A Scalable Structure for a Multiplier and an Inversion Unit in GF(2^m)

Chanho Lee and Jeongho Lee

Elliptic curve cryptography (ECC) offers the highest security per bit among the known public key cryptosystems. The operation of ECC is based on the arithmetic of the finite field. This paper presents the design of a 193-bit finite field multiplier and an inversion unit based on a normal basis representation in which the inversion and the square operation units are easy to implement. This scalable multiplier can be constructed in a variable structure depending on the performance area trade-off. We implement it using Verilog HDL and a 0.35 µm CMOS cell library and verify the operation by simulation.

Keywords: Elliptic curve cryptography, multiplier, inversion, finite field.

I. Introduction

Electronic security has been of considerable interest in recent years because of the increase in electronic transactions. The developing technology requires a longer key length to satisfy higher levels of security. However, as the key length becomes longer, the operation time increases even more, as does the design complexity and area. Hence, a cryptography algorithm with a short key length and a satisfactory security level is desirable.

Elliptic curve cryptography (ECC) offers the highest security per bit among the known public key cryptosystems [1]-[3]. For example, the RSA system (Rivest, Shamir and Adelman) [4] with a 1024-bit key has a security level similar to an ECC system with a 160-bit key [5]. The benefit of smaller key sizes makes ECC particularly attractive for embedded applications since its implementation requires less memory and processing power [3].

Several standard specifications of ECC recommend an elliptic curve over a finite field with a size of 160 bits or more. The elliptic curve cryptosystem with a 160-bit modulus is expected to be secure for 10 years [6], [7]. We expect that the elliptic curve cryptosystem with a 193-bit modulus will be secure for 20 years [7].

The operation of ECC is based on the arithmetic of the finite field. The most frequently used finite field arithmetic operations in ECC are addition and multiplication, and the most time-consuming finite field arithmetic operation in ECC is inversion. The finite field operation can be performed based on a polynomial basis representation or a normal basis representation. The square operation and inverse operation are easy to implement on a normal basis representation [8]. The shortcoming of the multiplier based on a normal basis

Manuscript received Jan. 15, 2003; revised June 30, 2003.

This work was supported by the Soongsil University Research Fund.

Chanho Lee (phone: +82 2 820 0710, email: chanho@e.ssu.ac.kr) and Jeongho Lee (email: enom@e.ssu.ac.kr) are with the Department of Electronic Engineering, Soongsil University, Seoul, Korea.

operation is that the binary function for the multiplication must be recalculated for different sizes of operands. However, since only several ECC systems are commonly used, this shortcoming is not significant.

There are roughly three types of efficient inversion algorithm for the normal basis representation. Itoh and Tsujii [9] proposed one, G. L. Feng [10] another, and Takagi [11] yet another. The performance of the inversion unit in a normal basis representation depends on that of the multiplier. There are three types of multipliers: the parallel input and serial output by Massey and Omura [12] and Reyhani-Masoleh [13], the serial input and parallel output by G. L. Feng [10], and the parallel input and parallel output by C. C. Wang [14] and Reyhani-Masoleh [15]. The first and second types take *m* cycles to obtain the result for m-bit operation. The last one needs more than *m* times the area of the others though it takes only 1 cycle to obtain the result.

In this paper, we propose a new multiplier structure with scalable output sizes and operation cycles in GF(2^m) using a normal basis. The number of output bits can be freely chosen in the new architecture with the performance area trade-off depending on the application. Using this architecture, we designed a 193-bit multiplication/inversion unit with a multiplier with an 8-bit output/cycle and a 25 cycle-operation time. We implemented it using 0.35 μm CMOS technology and verified the operation by simulation.

