A_0}$, and $\overline{a_i}=0$ if $a_i=1$; $\overline{a_i}=1$ if $a_i=0$, for i = 0, 1, …, k-1.

From Equation (5), the multiplication of two integers A and B can be expressed as [4]:

$$A*B=A*[(10...0)_{(k+1)bits}-\overline{B}-1] \quad (6)$$

Here we describe how to speed up A*B efficiently in the advantage of performing complements. We first check the Hamming weight, which is the number of 1's in the bit-string representation and indicated as Ham(A) of A. If the Ham(A) is greater than $\frac{k}{2}$, where k is the bit-length of A, we can employ Equation (6) to speed up the numbers of the multiplication. When Ham(A) is greater than $\frac{k}{2}$, then the Hamming weight of its complement will be less than $\frac{k}{2}$. Most importantly, if the possible Hamming weight of a multiplier in average case is $\frac{k}{2}$, then that of its complement will be $\frac{k}{4}$.

Let us show one example using Equation (5) as follows. If we assume A = $(10101010)_2$, then $\overline{A}$ = $(01010101)_2$, so we can obtain A = $(100000000)_2$ – $(01010101)_2$ – 1.

3. Montgomery Algorithm and its Improved Versions

There are two revised versions of Montgomery algorithms for speeding up both the computation of multiplications and exponentiations. Firstly, we depict the general integer multiplications in Section 3.1 and the Montgomery algorithm in Section 3.2. Then, we will describe two famous improved versions of Montgomery algorithms in Section 3.3 and Section 3.4, respectively.

3.1 General Integer Multiplications

For the general integer multiplication $S_1 = A*B$, when the key size is k-bits, if we partition the input B by half, it can be expressed as:

$$B = B_H * 2^{\frac{k}{2}} + B_L . \quad (7)$$

Then, S_1 can be computed as [29]:

$$S_1 = A * B_H * 2^{\frac{k}{2}} + A * B_L . \quad (8)$$

Example 1

Let A = $(1000000000)_2$ = $(512)_{10}$, B = $(1010110100)_2$ = $(692)_{10}$, N = 7, k = 10, Evaluate $S_1 = A*B$.

Here, B_H = $(10101)_2$ = $(21)_{10}$ and B_L = $(10100)_2$ = $(20)_{10}$ from Equation (7).

Then, as in Figure 1, $S_1 = A * B = (A * B_H * 2^{\frac{k}{2}} + A * B_L)$ = 354,304.

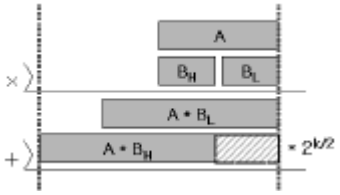

Figure 1. General integer multiplication (*AB*).**

3.2 Montgomery Algorithm

For the Montgomery multiplication $S_2 = A*B*2^{-k} \bmod N$, when the key size is k bits, if we still partition input B by half, it can be also expressed as Equation (7). Then S_2 can be computed as:

$$S_2 = (A * B_H * 2^{\frac{k}{2}} + A * B_L) * 2^{-k} \bmod N$$

$$= (A * B_H \bmod N + A * B_L * 2^{\frac{-k}{2}} \bmod N) * 2^{\frac{-k}{2}} \bmod N \quad (9)$$

By Equation (9), we see that two terms in parenthesis have weight difference of $2^{\frac{k}{2}}$ [29]. In Figure 2, it presents the case that we use Montgomery algorithm on both partitioned parts simultaneously. In partitioned Montgomery method, the results of each part are $A*B_H*2^{\frac{k}{2}} \bmod N$ and $A*B_L*2^{\frac{-k}{2}} \bmod N$,

and the final modular addition is required with $A * B_H \bmod N$ and $\frac{k}{2}$ bits right-shifted version of $A * B_L * 2^{\frac{-k}{2}} \bmod N$, i.e. $A * B_L * 2^{-k} \bmod N$, to get the final result of $A * B * 2^{-k} \bmod N$ [29].

