A New Word-Parallel Bit-Serial Normal Basis Multiplier over $GF(2^m)$

Yong Suk Cho[1] and Jae Yeon Choi[2]

[1]*Department of Information & Communication Security, Youngdong University, Chung Buk, 370-701, Korea*
[2]*Department of Information & Communication, Namseoul University, Chung Nam, 331-707, Korea*
[1]*yscho@yd.ac.kr*, [2]*cjy@nsu.ac.kr*

Abstract

Hardware implementations of finite field arithmetic using normal basis are advantageous due to the fact that the squaring operation can be done at almost no cost. In this paper, a new word-parallel bit-serial finite field multiplier using normal basis is presented. The proposed architecture takes w clock cycles to compute the product bits, where the value for w, $1 \leq w \leq m$, can be arbitrarily selected by the designer to set the trade-off between area and speed. It has been shown that the proposed architecture has significantly lower complexity and critical path delay in comparison to the previously proposed architectures.

Keywords: *Finite field multiplier, Normal basis, Cryptography*

1. Introduction

Finite fields are the most commonly used arithmetical structures in cryptography and coding. Many algorithms in cryptographic and coding applications are defined in terms of finite field arithmetic operations [1, 2]. The elliptic curve cryptosystems and the Diffie-Hellman key exchange algorithm are important examples of such cryptographic applications [3]. Also, common error control codes such as Reed-Solomon and BCH codes are based on finite field theory [4].

Finite fields are algebraic fields with finite number of elements. These fields take the place of the familiar fields like the real numbers in cryptography and coding. Because they have finite number of elements, the operations on them cannot produce infinitely large results. Also, the finite field operations always produce exact results, not approximate results. Thus, they do not suffer from truncation errors like the floating point operations.

Addition and multiplication are two basic operations in the finite field $GF(2^m)$. Addition in $GF(2^m)$ is easily realized using m two-input XOR gates while multiplication is costly in terms of gate count and time delay. The other operations of finite fields, such as exponentiation, division, and inversion can be performed by repeated multiplications. As a result, there is a need to have fast multiplication architecture with low complexity.

One important factor that could greatly affect computation performance is the basis in which finite field elements are represented. The most commonly used bases include polynomial basis (PB) or standard basis and normal basis (NB). PB probably is the most popular basis and is efficient in both software and hardware implementations. On the other hand, NB has a distinct feature that its squaring operation in a binary field is free. Therefore, NB is especially attractive when performing exponentiation, where squaring operations are extensively required.

Hardware implementation of binary finite field multipliers can be categorized into three categories. First category is bit-serial multiplier [5, 6, 7]. A bit-serial multiplier takes m clock

cycles to finish one multiplication in a binary field of size m. The multipliers in this class are considered to be low power and taking small area of silicon. Their main disadvantage is their low multiplication speed for large field sizes. The second category is bit-parallel multiplier [8, 9]. A bit-parallel multiplier takes one clock cycle to finish the multiplication for any field size. These multipliers are not practical since they require large silicon area and are not economical to be made.

The third category is word level finite field multiplier which is the most commonly used in practice [10, 11, 12]. A word level multiplier takes w clock cycles, $1 \leq w \leq m$, to finish one multiplication operation in a binary field of size m. The value of w can be selected by designer to set the trade off between area and speed. Decreasing the value of w will result in faster and larger multipliers while increasing w will make smaller and slower multipliers.

In this paper a new word-parallel bit-serial finite field multiplier using a normal basis is presented. The proposed multiplier divides the data into the same length of word, uses bit-serial multiplier inside the word, and processes the multiplications using word-parallel method. The complexity of the proposed architecture is compared with those of the previously reported structures and it is shown that the proposed multiplier is better than that of the existing architectures.

The organization of this paper is as follows: Normal basis and bit-serial multiplication using this basis [6] are briefly reviewed in Section 2. In Section 3, a new word-parallel bit-serial multiplier using normal basis is proposed and architectural complexity comparison with similar proposals is presented. Finally some concluding remarks are given in Section 4.