II. Multiplier

A normal basis for GF(2^m) is a set of the form

$$\{\alpha^{2^0}, \alpha^{2^1}, \alpha^{2^2}, \Lambda, \alpha^{2^{m-2}}, \alpha^{2^{m-1}}\}. \qquad (1)$$

The representation of GF(2^m) via the normal basis is carried out by interpreting the bit string ($a_0a_1a_2\cdots a_{m-1}$) as the element

$$A = a_0\alpha^{2^0} + a_1\alpha^{2^1} + a_2\alpha^{2^2} + \Lambda + a_{m-1}\alpha^{2^{m-1}}. \qquad (2)$$

In the normal basis representation, A^2 is a cyclic shift of A.

$$A^2 = a_{m-1}\alpha^{2^0} + a_0\alpha^{2^1} + a_1\alpha^{2^2} + \Lambda + a_{m-2}\alpha^{2^{m-1}}. \qquad (3)$$

Let $A=(a_0a_1a_2\cdots a_{m-1})$ and $B=(b_0b_1b_2\cdots b_{m-1})$ be two elements GF(2^m) in a normal basis representation and $C=(c_0c_1c_2\cdots c_{m-1})$ be the product. The last term c_{m-1} of the product *C* is some binary function of the components of *A* and *B*.

$$c_{m-1} = f(a_0, a_1, \Lambda, a_{m-1}; b_0, b_1, \Lambda, b_{m-1}). \qquad (4)$$

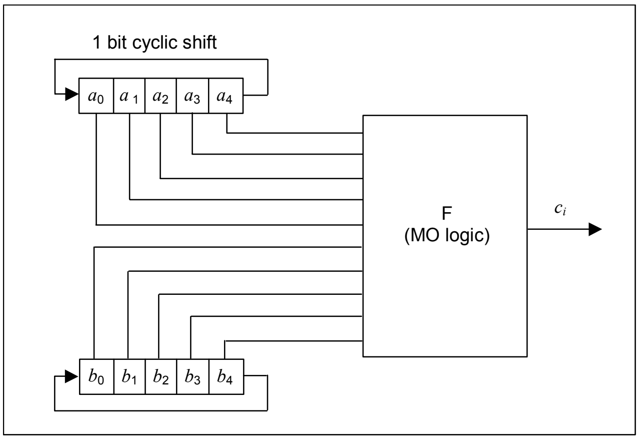


Fig. 1. Massey-Omura multiplier.

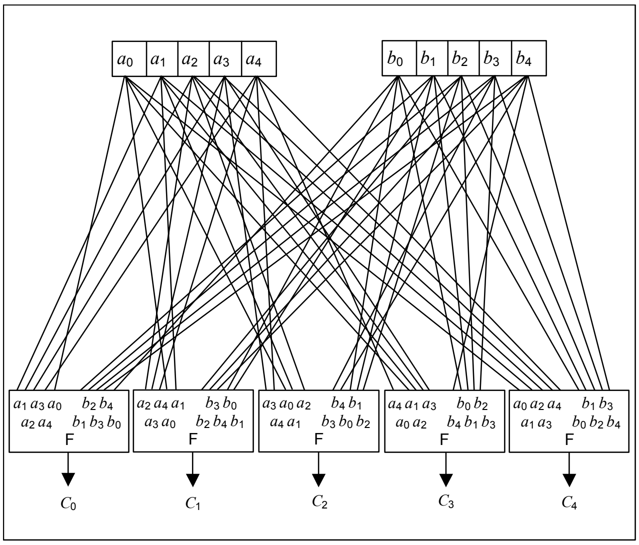

a_0 a_1 a_2 a_3 a_4
b_0 b_1 b_2 b_3 b_4
$a_1\,a_3\,a_0$ $b_2\,b_4$ $a_2\,a_4$ $b_1\,b_3\,b_0$ F
$a_2\,a_4\,a_1$ $b_3\,b_0$ $a_3\,a_0$ $b_2\,b_4\,b_1$ F
$a_3\,a_0\,a_2$ $b_4\,b_1$ $a_4\,a_1$ $b_3\,b_0\,b_2$ F
$a_4\,a_1\,a_3$ $b_0\,b_2$ $a_0\,a_2$ $b_4\,b_1\,b_3$ F
$a_0\,a_2\,a_4$ $b_1\,b_3$ $a_1\,a_3$ $b_0\,b_2\,b_4$ F
C_0
C_1
C_2
C_3
C_4

Fig. 2. Parallel type Massey-Omura multiplier.

