A Division Algorithm Using Bisection Method in Residue Number System

[1*]Chin-Chen Chang and [2]Jen-Ho Yang

Abstract

Residue Number System (RNS) has computational advantages for large integer arithmetic. It provides the benefits of parallel, carry-free, and high-speed arithmetic in addition, subtraction, and multiplication. However, overflow detection, sign detection, magnitude detection, and division are time-consuming operations in RNS. The most interesting one of the above operations is division, and many related researches have been proposed to make it more efficient. Hiasat and Abdel-Aty-Zohdy proposed a high-speed division algorithm using the highest power comparison to speed up division in RNS. Their algorithm evaluates the quotient according to the highest powers of 2 in the dividend and the divisor. Nevertheless, the evaluated quotient is underestimated such that there are redundant execution rounds in their algorithm. Thus, Yang et al. proposed a division algorithm in RNS using parity checking technique in 2004. In this algorithm, the evaluated quotient is estimated precisely such that the actual quotient can be found quickly. However, computing the highest powers of 2 in the dividend and the divisor is time-consuming in RNS, and this computation must be performed in each execution round of Yang et al.'s algorithm. Consequently, we propose the bisection method in RNS to design a division algorithm in the paper. Our new algorithm uses the bisection method in RNS to find the quotient in a possible interval efficiently. Compared with Yang et al.'s algorithm, our algorithm has less execution rounds and greatly reduces the times of the highest-power computation in RNS.

Keywords: Residue Number System, division, bisection method

*Corresponding Author: Chin-Chen Chang
(E-mail: alan3c@gmail.com).
[1] Department of Computer Science and Information Engineering, Asia University, Taichung, 413, Taiwan; Department of Information Engineering and Computer Science, Feng Chia University, Taichung, 407, Taiwan
[2]Department of Multimedia and Mobile Commerce, Kainan University, No. 1, Kannan Rd., Luzhu, Taoyuan County, 33857, Taiwan

1. Introduction

Nowadays, numerous number systems have been utilized to make computers more and more powerful. The most popular one of these number systems is Residue Number System (RNS) because it has many advantages of computing large numbers in computers. In RNS, a number is represented by the residues of all moduli, and the arithmetic can be performed on each modulus independently. Thus, RNS offers the properties of parallel, carry-free, and high-speed arithmetic [1]. However, the overflow detection, sign detection, magnitude detection, and division are time-consuming operations in RNS. The most interesting research topic of them is division because it has many applications such as modular operations. Thus, several algorithms have been proposed to solve division problem in RNS [1-9]. However, all proposed algorithms have the drawbacks of long execution time and large hardware requirements because Mixed-Radix Conversion (MRC) and Chinese Remainder Theorem (CRT) are used. Instead of employing MRC and CRT, Hiasat and Abdel-Aty-Zohdy used the highest power comparison between the dividend and the divisor to design a division algorithm in RNS [10]. The execution time and hardware requirements in their algorithm are less than those in other division algorithms. Their algorithm computes the evaluated quotient according to the highest powers of the dividend and the divisor, and obtains the actual quotient by computing the sum of all evaluated quotients until the product of the divisor and the evaluated quotient is less than the dividend. However, the evaluated quotient is underestimated in their algorithm such that the number of execution rounds increases. Thus, Yang et al. proposed a division algorithm in RNS using the parity checking technique in 2004 [11]. In the algorithm, the evaluated quotient is estimated precisely such that approximating the actual quotient by adding the evaluated quotients is two times faster than that in Hisat and Abdel-Aty-Zohdy's algorithm. However, computing the highest power of 2 of a number is time-consuming in RNS, and Yang et al.'s algorithm needs to perform this computation in each execution round. This extra load is the bottleneck of this algorithm.