2. The Bit-Serial Normal Basis Multiplier over $GF(2^m)$

Let a normal basis of $GF(2^m)$ over $GF(2)$ be

$$N = \{\beta,\ \beta^{2^1},\ \beta^{2^2} \cdots\cdots,\ \beta^{2^{m-1}}\}, \tag{1}$$

where $\beta \in GF(2^m)$. It is well known that there exists such a normal basis for any positive integer $m > 1$ [2]. Using such a basis, any field element $A \in GF(2^m)$ can be represented as a linear combination of the elements of N, *i.e.*,

$$A = \sum_{i=0}^{m-1} a_i \beta^{2^i} = (a_0, a_1, \cdots\cdots, a_{m-1}) \tag{2}$$

where $a_i \in GF(2),\ 0 \leq i \leq m-1$, are referred to as coordinates of A with respect to (w.r.t.) N. In hardware implementation, A^2 is almost free of cost and can be easily performed by right cyclic shifts, *i.e.*,

$$A^{2^i} = (a_{m-i}, a_{m-i+1}, \cdots\cdots, a_{m-i-1}). \tag{3}$$

However, multiplication is not as easy as squaring. Below we briefly review bit-serial normal basis multiplier due to Beth and Gollmann [6].

Let $A = (a_0,\ a_1,\ \cdots\cdots,\ a_{m-1})$ and $B = (b_0,\ b_1,\ \cdots\cdots,\ b_{m-1})$ be two elements of $GF(2^m)$, where a_i's and b_i's are their respective normal basis coordinates. Let $C = (c_0,\ c_1,\ \cdots\cdots,\ c_{m-1})$ be their product: $C = AB$. Then, any coordinates of, say c_{m-1}, is a function u of A and B which can be obtained by a matrix multiplication, *i.e.*,

$$c_{m-1} = u(A,\ B) = \mathbf{a} \cdot \mathbf{M} \cdot \mathbf{b}^T, \tag{4}$$

where $\mathbf{M}$ is a binary $m \times m$ matrix known as the multiplication matrix [13], $\mathbf{a} = [a_0,\ a_1, \cdots\cdots,\ a_{m-1}]$, $\mathbf{b} = [b_0,\ b_1, \cdots\cdots,\ b_{m-1}]$ and T denotes vector transposition. The numbers of 1's in $\mathbf{M}$ is known as the complexity of the normal basis N [14] and is denoted as

C_N which determines the gate counts and hence time delay for a normal basis multiplier. It is well known that $C_N \geq 2m-1$. If $C_N = 2m-1$, then the normal basis is called an optimal normal basis (ONB).

Let field elements $A, B \in GF(2^m)$ be represented w.r.t. the normal basis N as

$$A = a_{m-1}\beta^{2^{m-1}} + a_{m-2}\beta^{2^{m-2}} + \cdots\cdots + a_0\beta, \;\; a_i \in GF(2), \tag{5}$$

$$B = b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_0\beta, \;\; b_i \in GF(2), \tag{6}$$

respectively. Then the product A and B can be given by

$$\begin{aligned} C = A \cdot B &= A \cdot \left(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_0\beta\right) \\ &= b_{m-1}\left(A\beta^{2^{m-1}}\right) + b_{m-2}\left(A\beta^{2^{m-2}}\right) + \cdots\cdots + b_0(A\beta) \\ &= b_{m-1}\left(A^{1/2^{m-1}}\beta\right)^{2^{m-1}} + b_{m-2}\left(A^{1/2^{m-2}}\beta\right)^{2^{m-2}} + \cdots\cdots + b_0(A\beta). \end{aligned} \tag{7}$$

Also, (7) can be rewritten like the following

$$C = \left[\cdots\left[\left[b_{m-1}A^{1/2^{m-1}}\beta\right]^2 + b_{m-2}A^{1/2^{m-2}}\beta\right]^2 \cdots\cdots\right]^2 + b_0A\beta. \tag{8}$$

In order to realize (8), the structure of Figure 1 is proposed. In the Figure 1, the thick line is the m-bit bus, ▭ is m-bit register, ⊕ is m 2-input XOR gates, ⊙ is m 2-input AND gates, Ⓑ is a constant multiplier that multiplies the element β of $GF(2^m)$. If the register U and V are initialized with two input elements respectively and the register Z is cleared, then we can verify that after the m-th clock cycle the register Z contains the product $C = AB$.