The squaring is the same as the cyclic shift and $C^2=A^2\times B^2$, and therefore c_{m-2} is obtained using the same binary function as that used to obtain c_{m-1} except that the components of *A* and *B* are rotated.

$$c_{m-2} = f(a_{m-1}, a_0, \Lambda, a_{m-2}; b_{m-1}, b_0, \Lambda, b_{m-2}). \qquad (5)$$

Other terms of the product can be obtained using the same binary function as shown below.

$$\begin{aligned} c_{m-1} &= f(a_0, a_1, \Lambda, a_{m-1}; b_0, b_1, \Lambda, b_{m-1}) \\ c_{m-2} &= f(a_{m-1}, a_0, \Lambda, a_{m-2}; b_{m-1}, b_0, \Lambda, b_{m-2}) \\ &\mathrm{M} \\ c_0 &= f(a_1, a_2, \Lambda, a_0; b_1, b_2, \Lambda, b_0). \end{aligned} \qquad (6)$$

Equation (6) defines the serial Massey-Omura (MO) multiplier [11]. Figure 1 depicts the block diagram of the serial MO multiplier in GF(2^5).

The serial MO multiplier is the serial-output type, and it takes *m* clock to obtain all the terms of the product. The parallel-output multiplier can be implemented using *m* binary function logic units in 1 cycle. However, it needs more than *m* times the area. Figure 2 shows the parallel multiplier in GF(2^5) [16].

Equation (4) can be rewritten as shown below using a number, *n* (*n* is an integer, $1 \le n \le m$):

$$
\begin{aligned}
&(c_{m-1}, \ldots, c_{m-n}) \\
&= (f(a_0, \ldots, a_{m-1}; b_0, \ldots, b_{m-1}), \ldots, \\
&\quad f(a_{m-n+1}, \ldots, a_{m-n}; b_{m-n+1}, \ldots, b_{m-n})) \\
&(c_{m-n-1}, \ldots, c_{m-2n}) \\
&= (f(a_{m-n}, \ldots, a_{m-n-1}; b_{m-n}, \ldots, b_{m-n-1}), \ldots, \\
&\quad f(a_{m-2n+1}, \ldots, a_{m-2n}; b_{m-2n+1}, \ldots, b_{m-2n})) \\
&\vdots \\
&(c_{m-kn-1}, \ldots, c_0, c_{m-1}, \ldots, c_{m-n+r}) \\
&= (f(a_{m-kn}, \ldots, a_{m-1}, a_0, \ldots, a_{m-kn-1}; \\
&\qquad b_{m-kn}, \ldots, b_{m-1}, b_0, \ldots, b_{m-kn-1}), \ldots, \\
&\quad f(a_{m-n+r+1}, \ldots, a_{m-1}, a_0, \ldots, a_{m-n+r}; \\
&\qquad b_{m-n+r+1}, \ldots, b_{m-1}, b_0, \ldots, b_{m-n+r}))
\end{aligned}
\tag{7}
$$

where $k-1 = \lceil m/n \rceil$ and $m = k \cdot n + r$. An *m*-bit multiplier can be implemented using *n* binary F-function logic units and *n*-bit cyclic shift LFSRs if *C* is partitioned into subsets with *n* elements as shown in (7). The complexity of an F-function logic unit depends on the generator polynomial and m. The product can be obtained in $\lceil m/n \rceil$ cycles. Figure 3 gives the block diagram of the proposed multiplier in GF(2^5).

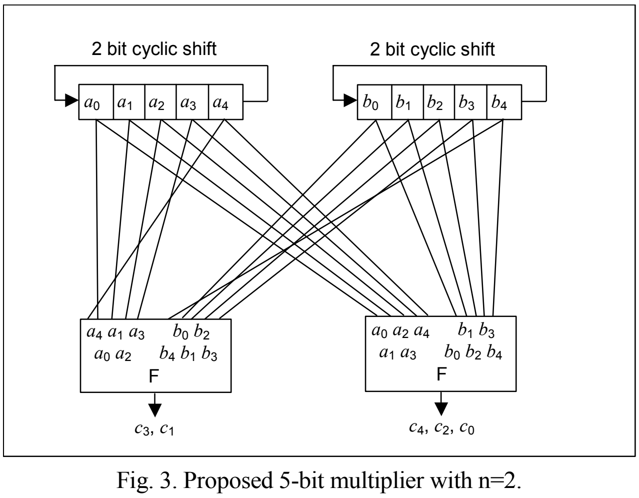


Fig. 3. Proposed 5-bit multiplier with n=2.