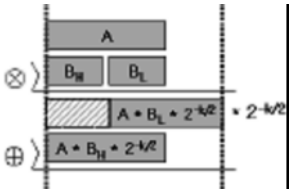

Figure 2. Montgomery algorithm ($A*B*2^{-k} \bmod N$).

Example 2
Let $A = (1000000000)_2 = (512)_{10}$, $B = (1010110100)_2 = (692)_{10}$, $N = 7$, $k = 10$, Evaluate $S_2 = A * B * 2^{-k} \bmod N$.
Here, $B_H = (10101)_2 = (21)_{10}$ and $B_L = (10100)_2 = (20)_{10}$ from Equation (7).
Then, as in Figure 2,

$$S_2 = A*B*2^{-k} \bmod N = (A*B_H*2^{\frac{k}{2}} + A*B_L)*2^{-k} \bmod N = 3.$$

3.3 Jeong-Burleson Algorithm

In Figure 3, we show Jeong-Burleson algorithm [9] on both partitioned parts simultaneously. The results of each part are $A * B_L \bmod N$ and $A * B_H \bmod N$, and the final modular addition is required with $A * B_L \bmod N$ and $\frac{k}{2}$ bits left-shifted version of $A * B_H \bmod N$, i.e. $A * B_H * 2^{\frac{k}{2}} \bmod N$, to get the final result S_3 [29], where

$$S_3 = A * B \bmod N. \quad (10)$$

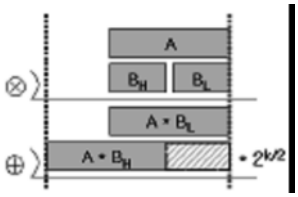

Figure 3. Jeong-Burleson algorithm ($A*B \bmod N$).

Example 3
Let $A = (1000000000)_2 = (512)_{10}$, $B = (1010110100)_2 = (692)_{10}$, $N = 7$, $k = 10$, Evaluate $S_3 = A * B \bmod N$.
Here, $B_H = (10101)_2 = (21)_{10}$ and $B_L = (10100)_2 = (20)_{10}$ from Equation (7).
Then, as in Figure 3,

$$S_3 = A * B \bmod N = (A * B_H * 2^{\frac{k}{2}} + A * B_L) \bmod N$$
$$= (A * B_H * 2^{\frac{k}{2}} \bmod N + A * B_L \bmod N) \bmod N = 6.$$

3.4 Lee-Jeong-Kwon Algorithm

Lee-Jeong-Kwon algorithm computes modular multiplication by using the upper part of the multiplication through the Jeong-Burleson algorithm and the lower part through the Montgomery algorithm. The position rearrangement can be avoided by using Lee-Jeong-Kwon algorithm because the output of Jeong-Burleson algorithm ($A * B_H \bmod N$) is a correct answer and that of Montgomery algorithm $A * B_L * 2^{\frac{-k}{2}} \bmod N$ is a $\frac{k}{2}$-bits right-shifted one, which results in weight difference of $2^{\frac{k}{2}}$ between the outputs from each part. Therefore the operand width for a final modular addition is bounded to k bits, then the result S_4 is obtained, where

$$S_4 = A * B * 2^{\frac{-k}{2}} \bmod N. \quad (11)$$

Example 4
Let $A = (1000000000)_2 = (512)_{10}$, $B = (1010110100)_2 = (692)_{10}$, $N = 7$, $k = 10$, Evaluate $S_4 = A * B * 2^{\frac{-k}{2}} \bmod N$.
Here, $B_H = (10101)_2 = (21)_{20}$ and $B_L = (10100)_2 = (20)_{20}$ from Equation (7).
Then, as in Figure 4,

$$S_4 = A*B*2^{\frac{-k}{2}} \bmod N = (A*B_L*2^{\frac{-k}{2}} \bmod N) + (A*B_H \bmod N) = 5.$$

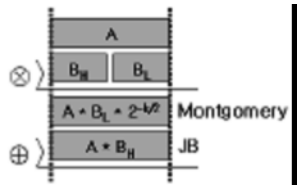

Figure 4. Lee-Jeong-Kwon algorithm ($A*B*2^{\frac{-k}{2}} \bmod N$).