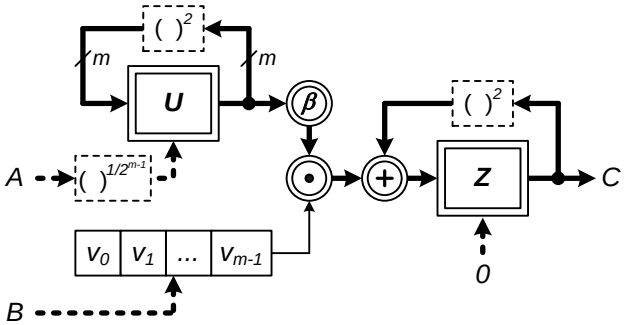

Figure 1. *GF*(2^m) bit-serial normal basis multiplier

Consider the finite field $GF(2^5)$ generated by the irreducible polynomial $p(x) = x^5 + x^2 + 1$ and α be its root, i.e., $p(\alpha) = 0$. We choose $\beta = \alpha^5$, then $\{\beta, \beta^2, \beta^4, \beta^8, \beta^{16}\}$ is a type 2 optimal normal basis. Using (7), C is computed as

$$C = A \cdot B = A \cdot \left(b_4\beta^{2^4} + b_3\beta^{2^3} + b_2\beta^{2^2} + b_1\beta^2 + b_0\beta\right) \tag{9}$$

$$= b_4\left(A^{1/2^4}\beta\right)^{2^4} + b_3\left(A^{1/2^3}\beta\right)^{2^3} + b_2\left(A^{1/2^2}\beta\right)^{2^2} + b_1\left(A^{1/2}\beta\right)^2 + b_0(A\beta)$$

$$= \left[\left[\left[\left[b_4 A^{1/2^4}\beta\right]^2 + b_3 A^{1/2^3}\beta\right]^2 + b_2 A^{1/2^2}\beta\right]^2 + b_1 A^{1/2}\beta\right]^2 + b_0 A\beta,$$

and corresponding $GF(2^5)$ bit-serial normal basis multiplier is shown in Figure 2.

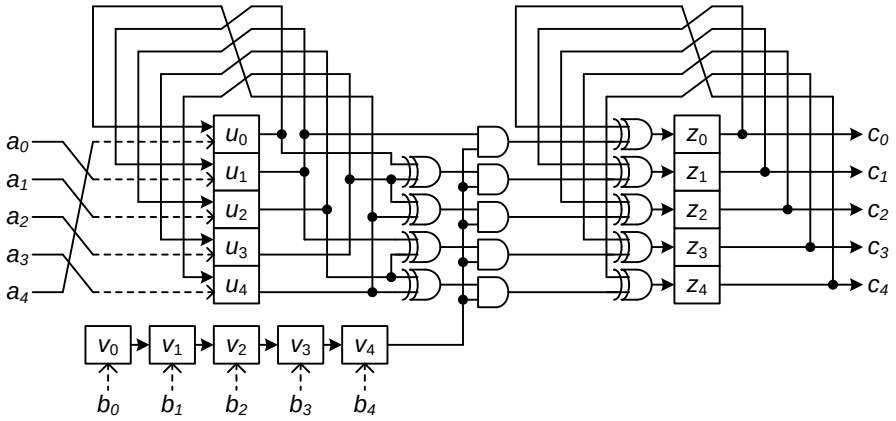

Figure 2. GF(2^5) bit-serial normal basis multiplier

3. The word-parallel bit-serial normal basis multiplier

The bit-serial normal basis multiplier like the Fig. 1 requires m clock cycles to perform one operation. To speed up the process, one element in $GF(2^m)$, B is divided into d $(= \lceil m/w \rceil)$ words of w bits as follows:

$$\begin{aligned} C = {} & A \cdot \left(b_{m-1}\beta^{2^{m-1}} + b_{m-2}\beta^{2^{m-2}} + \cdots\cdots + b_{m-w}\beta^{2^{m-w}}\right) \\ & + A \cdot \left(b_{m-w-1}\beta^{2^{m-w-1}} + b_{m-w-2}\beta^{2^{m-w-2}} + \cdots\cdots + b_{m-2w}\beta^{2^{m-2w}}\right) \\ & + \cdots\cdots\cdots \\ & + A \cdot \left(b_{m-(d-1)w-1}\beta^{2^{m-(d-1)w-1}} + \cdots\cdots + b_1\beta^2 + b_0\beta\right) \end{aligned} \tag{10}$$