III. Inversion Unit

Among the three types of inversion algorithm, the inversion unit by G. L. Feng is preferred for implementation because of the smaller multiplication count for 193-bit operations. The inversion unit in GF(2^m) is designed according to the algorithm by G. L. Feng as shown below [10].

```
Step 1    $\delta := \beta$
Step 2    for $i=q$ to 1 do
Step 3    begin
              if $m_i=1$ then $\delta := (\delta)^{2^{-2^i}}$
Step 4        $\delta := \delta \times (\delta)^{2^{2^{i-1}}}$
Step 5        if $m_{i-1}=1$ then $\delta := \delta \times \beta$
          end
Step 6    $\beta^{-1} := (\delta)^{2^{m-m_0}}$
```

where $m_q m_{q-1} \cdots m_0$ is the binary expression of the number $m-1$ and β is the input of the inversion. For *i* ($1 \le i \le q$), step 3, step 4, and step 5 are executed. For example, if $m=193$ then $m-1 = 192 = (11000000)_2 = (m_7 m_6 m_5 m_4 m_3 m_2 m_1 m_0)_2$ and $q=7$. $(\delta)^{2^{-2^i}}$ denotes the 2^i bit cyclic left shift of δ and $(\delta)^{2^{2^{i-1}}}$ denotes the 2^{i-1} bit cyclic right shift of δ. Finally, if $m_0=1$, the 1 bit cyclic left shift of δ is the final result, and otherwise, δ is the final result.

IV. Implementation

We use the trinomial $f(x)=x^{193}+x^{15}+1$ that is recommended in IEEE standard 1363 [17] for a generator polynomial, and the F-function consists of roughly 4m AND and XOR gates. The multiplier and inversion unit are implemented using a 0.35 μm CMOS technology and Verilog HDL.

We implement a multiplier that is operated with 8 bits per cycle because the implementation efficiency is the best. The block diagram of the implemented multiplier is shown in Figure 4. The operation bit per cycle of the multiplier can be changed easily by modifying the parameters of the HDL code.

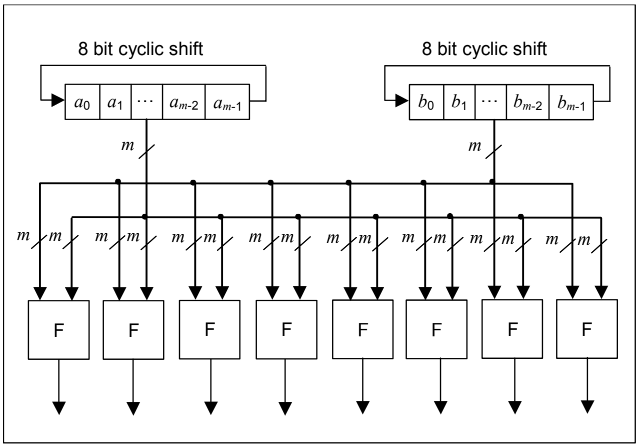


Fig. 4. The block diagram of the m=193-bit multilpier.

Figure 5 shows the implementation efficiency of the multiplier according to the operation digit. The implementation efficiency of a multiplier is defined as the ratio of the throughput and the gate count in the unit of a 2-input NAND gate, which means the number of operations per gate per second in a structure. The 193-bit scalable multiplier units with various operation digits are synthesized using Synopsys tools and a 0.35 μm CMOS technology. Gate counts, operating frequencies, throughputs, and implementation efficiencies are analyzed according to the synthesis results. The implementation efficiencies are calculated using the throughputs and the gate counts for various *n*. High implementation efficiency means that an operation unit is implemented with a smaller gate count for the same throughput. A multiplier with an 8-bit operation has the best implementation efficiency. The maximum operation frequency is 217 MHz, and the maximum throughput is 7.4 million operations per second (Mop/s).