4. The Proposed Algorithm

In Section 3, we describe general integer multiplication, Montgomery algorithm, Jeong-Burleson algorithm, and Lee-Jeong-Kwon algorithm. For the k-bit key size, both Montgomery and Jeong-Burleson algorithm require k additions [29] But the Lee-Jeong- Kwon algorithm only requires $\frac{k}{2}+1$ additions.

We adopt the complement technique to Lee-Jeong-Kwon algorithm to efficiently evaluate modular multiplications. Here we assume there are k bits for the bit-length of both A and B. The modular arithmetic of $A*B*2^{\frac{-k}{2}} \bmod N$ in Equation (11) should be considered by combining Lee-Jeong-Kwon algorithm and complement technique. So we can calculate the modular arithmetic in two cases as follows.

Case 1: Ham (A) > $\frac{k}{2}$, and

Case 2: Ham (A) $\leqq \frac{k}{2}$.

Let the computational complexity of the modular multiplication "$A*B*2^{\frac{-k}{2}} \bmod N$" is the number of iterations for the modular multiplication "$A*B*2^{\frac{-k}{2}} \bmod N$". Then the overall computational complexity of the modular arithmetic of "$A*B*2^{\frac{-k}{2}} \bmod N$" in the proposed algorithm can be depicted as follows [26].

The number of iterations for the modular multiplication "$A*B*2^{\frac{-k}{2}} \bmod N$"
= $\frac{1}{2}$ * (the number of iterations for modular multiplication in Case 1)
+ $\frac{1}{2}$ * (the number of iterations for modular multiplication in Case 2). (12)

The first and the second items "$\frac{1}{2}$" in Equation (12) are described as the probabilities of Ham (A)> $\frac{k}{2}$ and Ham (A)$\leqq \frac{k}{2}$.

Now we introduce the computational complexities of Case 1 and Case 2 respectively.

- Case 1: Ham(A) > $\frac{k}{2}$

  When Ham(A) > $\frac{k}{2}$ in Case 1, we take 1's complement for A. In other words, it can be represented by Ham($\overline{A}$) ≤ $\frac{k}{2}$. We can replace the pattern for $A*B_L*2^{\frac{-k}{2}} \bmod N$ with $[(10\ldots0)_{(k+1)\,bits} - \overline{A} - 1]*B_L*2^{\frac{-k}{2}} \bmod N$. We can also replace the pattern for $A*B_H \bmod N$ with $[(10\ldots0)_{(k+1)bits} - \overline{A} - 1]*B_H \bmod N$. So the number of iterations for a modular multiplication in Case 1 is $\frac{1}{2}*(\frac{k}{2}+1) = \frac{k}{4}+\frac{1}{2}$.
- Case 2: Ham(A) ≤ $\frac{k}{2}$

  When Ham(A)≤ $\frac{k}{2}$ in Case 2, we adopt the form "$A*B_L*2^{\frac{-k}{2}} \bmod N$" and "$A*B_H \bmod N$". So the number of iteration for a modular multiplication in Case 2 is $\frac{1}{2}*(\frac{k}{2}+1) = \frac{k}{4}+\frac{1}{2}$.