Then, (10) can be written as

$$\begin{aligned} C = {} & \left[\cdots\left[\left[b_{m-1}A^{1/2^{w-1}}\beta^{2^{m-w}}\right]^2 + b_{m-2}A^{1/2^{w-2}}\beta^{2^{m-w}}\right]^2\cdots\right]^2 + b_{m-w}A\beta^{2^{m-w}} \\ & + \left[\cdots\left[\left[b_{m-w-1}A^{1/2^{w-1}}\,\beta^{2^{m-2w}}\right]^2 + \cdots\right]^2\cdots\cdots\right]^2 + b_{m-2w}A\beta^{2^{m-2w}} \\ & + \cdots\cdots\cdots \\ & + \left[\cdots\left[\left[b_{m-(d-1)w-1}A^{1/2^{w-1}}\,\beta^{2^{m-dw}}\right]^2 + \cdots\right]^2\cdots\cdots\right]^2 + b_0 A\beta \end{aligned} \tag{11}$$

Based on (11), we now present a word-parallel bit-serial normal basis multiplier structure. The architecture is shown in Figure 3. In operation of this circuit, the register Z is initially empty and two input elements are loaded into the register U and V, respectively. The input operand B is required to be fed into the multiplier in a comb style. Let B be divided into

$d = \lceil m/w \rceil$ words of w bits. Then, in the first clock cycle the inputs are the last bit of every word, b_{m-1}, b_{m-w-1}, ..., $b_{m-(d-1)w-1}$. In the second clock cycle, the inputs are b_{m-2}, b_{m-w-2}, ..., $b_{m-(d-1)w-2}$. Finally, in the w-th clock cycle the inputs are b_{m-w}, b_{m-2w}, ..., b_0. Note that if the subscript of an input bit exceeds m then it is replaced by a zero bit. After w clock cycle, the register Z contains the product AB.

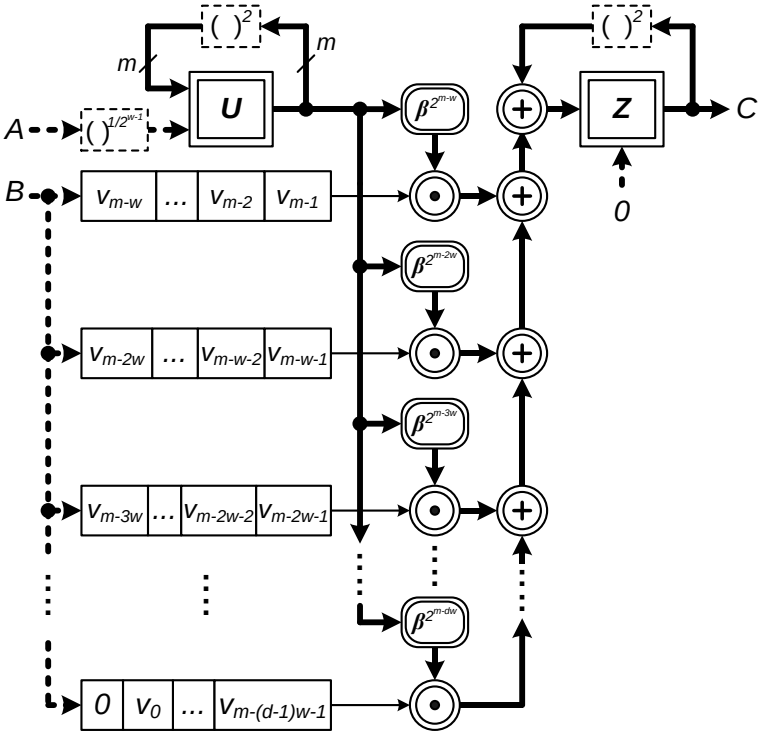

Figure 3. The proposed word-parallel bit-serial normal basis multiplier of GF(2^m)