As a result, a bisection method in RNS is proposed to design a division algorithm in this paper. A new algorithm in this research uses the bisection method to find the quotient in a probable interval efficiently, and the algorithm only needs to compute the highest-power comparison in RNS in the first round. This improvement greatly decreases the execution time of each round in our algorithm. Furthermore, Yang et al.'s algorithm needs to determine the magnitudes of the dividend and the divisor by using the parity checking technique twice in each round. However, using the bisection method in RNS, our algorithm only performs the parity checking technique once to determine the magnitude of the residue in each execution round. Moreover, the proposed division algorithm can be efficiently applied to many applications, such as image processing [12], cryptosystems [13], and modular exponentiation arithmetic [14].

The remainder of this paper is organized as follows. A review of popular image compression algorithms that support the ROI capability is given in Section II. In Section III, we shall address our proposed multiple-ROI image compression scheme. Then the simulation results and a further discussion are provided in Section IV. Finally, a brief conclusion is drawn in Section V.

2. Hiasat and Abdel-Aty-Zohdy's division algorithm

Given a set of relatively prime moduli $B=\{m_1, m_2, \ldots, m_n\}$, where $\gcd(m_i, m_j)=1$, $i \neq j$, we compute $M=\prod_{i=1}^{n} m_i$. Here, B and M are called the base and the range of RNS. In RNS, any number X less than the range M can be denoted as a vector $X=(x_1, x_2, \ldots, x_n)$, where $x_i = |X|_{m_i} = X \bmod m_i$. According to Chinese Remainder Theorem, any X less than M has only one RNS representation. We assume that two integers $x, y \in [0, M)$ in RNS are $X=(x_1, x_2, \ldots, x_n)$ and $Y=(y_1, y_2, \ldots, y_n)$, respectively, and compute $Z = X \otimes Y = (z_1, z_2, \ldots, z_n)$, where $\otimes$ can be addition, subtraction or multiplication in RNS. Then Z can be obtained by computing $X \otimes Y = (|x_1 \otimes y_1|_{m_1}, |x_2 \otimes y_2|_{m_2}, \ldots, |x_n \otimes y_n|_{m_n}) = (z_1, z_2, \ldots, z_n)$. That is, the computation can be performed concurrently and independently on each residue in RNS. Therefore, addition, subtraction, and multiplication of two large numbers, individually converted into RNS representations, can be efficiently computed because of the properties of parallelism and no carries.

Assume that we want to compute $Q = \left\lfloor \frac{X}{Y} \right\rfloor$ in RNS, where X, Y, and Q are positive integers. In Hiasat and Abdel-Aty-Zohdy's algorithm, $h(I)$ can be computed the highest power of 2 in the variable I represented by RNS. According to [10], $h(I)$ is implemented by a proper combinational circuit like a priority encoder and an n-operand binary adder, where n is the number of the moduli in RNS. Hiasat and Abdel-Aty-Zohdy's algorithm is shown as follows.

Step 1: Set the quotient $Q=0$.

Step 2: Compute $j=h(X)$ and $k=h(Y)$, where j and k are the highest powers of 2 in X and Y, respectively.

Step 3: If $j>k$, then compute $Q' = Q + 2^{j-k-1}$, $X' = X - 2^{j-k-1} * Y$, $Q=Q'$, $X=X'$. Go to Step 2.

Step 4: If $j=k$, then compute $X' = X - Y$ and $j' = h(X')$. If $j' < j$ then $Q=Q+1$. Otherwise, Q is unaltered. End the procedure.

Step 5: If $j<k$, then Q is unaltered. End the procedure.

Hiasat and Abdel-Aty-Zohdy's algorithm utilizes addition, subtraction, and multiplication according to the highest powers of 2 in dividend and divisor to achieve division in RNS. Besides, the algorithm performs division efficiently by avoiding the magnitude detection and overflow detection in RNS.

