http://dx.doi.org/10.6109/ijice.2011.9.6.683

Speed Optimized Implementation of HUMMINGBIRD Cryptography for Sensor Network

HwaJeong Seo, Howon Kim*, *Member, KIICE*

Abstract— **The wireless sensor network (WSN) is well known for an enabling technology for the ubiquitous environment such as real-time surveillance system, habitat monitoring, home automation and healthcare applications. However, the WSN featuring wireless communication through air, a resource constraints device and irregular network topology, is threatened by malicious nodes such as eavesdropping, forgery, illegal modification or denial of services. For this reason, security in the WSN is key factor for utilizing the sensor network into the commercial way. There is a series of symmetric cryptography proposed by laboratory or industry for a long time. Among of them, recently proposed HUMMINGBIRD algorithm, motivated by the design of the well-known Enigma machine, is much more suitable to resource constrained devices, including smart card, sensor node and RFID tags in terms of computational complexity and block size. It also provides resistance to the most common attacks such as linear and differential cryptanalysis. In this paper, we implements ultra-lightweight cryptography, HUMMINGBIRD algorithm into the resource constrained device, sensor node as a perfectly customized design of sensor node.**

Index Terms— **HUMMINGBIRD ALGORITHM, Implementation, MSP430, Sensor network.**

I. INTRODUCTION

RECENT advances in ubiquitous computing, which is a post-desktop model of human-computer interaction, has enabled the development of applications including health care system, home automation, surveillance system and environment monitoring. Since ubiquitous computing with sensor network has been getting important for managing the system more efficiently, many laboratories have researched into a technology in terms of a sensor network, hardware architecture of sensor node and applications using sensor network and security on sensor network. However, these services are utilized over security matters.

Manuscript received September 9, 2011; revised October 9, 2011; accepted October 28, 2011.

Hwajeong Seo is with the Department of Computer Engineering, Pusan National University, Pusan, 609-735, Korea (Email: hwajeong@pusan.ac.kr)

Howon Kim* (Corresponding author) is Professor in the Department of Computer Engineering, Pusan National University, Pusan, 609-735, Korea (Email: howonkim@pusan.ac.kr)

For this reason, sensor network is required to consolidate a security over communication line between each user. Compared to existing network, sensor network faces unique challenges. The sensor is the resource constraints device in terms of capacity of memory, computational performance and limited power supply. Moreover, strong security is also required because it is exposed to malicious node than traditional networking as installation of accessible area and communication through the shared medium. In these circumstances, recently proposed HUMMINGBIRD cryptography featuring reduced block size and light weight cryptography is the best practice. HUMMINGBIRD has a hybrid structure of block cipher and stream cipher providing the designed security with small block size and resistance to the most common attacks such as linear and differential cryptanalysis. In this paper, we present the speed optimized implementation of HUMMINGBIRD cryptography for representative sensor mote, MSP430 with assembly code. This reduces the clock consumption significantly.

This paper is organized as follow: In section 2, we introduce related works including HUMMINGBIRD Cryptography, experiential environment. In section 3, we present an optimized implementation of HUMMINGBIRD Cryptography over sensor node, MSP430 processor. The evaluation is addressed in section 4. Finally, the conclusion is drawn in section 5.

II. RELATED WORK

A. HUMMINGBIRD Cryptography

HUMMINGBIRD is motivated by the design of the well-known enigma machine, combining of the block ciphers or stream ciphers with a 16-bit block size, 256-bit key size and 80-bit internal state. The size of the key and the internal state of HUMMINGBIRD provides a security level which is adequate for many applications such as sensor node and RFID Tags [1][3][5].

1) Notation

The following are the notation and description related with HUMMINGBIRD algorithm.