To further illustrate the new architecture, the proposed word-parallel bit-serial multiplier in $GF(2^5)$ using the normal basis derived in previous example is shown in Figure 4. Figure 4 can be derived from Figure 3 by letting $m = 5$, $w = 3$, and $d = 2$. By substituting these parameters into (10), we have

$$C = A \cdot B = A \cdot \left(b_4\beta^{2^4} + b_3\beta^{2^3} + b_2\beta^{2^2}\right) + A \cdot (b_1\beta^2 + b_0\beta) \tag{12}$$

$$= \left[\left[b_4 A^{1/2^2}\left(\beta^{2^2}\right)\right]^2 + b_3 A^{1/2}\left(\beta^{2^2}\right)\right]^2 + b_2 A\left(\beta^{2^2}\right)$$

$$+ \left[\left[b_1 A^{1/2^2}\left(\beta^{1/2}\right)\right]^2 + b_0 A^{1/2}\left(\beta^{1/2}\right)\right]^2 + 0A\left(\beta^{1/2}\right)$$

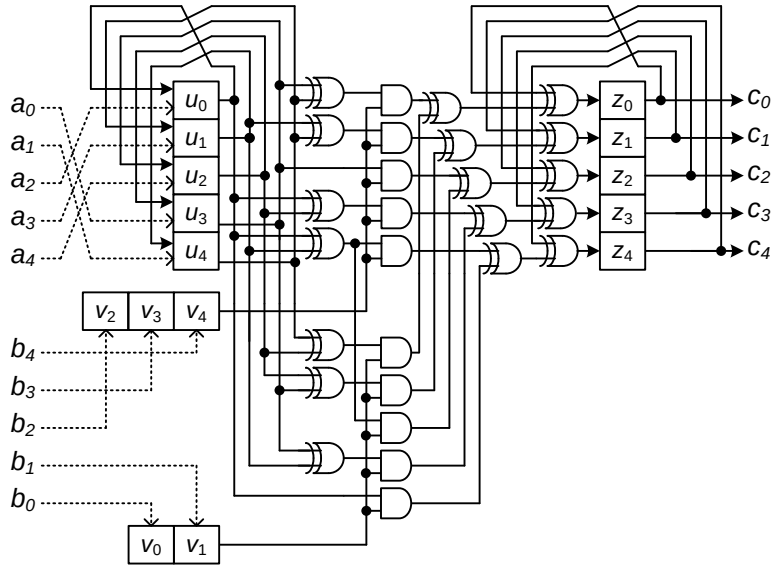

Figure 4. *GF*(2^5) word-parallel bit-serial normal basis multiplier with $w = 3$

Complexity comparison between the proposed multiplier and some other multipliers in the literature is made and shown in Table 1. From the top row, Table 1 represents the word-level Massey-Omura (WLMO) multiplier which uses d identical bit-level Massey-Omura multipliers [5], the AND-efficient digit-serial (AEDS) and XOR-efficient digit-serial (XEDS) multipliers proposed in [10], the word-level sequential multipliers with parallel output type I and type II reported in [11], and the comb style architecture proposed in [12]. The last row of the table presents the proposed architecture. Compared to the other architectures, it can be seen from the table the proposed architecture has much smaller area and delay cost.

Table 1. Complexities Comparison between different word levle multipliers ($d = \lceil m/w \rceil$)

Multiplier	# AND	# XOR	Critical path delay	Basis
WLMO [5]	$d(2m-1)$	$d(2m-2)$	$T_A + \lceil \log_2(2m-1) \rceil T_X$	ONB II
AEDS [10]	$d(m - \frac{d}{2} + \frac{1}{2})$	$d(3m-d-2)$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$	ONB II
XEDS [10]	$d(2m-d)$	$d(2m - \frac{d}{2} + \frac{3}{2})$	$T_A + (1 + \lceil \log_2 m \rceil) T_X$	ONB II
w-SMPOI [11]	$d\left(\lfloor \frac{m}{2} \rfloor + 1\right) + m$	$d(2m-1)$	$2T_A + (3 + \lceil \log_2(d-1) \rceil) T_X$	ONB II
w-SMPOII [11]	$dm + m$	$d(m + \lfloor \frac{m}{2} \rfloor)$	$2T_A + (3 + \lceil \log_2(d-1) \rceil) T_X$	ONB II
Comb Style [12]	dm	$2dm$	$T_A + (1 + \lceil \log_2(d+1) \rceil) T_X$	RNB
Proposed	dm	$d(2m-1)$	$T_A + (1 + \lceil \log_2(d+1) \rceil) T_X$	ONB II

4. Conclusion

In this paper, a new word-parallel bit-serial normal basis multiplier is proposed. Unlike the bit-serial multipliers which require m clock cycles for the latency, the proposed word-parallel bit-serial multiplier has the latency of w ($1 \leq w \leq m$) clock cycle. The word-parallel bit-serial multiplier is faster than bit-serial multipliers and has lower hardware area complexity than bit-parallel multipliers. Therefore, the most significant feature of the proposed multiplier is a proper trade-off between hardware complexity and delay time.