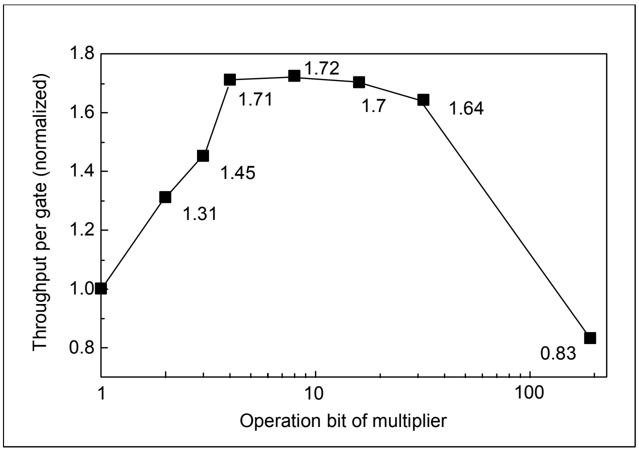


Fig. 5. The implemetation efficiency of multipliers.

Table 1. Characteristics of multipliers.

Operation digit	1^*	4	8	16	193^{**}
Total number of gates (normalized)	8,451 (1)	19,335 (2.3)	36,228 (4.3)	67,269 (8.0)	1,646,978 (194.9)
Maximum clock frequency (MHz)	227	217	217	208	192
Maximum throughput (Mop/s) (normalized)	1.18 (1)	4.63 (3.9)	8.68 (7.4)	16 (13.6)	192 (162.7)
Implementation efficiency (op/s/gate) (normalized)	140 (1)	239 (1.71)	240 (1.72)	238 (1.7)	117 (0.83)

* the same as the serial MO multiplier

** the same as the parallel MO multiplier

Table 1 presents a summary of the characteristics of the implemented multiplier. The proposed multiplier with an 8-bit operation per cycle takes 4 times the area and 7 times the throughput compared with the serial MO multiplier. However, the parallel MO multiplier takes 195 times the area and 163 times the throughput compared with the serial MO multiplier. Table 1 confirms that our proposed multiplier is more efficient in implementation than the previous ones. The operation frequency degrades due to the increased load though the operation frequency should be the same in principle since the same F-functions are employed. The proposed multiplier with an 8-bit operation per cycle has twice the implementation efficiency of the parallel MO multiplier. Another paper presented a similar approach of a configurable structure of multiplier using the polynomial basis in which decomposing a structure was relatively easy [18].

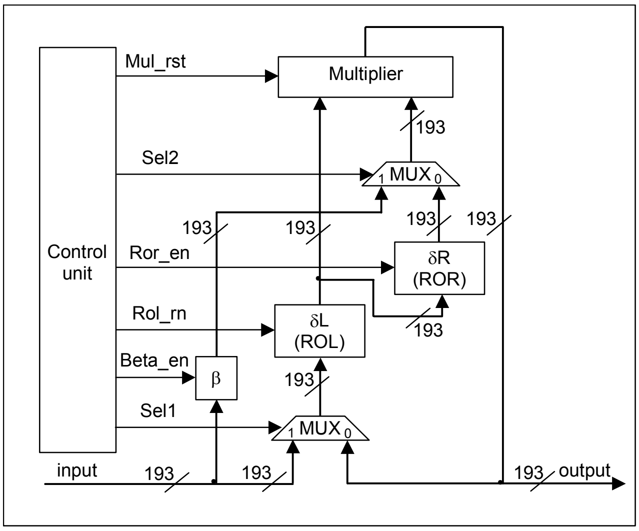


Fig. 6. The block diagram of the inversion unit.

The block diagram of the implemented inversion unit is shown in Fig. 6. The β register stores the input data of the inversion unit, the δL and δR registers perform a square root and a square operation in the finite field, respectively. The multiplier is a scalable finite field multiplier. The control unit issues appropriate control signals according to the inversion algorithm described in section III. The inversion unit can also be used for multiplication.