TABLE I
NOTATION

Notation	Description
PT_i	The i-th 16-bit plaintext block, $i = 1,2,...,n$
CT_i	The i-th 16-bit ciphertext block, $i = 1,2,...,n$
K	The 256-bit secret key
$E_K(\cdot)$	The encryption function of Hummingbird with 256-bit secret key K
$D_K(\cdot)$	The decryption function of Hummingbird with 256-bit secret key K
k_i	The 64-bit subkey used in the i-th block cipher, $i = 1,2,3,4$ such that $K = k_1 \| k_2 \| k_3 \| k_4$
$E_{k_i}(\cdot)$	A block cipher encryption algorithm with 16-bit input, 64-bit key k_i ,and 16-bit output, i.e., $E_{k_i} : \{0,1\}^{16} \times \{0,1\}^{16} \to \{0,1\}^{16}, i = 1,2,3,4$
$D_{k_i}(\cdot)$	A block cipher decryption algorithm with 16-bit input, 64-bit key k_i , and 16-bit output, i.e., $D_{k_i} : \{0,1\}^{16} \times \{0,1\}^{16} \to \{0,1\}^{16}, i = 1,2,3,4$
RS_i	The i-th 16-bit internal state register, $i = 1,2,3,4$
$LFSR$	A 16-stage Linear Feedback Shift Register with the characteristic polynomial $f(x) = x^{16} + x^{15} + x^{12} + x^{10} + x^7 + x^3 + 1$
$\boxplus$	Modulo 2^{16} addition operator
$\boxminus$	Modulo 2^{16} subtraction operator
$\oplus$	Exclusive-OR (XOR) operator
$m \ll l$	Left circular shift operator, which rotates all bits of m to the left by l bits, as if the left and the right ends of m were joined,
$K_j^{(i)}$	The j-th 16-bit key used in the i-th block cipher, $j = 1,2,3,4$ such that $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$
$S_i(x)$	The i-th 4-bit to 4-bit S-box used in the block cipher, $S_i(x) : F_2^4 \to F_2^4, i = 1,2,3,4$
$NONCE_i$	the i-th nonce which is a 16-bit random number, $i = 1,2,3,4$
IV	The 64-bit initial vector, such that $IV = NONCE_1 \| NONCE_2 \| NONCE_3 \| NONCE_4$

2) Encryption & Decryption process

The structure of the HUMMINGBIRD encryption algorithm consists of four 16-bit block ciphers, four 16-bit internal state registers and a 16-stage LFSR. The 256-bit secret key is divided into four 64-bit subkeys which are used in the four block ciphers respectively. A 16-bit plaintext block is encrypted by first executing a modulo 2^{16} addition of plaintext and the content of the first internal state register. The result of the addition is then encrypted by the first block cipher. This procedure is repeated 3 times more. The state of the four internal state registers will also be updated unpredictable way based on their current states, the output of three block ciphers and the state of the LFSR

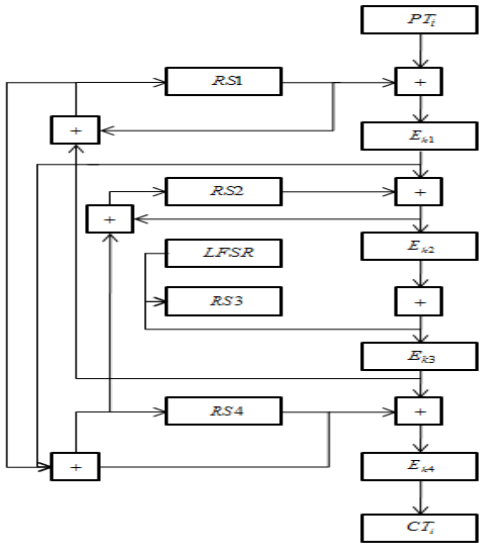

Fig. 1. Encryption process

The decryption process is similar to the encryption process, which conducts inverse operation such as inverse order of internal state, a modulo 2^{16} subtraction.

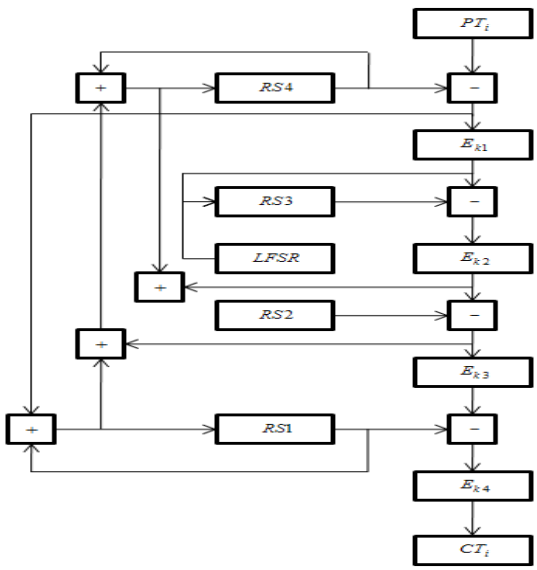

Fig. 2. Decryption process

3) 16-bit block cipher encryption & decryption

The 16-bit block cipher is typical substitution-permutation network with 16-bit block size and 64-bit key as shown in Figure 3. It consists of 4 regular rounds and a final round that only includes the key mixing and the S-box substitution steps. Like any other SP network, one regular round comprises of three stages: a key mixing step, a substitution layer and a permutation layer. For the key mixing, a simple exclusive or operation is used. The substitution layer is composed of 4 S-boxes with 4-bit inputs and 4-bit outputs[4].