For the proposed multiplier, architectural level details have been presented and they have been compared with other similar multipliers in terms of the number of AND gates, XOR gates, and critical path delays. The comparison results show that the proposed multipliers have the least number of gates and critical path delay. The proposed multiplier structure is also modular and, hence, suitable for VLSI realization.

Acknowledgements

Funding for this paper was provided by Namseoul University.

References

[1] M. Y. Rhee, "Error-Correcting Coding Theory", McGraw-Hill, **(1989)**.
[2] R. Lidl and H. Niederreiter, "Introduction to Finite Fields and Their Applications", Cambridge Univ. Press, **(1994)**.
[3] D. Hankerson, A. Menezes and S. Vanstone, "Guide to Elliptic Curve Cryptography", Springer-Verlag, **(2004)**.
[4] S. B. Wicker and V. K. Bhargava, (eds.), "Reed-Solomon Codes and Their Applications", IEEE Press, New York, NY, **(1994)**.
[5] J. L. Massey and J. K. Omura, "Computational method and apparatus for finite field arithmetic", U.S. patent application, **(1984)**.
[6] T. Beth and D. Gollmann, "Algorithm Engineering for Public Key Algorithms", IEEE Journal on Selected Areas in Communications, vol.7, no.4, **(1989)** May, pp.458-466.
[7] G. B. Agnew, R. C. Mullin, I. M. Onyszchuck and S. A. Vanstone, "An Implementation for a Fast Public-Key Cryptosystem", Journal of Cryptology, vol.3, **(1991)**, pp.63-79.
[8] A. Reyhani-Masoleh and M. A. Hasan, "A new construction of Massey-Omura parallel multiplier over $GF(2^m)$", IEEE Transactions on Computers, vol. 51, no. 5, **(2002)** May, pp. 511–520.
[9] C. K. Koc and B. Sunar, "Low-complexity bit-parallel canonical and normal basis multipliers for a class of finite fields", IEEE Transactions on Computers, vol. 47, no. 3, **(1998)** March, pp. 353–356.
[10] A. Reyhani-Masoleh and M. A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over Binary Extension Fields", ACM Transactions on Embedded Computing Systems, vol. 3, no. 3, **(2004)** August, pp. 575-592.
[11] A. Reyhani-Masoleh and M. A. Hasan, "Low Complexity Word-Level Sequential Normal Basis Multipliers", IEEE Transactions on Computers, vol. 54, no. 2, **(2005)** February, pp. 98-110.
[12] A. H. Namin, H. Wu and M. Ahmadi, "Comb Architectures for Finite Field Multiplication in F_{2^m}", IEEE Transactions on Computers, vol. 56, no. 7, **(2007)** July, pp.909-916.
[13] IEEE Std 1363-2000, "IEEE Standard Specifications for Public-Key Cryptography", **(2000)** January.
[14] R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone and R. M. Wilson, "Optimal Normal Bases in $GF(p^n)$", Discrete Applied Mathematics, vol. 22, **(1988/89)**, pp.149–161.

Authors

Yong-Suk Cho

He received the B.S., M.S., and Ph.D. degrees in electronic communication engineering from Hanyang University, Seoul Korea in 1986, 1988 and 1998, respectively. From 1989 to 1996, he was with Korea Telecom in Korea. Since 1996 he has been with Department of Information & Communication Security, Yonungdong University, Korea, where he is now an associate professor. His current research interests include finite field arithmetic, cryptography, and error-control coding.

Jae-Yeon Choi

He received the B.S., M.S., and Ph.D. degrees in electronic communication engineering from Hanyang University, Seoul Korea in 1985, 1987 and 1998, respectively. From 1987 to 1989, he was with Samsung Advanced Institute of Technology. From 1989 to 1992, he was with LG Information & Communication Research Center. Since 1996 he has been with Department of Information & Communication, Namseoul University, Korea, where he is now a professor. His current research interests include finite field arithmetic, cryptography, and error-control coding. (Corresponding author of this paper)