3. Yang et al.'s algorithm

Assume that we want to compute $Q = \left[\frac{X}{Y}\right]$ in RNS, where X, Y, and Q are integers. Note that $h(I)$ denotes the highest power of 2 in the variable I, where I is RNS representation of an integer. For example, 10 = (0, 1, 0) and (–7) = (1, 2, 3) with the moduli (2, 3, 5) in RNS, then we have $h((0,1,0)) = 3$ and $h((1,2,3)) = 2$. Besides, $S(X)$ is defined as the sign of variable X. If $S(X) = S(Y)$, it denotes that X and Y are with the same sign. In addition, the parity checking technique [8] is used in this algorithm to determine the sign of a number in RNS. Unlike other sign detection algorithms in RNS using MRC or CRT, the parity checking technique can easily obtain the sign of a number in RNS by checking its parity and looking up a table. This technique makes the sign detection and the magnitude comparison in RNS more efficient. The details of the parity checking technique are available in [8]. Yang et al.'s algorithm is shown in the following.

Step 1: Set the quotient $Q = 0$ and $c = 0$.

Step 2: Compute $j = h(X)$ and $k = h(Y)$, where j and k are the highest power of 2 in X and Y, respectively. Then, check the signs of X and Y by parity checking. According to the relationships between j and k, we perform one of the following three steps.

Step 3: If $j > k$, we perform the following operations:
Set $c = 1$.
If $S(X) = S(Y)$, then we compute $Q' = Q + 2^{j-k}$, $X' = X - 2^{j-k} * Y$, $Q = Q'$, $X = X'$; otherwise, we compute $Q' = Q - 2^{j-k}$, $X' = X + 2^{j-k} * Y$, $Q = Q'$, and $X = X'$. Go to Step 2.

Step 4: If $j = k$, we perform the following operations.
If $S(X) = S(Y)$ and $c = 1$, then we compute $X' = X - Y$ and $j' = h(X')$. If $j' < j$ then we compute $Q = Q + 1$. Otherwise, Q is unchanged. End procedure.
If $S(X) \neq S(Y)$ and $c = 1$, then we compute $X' = X + Y$ and check the sign of X' by parity checking. If $X' > 0$, then we compute $Q = Q - 1$. Otherwise, we set $Q = Q - 2$. End procedure.
If $S(X) \neq S(Y)$ and $c = 0$, then we set $Q = -1$. End procedure.

Step 5: If $j < k$, we perform the following operations.
If $c = 0$, then Q is unchanged. End procedure.
Otherwise, we compute $Q = Q - 1$. End procedure.

In this algorithm, the evaluated quotient is 2^{j-k} which is larger than 2^{j-k-1} in Hiasat and Abdel-Aty-Zohdy's algorithm such that the actual quotient can be obtained quickly in Yang et al.'s algorithm. Compared with Hiasat and Abdel-Aty-Zohdy's algorithm, Yang et al.'s algorithm reduces the number of execution rounds by 50%.

4. The proposed algorithm

After observing Yang et al.'s algorithm, we find $h(I)$ and the parity checking technique must be performed twice in each round of their algorithm. If we decrease the numbers of these two computations in each round, the execution time can be reduced. Thus, we propose a bisection method in RNS to accomplish this purpose. In the following, our algorithm is shown.

Input: (X, Y : are expressed in RNS)

Output: (Q : is expressed in RNS)

Step 1: Compute $j = h(X)$ and $k = h(Y)$, where j and k are the highest powers of 2 in X and Y , respectively.

Step 2: If $j = k$, then use the parity checking technique to compare X with Y . If $X > Y$, then set $Q = 1$. Otherwise, set $Q = 0$. End the program.

If $j < k$, then set $Q = 0$. End the program.

Step 3: Compute $Q_u = 2^{j+1-k}$ and $Q_l = 2^{j-k-1}$.

Step 4: Compute $Q' = Q_u +_{RNS} Q_l$. If Q' is odd, then set $Q' = Q' +_{RNS} (1, 1, \ldots, 1)$.