  From the Case 1 and Case 2 depicted as above, we can therefore get the overall computational complexity of the modular arithmetic $A*B_L*2^{\frac{-k}{2}} \bmod N$ from Equation (12):

$$\frac{1}{2}*(\frac{k}{4}+\frac{1}{2})+\frac{1}{2}*(\frac{k}{4}+\frac{1}{2}) = \frac{k}{4}+\frac{1}{2}. \quad (13)$$

Therefore, the speedup ratio comparing Lee-Jeong-Kwon algorithm with the proposed algorithm can be calculated as follows:

$$\text{Speedup ratio} = \frac{\frac{k}{2}+1}{(\frac{k}{2}+1)-(\frac{k}{4}+\frac{1}{2})} = 2. \quad (14)$$

5. Conclusions and future works

In this paper, we propose a new method to fast evaluate modular multiplications, which combine modular arithmetic, a complement technique, and Lee-Jeong-Kwon algorithm by performing complements on the multiplicand and partitioning the key size of the multiplier. The computational complexity of the proposed algorithm is $\frac{k}{4}+\frac{1}{2}$, which is less than $\frac{k}{2}+1$ in Lee-Jeong-Kwon algorithm [29], where k is the bit-length of the multiplicand and the multiplier, i.e., the proposed modular multiplication algorithm has speedup ratio of 2. We can efficiently speed up the overall performance of the modular multiplication.

Recently, some new techniques are studied and proposed to fast evaluate modular multiplications such as the unified-multipliers method [25, 30], elliptic curve encrypt method [2], and improved Montgomery algorithm [5, 7, 16, 22]. In the future, we will try to properly study the combination of techniques mentioned above in binary multiplication to further decrease the computational complexity for cryptographic applications [4, 5, 8] such as the smart card and key exchange schemes.

References

[1] A. B. Premkumar, E. L. Ang, and E. M.-K. Lai, "Improved memoryless RNS forward converter based on the periodicity of residues," IEEE Transactions on Circuits and Systems II: Express Briefs, Vol. 53, No. 2, pp. 133-137, Feb. 2006.

[2] A. F. Tenca and C. K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," IEEE Transactions on Computers, Vol. 52, No. 9, pp.1215-1221, Sep. 2003.

[3] A. F. Tenca, E. Savas, and C. K. Koc, "A Design Framework for Scalable and Unified Multipliers in GF(p) and GF(2^m)," International Journal of Computer Research, Vol. 13, No. 1, pp. 68-83, 2004.

[4] A. Z. Alkar and R. Sonmez, "A Hardware Version of the RSA Using the Montgomery's Algorithm with Systolic Arrays," Integration, the VLSI Journal, Vol. 38, No. 2, pp. 299-307, Dec. 2004.

[5] C. Heuberger and H. Prodinger, "Carry Propagation in Signed Digit Representations," European Journal of Combinatorics, Vol. 24, No. 3, pp. 293-320, April 2003.

[6] C.-C. Chang, Y.-T. Kuo, and C.-H. Lin, "Fast

Algorithms for Common-Multiplicand Multiplication and Exponentiation by Performing Complements," IEEE Proceedings of the 17th International Conference on Advanced Information Networking and Applications, pp. 807-811, 2003.

[7] C.-L. Wu, D.-C. Lou, and T.-J. Chang, "An Efficient Exponentiation Algorithm for Cryptographic Applications," Informatica, Vol. 16, No. 3, pp. 449-468, April 2005.

[8] D. E. Knuth, The Art of Computer Programming: Semi-Numerical Algorithms, Vol. 2, 3rd Edition, Addition-Wesley, 1997.

[9] D.-C. Lou and C.-L. Wu, "Parallel Modular Exponentiation Using Signed-Digit-Folding Technique," Informatica, Vol. 28, No. 2, pp. 197-205, July 2004.

[10] D.-C. Lou, C.-L. Wu, and C.-Y. Chen, "Fast Exponentiation by Folding the Signed-Digit Exponent in Half," International Journal of Computer Mathematics, Vol. 80, No. 10, pp. 1251-1259, Oct. 2003.

[11] F. Huang and Z. H. Guan, "A Modified Method of a Class of Recently Presented Cryptosystems," Chaos, Solitons and Fractals, Vol. 23, No. 5, pp. 1893-1899, March 2005.

[12] F. J. Taylor, "Residue Arithmetic: A Tutorial with Examples," IEEE Computer Magazine, Vol. 17, No. 5, pp. 50-62, May 1984.