Figure 7 presents the implementation efficiency of the inversion unit according to the operation bits. We used an inversion unit with an 8-bit operation because it has the best efficiency. The maximum operation frequency is 217 MHz, and the maximum throughput is 0.97 Mop/s as a result of post-synthesis simulation using a 0.35 μm CMOS technology.

Table 2 summarizes the characteristics of the implemented inversion unit. The inversion unit with the proposed multiplier takes 2.3 times the area and 7 times the throughput compared with the one with a serial MO multiplier. However, the inversion unit with the parallel MO multiplier takes 88 times the area and 40 times the throughput compared with the one with a serial MO multiplier. Table 2 confirms that our proposed inversion unit is very efficient in implementation compared with the previous ones. The inversion unit with the proposed multiplier has 7 times the implementation efficiency compared with the one with the parallel MO multiplier. This means that the proposed inversion unit uses 1/7 of the gate counts for the same throughput as the fully parallel inversion unit.

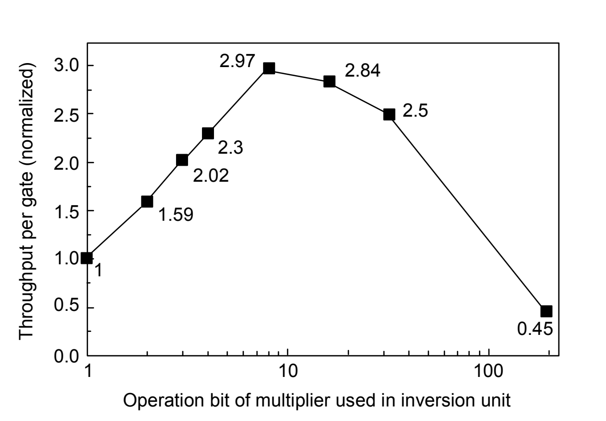


Fig. 7. The Implementation efficiency of inversion units.

Table 2. Characteristics of inversion units.

Operation digit	1*	4	8	16	193**
Total number of gates (normalized)	18,896 (1)	30,567 (1.6)	44,019 (2.3)	77,494 (4.1)	1,658,304 (87.8)
Maximum clock frequency (MHz)	217	217	217	208	179
Maximum throughput (Mop/s) (normalized)	0.14 (1)	0.52 (3.72)	0.97 (6.93)	1.63 (11.65)	5.59 (39.93)
Implementation efficiency (op/s/gate) (normalized)	7 (1)	17 (2.3)	22 (2.97)	21 (2.84)	3 (0.45)

* the same as the inversion unit with the serial MO multiplier
** the same as the inversion unit with the parallel MO multiplier

V. Conclusion

We presented a scalable finite field multiplier structure for ECC operation based on a normal basis representation over GF(2^{193}). The number of output bits of the multiplier can be freely chosen in the new architecture with the performance area trade-off depending on the application. Using this architecture, we designed a 193-bit multiplier and a 193-bit multiplication/inversion unit. Because the major limiting factor of the performance of the inversion unit is the number of multiplication operations, the inversion unit can be designed with a scalable design area and operation cycles using the proposed multiplier.

The multiplier and inversion units are designed using Verilog HDL and implemented using a 0.35 μm CMOS technology. The multiplier and the inversion unit have up to twice and 7 times the implementation efficiency compared with the conventional multiplier and inversion units, respectively. The area performance efficiency of the proposed inversion unit is much higher than the previous ones.

References

[1] N. Koblitz, "Elliptic Curve Cryptosystems," *Mathematics of Computation*, vol. 48, 1987, pp. 203-209.

[2] V. Miller, "Uses of Elliptic Curves in Cryptography," *Advances in Cryptology – CRYPTO'*85, Springer-Verlag, LNCS 218, 1986, pp. 417-726.

[3] K.H. Leung, K.W. Ma, W.K. Wong, and P.H.W. Leong, "FPGA Implementation of a Microcoded Elliptic Curve Cryptographic Processor," *IEEE Symposium on Field Programmable Custom Computing Machines*, 2000, pp. 68-76.