Step 5: Compute $Q = \frac{Q'}{2}$ and

$Z = X -_{RNS} Q \times_{RNS} Y$. Then, use the parity checking technique to determine the sign and the magnitude of Z.

Step 6: If $Z > Y > 0$, then set $Q_l = Q$. Go to Step 3.

Step 7: If $Z < 0$, then set $Q_u = Q$. Go to Step 3.

Step 8: If $Y > Z > 0$, then Q is the quotient. End the program

In our algorithm, we define $+_{RNS}$, $-_{RNS}$, and $\times_{RNS}$ as addition, subtraction, and multiplication in RNS, respectively. Besides, Q_u and Q_l are the upper and the lower bound of the actual quotient in our algorithm, respectively. In Step 4, we compute $Q = \frac{Q'}{2}$ to be the evaluated quotient in each round of our algorithm. If all chosen moduli in RNS are odd integers, then computing $\frac{Q'}{2}$ can be directly made in RNS without converting it into the binary or the decimal system. For example, assume that the multiplicative inverses of 2 modulo all moduli $(m_1, m_2, \ldots, m_n)$ in RNS are $(\left|\frac{1}{2}\right|_{m_1}, \left|\frac{1}{2}\right|_{m_2}, \ldots, \left|\frac{1}{2}\right|_{m_n})$, and the RNS representation of Q' is $(q_1, q_2, \ldots, q_n)$. Then, we have $\frac{Q'}{2}$ =

$$\left(\left|\left|\frac{1}{2}\right|_{m_1} \times q_1\right|_{m_1}, \left|\left|\frac{1}{2}\right|_{m_2} \times q_2\right|_{m_2}, \ldots, \left|\left|\frac{1}{2}\right|_{m_n} \times q_n\right|_{m_n}\right).$$

Moreover, the multiplicative inverses of 2 modulo all moduli in RNS can be precomputed to reduce the computation time.

5. Analyses

We simulate Hiasat and Abdel-Aty-Zohdy's algorithm, Yang et al.'s algorithm, and our algorithm in Visual C++ 6.0, and run the simulation programs for the values of X and Y both ranging from 1 to 30000. Table 1 shows the numbers of execution rounds in these three algorithms, and Table 2 shows the number of computing $h(I)$ and the times of performing the parity checking technique in these three algorithms. The analysis of the simulation results is shown as follows. In Hiasat and Abdel-Aty-Zohdy's algorithm, the evaluated quotient is the power of 2 (i.e. 2^{j-k-1}), and the actual quotient is formed by these evaluated quotients. For example, the actual quotient $Q = \left\lfloor \frac{258}{33} \right\rfloor = 7$ in Table 1. After running Hiasat and Abdel-Aty-Zohdy's algorithm, Q is formed by $2^2 + 2^0 + 2^0 + 2^0$. Note that even if 7 represented in the binary form is $(111)_2 = 2^2 + 2^1 + 2^0$, Q is probably formed by $2^2 + 2^0 + 2^0 + 2^0$ or another form in Hiasat and

Abdel-Aty-Zohdy's algorithm. This is because the evaluated quotient is underestimated in their algorithm such that 2^1 is formed by $2^0 + 2^0$ in the above example. In this case, their algorithm must execute two rounds to obtain 2^1. This property causes that there are some redundant rounds in their algorithm. Thus, it can be easily observed that the number of execution rounds in Hiasat and Abdel-Aty-Zohdy's algorithm depends on the components of the quotient Q represented in the binary system. Assume that representing Q_1 in the binary system requires $|\log Q|$ bits, then the number of execution rounds in Hiasat and Abdel-Aty-Zohdy's algorithm denoted as $rounds_{(1)}$ must fall in the range shown as Equation (1). This is because the maximum number of the binary digits used to form Q_1 is $2 \times |\log Q| - 1$.