[13] F. M. Dias, A. Antunes, J. Vieira, and A. Mota, "A sliding window solution for the on-line implementation of the Levenberg–Marquardt algorithm," Engineering Applications of Artificial Intelligence, Vol. 19, No. 1, pp. 1-7, Feb 2006.

[14] I. Koren, Computer Arithmetic, 2nd Edition, A. K. Peters, Natick, MA, 2002.

[15] K. Samoa, O. Semay, and T. Takagi, "Analysis of fractional window recoding methods and their application to elliptic curve cryptosystems," IEEE Transactions on Computers, Vol. 55, No. 1, pp.48-57, Jan. 2006.

[16] K. Takahiro and K. Yasumasa, "An Efficient Implementation of Parallel Eigenvalue Computation for Massively Parallel Processing," Parallel Computing, Vol. 27, No. 14, pp. 1831-1845, Dec. 2001.

[17] M. Joye and S.-M. Yen, "Optimal Left-to-Right Binary Signed-Digit Recoding," IEEE Transactions on Computers, Vol. 49, No. 7, pp. 740-748, July 2000.

[18] M. E. Kaihara and N. Takagi, "A Hardware Algorithm for Modular Multiplication/ Division", IEEE Transactions on Computers, Vol. 54, No. 1, pp. 12-21, Jan. 2005.

[19] N. A. S. Alwan, "A Fully Pipelined Systolic Array for Sinusoidal Sequence Generation," IEEE Transactions on Computers, Vol. 55, No. 5, pp.636-639, May 2006.

[20] N. Nedjah and L. M. Mourelle, "Three Hardware Architectures for the Binary Modular Exponentiation: Sequential, Parallel, and Systolic," IEEE Transactions on Vol. 53, No. 3, pp. 627-633, March 2006.

[21] N. Nedjah and M. M. Luiza, "A Review of Modular Multiplication Methods and Respective Hardware Implementations," INFORMATICA, Vol. 30, No. 1, pp. 111-129, 2006.

[22] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM, Vol. 21, No. 2, pp. 120-126, February 1978.

[23] R. Zimmermann, A. Curiger, H. Bonnenberg, H. Kaeslin, N. Felber, and W. Fichtner, "A 177 Mb/s VLSI Implementation of International Data Encryption Algorithm," IEEE Journal Solid-State Circuits , Vol. 29, No. 3, pp. 303-307, March 1994.

[24] S. Masui, K. Mukaida, M. Takenaka, and N. Torii, "Design Optimization of a High-Speed, Area-Efficient and Low-Power Montgomery Modular Multiplier for RSA Algorithm," IEICE Transactions on Electronics, Vol. 88, No. 4, pp. 576-581, April 2005.

[25] S.-Y. Lee, Y.-J. Jeong, and O.-J. Kwon, "A Faster Modular Multiplication Based on Key Size Partitioning for RSA Public-Key Cryptosystem," IEICE Transactions on Applications of Information Security Techniques, Vol. E85-D, No. 4, pp. 789-791, April 2002.

[26] T.-S. Chen, "A Threshold Signature Scheme Based on the Elliptic Curve Cryptosystem," Applied Mathematics and Computation, Vol. 162, No. 3, pp. 1119-1134, March 25, 2005.

[27] W. Diffie and M. E. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory, Vol. IT-22, No. 6, pp. 644-654, Nov. 1976.

[28] X. Lai and J. L. Massey, "A Proposal for a New Block Encryption Standard," in EUROCRYPT '90, Arahus, Denmark, May 1990.

[29] Y.-J. Jeong and W. Burleson "VLSI Array Algorithms and Architecture for RSA Modular Multiplication," IEEE Transactions on VLSI Systems, Vol. 5, No. 2, pp. 211-217, 1997.

[30] Y. Eslami, A. Sheikholeslami, P. G. Gulak, S. Masui, and K. Mukaida, "An Area-Efficient Universal Cryptography Processor for Smart Cards," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, Vol. 14, No. 1, pp. 43-56, Jan. 2006.