[4] R.L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, no. 2, 1978, pp. 120-126.

[5] SEC1, *Elliptic Curve Cryptography*, v.1.0, Sept. 20, 2000, p. 62.

[6] TTAS.KO-12.0015, *Digital Signature Mechanism with Appendix – Part 3: Korean Certificate-Based Digital Signature Algorithm Using Elliptic Curves*, pp. 8-10, 61-70.

[7] A. Lenstra and E. Verheul, "Selecting Cryptographic Key Sizes," *J. of Cryptology*, vol. 14, no. 4, 2001, pp. 255-293.

[8] S. Sutikno, R. Effendi, and A. Surya, "Design and Implementation of Arithmetic Processor for $F_{2^{155}}$ for Elliptic Curve Cryptosystems," *IEEE APCCAS* 1998, pp. 647-650.

[9] T. Itoh, O. Teechai, and S. Tsujii, "A fast Algorithm for Computing Multiplicative Inverses in GF(2^m) Using Normal Bases," *Information and Computation* 78, 1988, pp. 171-177.

[10] G.L. Feng, "A VLSI Architecture for Fast Inversion in GF(2^m)," *IEEE Trans. Comput.*, vol. 38, no. 10, Oct. 1989, pp. 1383-1386.

[11] N. Takagi, J. Yoshiki, and K. Takagi, "A Fast Algorithm for Multiplicative Inversion in GF(2^m) Using Normal Basis," *IEEE Trans. Comput.*, vol. 50, no. 5, 2001, pp. 394 -398.

[12] J.L. Massey and J.K. Omura, *Computational Method and Apparatus for Finite Field Architecture*, U.S. Patent Application, submitted 1981.

[13] A. Reyhani-Masoleh and M.A. Hasan, "Efficient Digit-Serial Normal Basis Multipliers over GF(2^m)," *ISCAS* 2002, vol. 5, pp. 781-784.

[14] C.C. Wang, T.K. Trung, H.M. Shao, L.J. Deutsch, J.K. Omura,

and I.S. Reed, "VLSI Architectures for Computing Multiplications and Inverses in GF(2^m)," *IEEE Trans. Comput.*, vol. C-34, Aug. 1985, pp. 709-717.

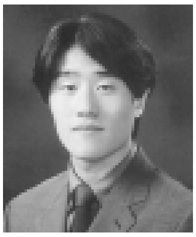
[15] A. Reyhani-Masoleh and M.A. Hasan, "A New Construction of Massey-Omura Parallel Multiplier over GF(2^m)," *IEEE Trans. Comput.*, vol. 51, no. 5, 2002, pp. 511-520.

[16] Y.R. Shayan and T. Le-Ngoc, "The Least Complex Parallel Massey-Omura Multiplier and its LCA and VLSI Designs, Circuits, Devices and Systems," *IEE Proc. G*, vol. 136, issue 6, Dec. 1989, pp. 345-349.

[17] IEEE Standard 1363-2000, *IEEE Standard Specifications for Public-Key Cryptography*, p. 110.

[18] L. Song and K.K. Pahri, "Low-Energy Digit-Serial/Parallel Finite Field Multipliers," *J. of VLSI Signal Processing*, vol. 1, no. 22, 1998, pp. 149-166.

Chanho Lee received his BS and the MS degrees in electronic engineering from Seoul National University, Seoul, Korea, in 1987 and in 1989, and the PhD degree from the University of California, Los Angeles, in 1994. In 1994, he joined the Semiconductor R&D Center of Samsung Electronics, Kiheung, Korea. Since 1995, he has been a faculty member of the School of Electronic Engineering, Soongsil University, Seoul, Korea, and he is currently an Associate Professor. His research interests are channel codec, SoC design methodology, security processors, low power SoC design, and on-chip networks. He is a Senior Member of IEEE.



Jeongho Lee received his BS and MS degrees in electronic engineering from Soongsil University, Seoul, Korea, in 2001 and 2003. He joined HWA Sound Source Co. in 2003 and is working on ASIC design. His research interests are the design of security modules, microprocessors, and digital signal processing units.