$$1 \le rounds_{(1)} \le 2 \times |\log Q| - 1 \qquad (1)$$

In our algorithm, the range of the actual quotient Q is $2^{j-k-1} < Q_1 < 2^{j-k+1}$ and we use some techniques in RNS to find Q in this range. With another point of view, our algorithm can be regarded as performing the bisection method in RNS to find Q in the interval $(2^{j-k-1},\ 2^{j-k+1})$. Thus, the range of the number of execution rounds in our algorithm denoted as $rounds_{(2)}$ is shown as Equation (2).

$$1 \le rounds_{(2)} \le \log(2^{j-k+1} - 2^{j-k-1} - 1) \qquad (2)$$

To compare $rounds_{(1)}$ with $rounds_{(2)}$, we adjust Equations (1) and (2) in the following steps.

Since $2^{j-k-1} < Q < 2^{j-k+1}$, Equation (3) can be obtained from Equation (1).

$$1 \le rounds_{(1)} \le 2 \times \log(2^{j-k+1} - 1) - 1 \qquad (3)$$

For the convenience of the analysis, Equation (3) is rewritten as Equation (4).

$$1 \le rounds_{(1)} \le 2 \times \log(2^{j-k+1}) \qquad (4)$$

Because $\log(2^{j-k+1}) = \log(2^{j-k-1} \times 2^2) = \log(2^{j-k-1}) + \log 4$, Equation (5) can be obtained from Equation (4).

$$1 \le rounds_{(1)} \le 2 \times (\log(2^{j-k-1}) + \log 4) \qquad (5)$$

From Equation (2), Equation (6) is obtained as follows.

$$1 \le rounds_{(2)} \le \log(3 \times 2^{j-k-1} - 1) \qquad (6)$$

Again, Equation (6) can be rewritten as Equation (7).

$$1 \le rounds_{(2)} \le \log(3 \times (2^{j-k-1})) \qquad (7)$$

Because $\log(3 \times 2^{j-k-1}) = \log 3 + \log(2^{j-k-1})$, we obtain Equation (8) from Equation (7).

$$1 \le rounds_{(2)} \le \log(2^{j-k-1}) + \log 3 \qquad (8)$$

Compared Equation (5) with Equation (8), $rounds_{(1)}$ is the double of $rounds_{(2)}$. The analysis confirms that Yang et al.'s algorithm reduces the number of execution rounds by 50% and also by comparing with Hiasat and Abdel-Aty-Zohdy's algorithm.

As the same reason, the number of execution rounds in Yang et al.'s algorithm also depends on the components of the quotient Q represented in the binary system. Each binary digit of Q in Yang et al.'s algorithm can be obtained by executing one round because the evaluated quotient is set larger and more precise. Thus, assume that representing Q in the binary system requires $|\log Q|$ bits, then the number of execution rounds in Yang et al.'s algorithm, denoted as $rounds_{(3)}$, is shown as Equation (9).

$$1 \le rounds_{(3)} \le |\log Q| \qquad (9)$$

Using the same thinking steps from Equation (3) to Equation (5), Equation (10) can be obtained from Equation (9).

$$1 \le rounds_{(3)} \le \log(2^{j-k-1}) + \log 4 \qquad (10)$$

Compared with Equation (8) and Equation (10) denoted as $rounds_{(2)}$ is a little less than $rounds_{(3)}$. Table 4 also shows this fact.

On the other hand, computing $h(I)$ in RNS is time-consuming, but Hiasat and Abdel-Aty-Zohdy's algorithm and Yang et al.'s algorithm both needs to compute $h(I)$ in each round. However, Our algorithm only computes $h(I)$ in the first round. This property greatly reduces the execution time in each round of our algorithm. Although our algorithm needs to compute the evaluated quotient by $Q' = \frac{Q_u +_{RNS} Q_l}{2}$, computing Q' by looking up a table is faster than computing $h(I)$ in RNS. As the densities and speed of RAMs increase, the use of looking up tables is beneficial. Besides, our algorithm decreases the times of performing the parity checking technique in each round. This improvement also reduces the execution time in each round of our algorithm. Table 5 shows the numbers of computing $h(I)$ and the times of performing the parity checking techniques in Hiasat and Abdel-Aty-Zohdy's algorithm, Yang et al.'s algorithm, and our algorithm for computing $\left\lfloor \frac{X}{Y} \right\rfloor$, where X and Y are all ranging from 1 to 30000.

Table 1: The numbers of execution rounds for the simulations

	X= 1~10000, Y= 1~10000	X= 1~20000, Y= 1~20000	X= 1~30000, Y= 1~30000
Hiasat and Abdel-Aty-Zohdy's algorithm	182340293	729054181	1662132965
Yang et al.'s algorithm	88279189	359358946	829759637
The number of rounds of our algorithm	88179771	358913373	828905709
The percentage of the numbers of rounds	48.37%	49.23%	49.87%

According to Table 2, our algorithm greatly reduces the number of computing $h(I)$ and the times of performing the parity checking techniques by 46% and 50%, respectively. Consequently, our algorithm is much faster than Yang et al.'s algorithm even if the execution rounds in our algorithm are a little less than those in Yang et al.'s algorithm.

Table 2: The times of computing *h*(*I*) and the parity checking techniques

	Hiasat and Abdel-Aty-Zohdy's algorithm	Yang et al.'s algorithm	Our algorithm
Parity checking technique		3324265930	1662132965
$h(I)$	3324265930	3324265930	1800000000

6. Conclusions

In this paper, we propose a division algorithm using the bisection method in RNS. Assume that the highest powers of 2 in the dividend and divisor are 2^j and 2^k, respectively, then the quotient would fall into the interval $(2^{j-k+1}, 2^{j-k-1})$. Our algorithm skillfully finds the quotient in the interval without converting the numbers from RNS into the binary or the decimal system. Compared with Yang et al.'s algorithm, our algorithm reduces the number of the highest power computation $h(I)$ and the parity checking techniques in RNS by 46% and 50%, respectively. According to the simulation results and the analysis in Section 5, it is obvious that our algorithm is more efficient than Yang et al.'s algorithm.

References

[1] N. Szabo and R. Tanaka, Residue Arithmetic and Its Applications to Computer Technology, McGraw Hill, New York, 1967.

[2] E. Kinoshita, H. Kosako, and Y. Kojima, "General Division in the Symmetric Residue Number System," *IEEE Transactions on Computers*, Vol. 22, 1973, pp. 134-142.

[3] D. Banerji, T. Cheung, and V. Ganesan, "A High-speed Division Method in Residue Arithmetic," *Proceedings of 5th IEEE Symposium on Computer Arithmetic*, Michigan, USA, 1981, pp. 158-164.

[4] W. Chren Jr., "A New Residue Number Division Algorithm," *Computer and Mathematics with Applications*, Vol. 19, No. 7, 1990, pp. 13-29.

[1] D. Gamberger, "New Approach to Integer

Division in Residue Number System," *Proceedings of 10th Symposium on Computer Arithmetic*, Grenoble, France, 1991, pp. 84-91.

[2] Y. Kier, P. Cheney, and M. Tannenbaum, "Division and Overflow Detection in Residue Number Systems," *IRE Transactions on Electronic Computers*, Vol. 11, 1962, pp. 501-507.

[3] L. Lin, E. Leiss, and B. Mcinnis, "Division and Sign Detection Algorithm for Residue Number Systems," *Computer and Mathematics with Applications*, Vol. 10, 1984, pp. 331-342.

[4] M. Lu and J. S. Chiang, "A Novel Division Algorithm for the Residue Number System," *IEEE Transactions on Computers*, Vol. 41, No. 8, 1992, pp. 1026-1032.

[5] A. A. Hiasat, and H. S. Abdel-Aty-Zohdy, "A High-speed Division Algorithm for Residue Number System," *Proceedings of IEEE International Symposium on Circuits and Systems*, Vol. 3, Washington, USA, 1995, pp. 1996-1999.

[6] A. A. Hiasat, and H. S. Abdel-Aty-Zohdy, "Design and Implementation of an RNS Division Algorithm," *Proceedings of 13th IEEE Symposium on Computer Arithmetic*, California, USA, 1997, pp. 240-249.

[11] J. H. Yang, C. C. Chang, and C. Y. Chen, "A High-Speed Division Algorithm in Residue Number System Using Parity-Checking Technique," *International Journal of Computer Mathematics*, Vol.81, No. 6, pp. 775-780, 2004.

[12] D.K.Taleshmekaeil and A. Mousavi, "The Use of Residue Number System for Improving the Digital Image Processing," *Proceedings of 10th IEEE International Conference on Signal Processing*, Beijing, China, pp. 195-204, 2010.

[13] J. Adikari, V. S. Dimitrov, and L. Imbert, "Hybrid Binary-Ternary Number System for Elliptic Curve Cryptosystems," *IEEE Transactions on Computers,* Vol. 60, No. 2, pp.254-265, 2011.

[14] F. Gandino, F. Lamberti, P. Montuschi, and J. Bajard, "A General Approach for Improving RNS Montgomery Exponentiation Using Pre-processing," *Proceedings of 20th IEEE Symposium on Computer Arithmetic*, Tübingen, Germany, pp. 195-204, 2011.

Chin-Chen Chang received his Ph.D. degree in computer engineering from National Chiao Tung University. His first degree is Bachelor of Science in Applied Mathematics and master degree is Master of Science in computer and decision sciences. Both were awarded in National Tsing Hua University. Dr. Chang served in National Chung Cheng University from 1989 to 2005. His current title is Chair Professor in Department of Information Engineering and Computer Science, Feng Chia University, from Feb. 2005. Prior to joining Feng Chia University, Professor Chang was an associate professor in Chiao Tung University, professor in National Chung Hsing University, chair professor in National Chung Cheng University. He had also been Visiting Researcher and Visiting Scientist to Tokyo University and Kyoto University, Japan. During his service in Chung Cheng, Professor Chang served as Chairman of the Institute of Computer Science and Information Engineering, Dean of College of Engineering, Provost and then Acting President of Chung Cheng University and Director of Advisory Office in Ministry of Education, Taiwan. Professor Chang has won many research awards and honorary positions by and in prestigious organizations both nationally and internationally. He is currently a Fellow of IEEE and a Fellow of IEE, UK. And since his early years of career development, he consecutively won Outstanding Talent in Information Sciences of the R. O. C., AceR Dragon Award of the Ten Most Outstanding Talents, Outstanding Scholar Award of the R. O. C., Outstanding Engineering Professor Award of the R. O. C., Distinguished Research Awards of National Science Council of the R. O. C., Top Fifteen Scholars in Systems and Software Engineering of the Journal of Systems and Software, and so on. On numerous occasions, he was invited to serve as Visiting Professor, Chair Professor, Honorary Professor, Honorary Director, Honorary Chairman, Distinguished Alumnus, Distinguished Researcher, Research Fellow by universities and research institutes. His current research interests include database design, computer cryptography, image compression and data structures.

Jen-Ho Yang received the B.S. degree in computer science and information engineering from I-Shou University, Kaoshiung in 2002, and the Ph.D. degree in computer science and information engineering from National Chung Cheng University, Chiayi County in 2009. Since 2009, he has been an assistant professor with the Department of Multimedia and Mobile Commerce in Kainan University, Taoyuan. His current research interests include electronic commerce, information security, cryptography, authentication for wireless environments, digital right management, and fast modular multiplication algorithm.