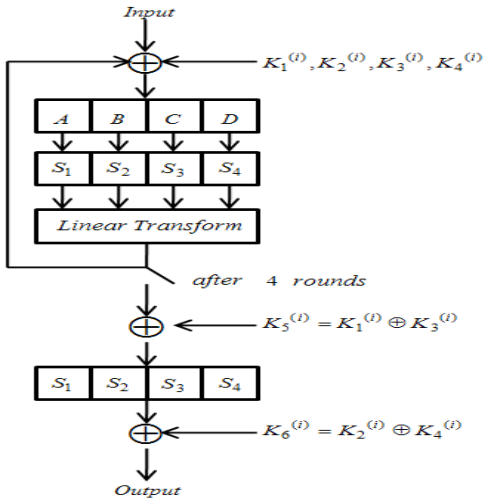

Fig. 3. The structure of Block cipher

TABLE II
16-BIT BLOCK CIPHER ENCRTYPTION, $E_{k_i}(\cdot)$

Input: A 16-bit data block $m = (m_0, m_1, ..., m_{15})$ and a 64-bit subkey k_i, such that subkey $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$

Output: A 16-bit data block $m' = (m'_0, m'_1, ..., m'_{15})$

1: *for* $j = 1$ *to* 4 *do*

2: $m \leftarrow m \oplus K_j^{(i)}$

3: $A = m_0 \| m_1 \| m_2 \| m_3, B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$

4: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$

5: $m \leftarrow m \oplus (m << 6) \oplus (m << 10)$

6: *end for*

7: $m \leftarrow m \oplus K_1^{(i)} \oplus K_3^{(i)}$

8: $A = m_0 \| m_1 \| m_2 \| m_3, B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$

9: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$

10: $m \leftarrow m \oplus K_2^{(i)} \oplus K_4^{(i)}$

11: *return* $m' = (m'_0, m'_1, ..., m'_{15})$

The 16-bit block cipher decryption process is similar to the encryption process, which conducts inverse operation such as S-box substitution and Linear transform.

TABLE III
16-BIT BLOCK CIPHER DECRYPTION, $E_{k_i}(\cdot)$

Input: A 16-bit data block $m' = (m'_0, m'_1, ..., m'_{15})$ and a 64-bit subkey k_i, such that subkey $k_i = K_1^{(i)} \| K_2^{(i)} \| K_3^{(i)} \| K_4^{(i)}$

Output: A 16-bit data block $m = (m_0, m_1, ..., m_{15})$

1: $m \leftarrow m' \oplus K_2^{(i)} \oplus K_4^{(i)}$

2: $A = m_0 \| m_1 \| m_2 \| m_3, B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$

3: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$

4: $m \leftarrow m \oplus K_1^{(i)} \oplus K_3^{(i)}$

5: *for* $j = 1$ *to* 4 *do*

6:
$m \leftarrow m \oplus (m << 2) \oplus (m << 4) \oplus (m << 12) \oplus (m << 14)$

7: $A = m_0 \| m_1 \| m_2 \| m_3, B = m_4 \| m_5 \| m_6 \| m_7$
$C = m_8 \| m_9 \| m_{10} \| m_{11}, D = m_{12} \| m_{13} \| m_{14} \| m_{15}$

8: $m \leftarrow S_1(A) \| S_2(B) \| S_3(C) \| S_4(D)$

9: $m \leftarrow m \oplus K_j^{(i)}$

10: *end for*

11: *return* $m = (m_0, m_1, ..., m_{15})$

4) Linear Transform operation

The permutation layer in this 16-bit block cipher is given by the linear transform defined as follows.

TABLE IV
LINEAR TRANSFORM IN ENCRYPTION

$L : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$
$L(m) = m \oplus (m << 6) \oplus (m << 10)$

In case of decryption, linear transform conducts inverse operation of encryption. For this reason, linear transform is derived by solving the equation related encryption. For example, encryption linear transform conducts exclusive-or operation with the message, 6bit shifted message and 12 bit shifted message. From this equation, we draw the relation between plaintext and cipher text as following equation.

TABLE V
LINEAR TRANSFORM IN DECRYPTION

$L : \{0,1\}^{16} \rightarrow \{0,1\}^{16}$
$L(m) = m \oplus (m << 2) \oplus (m << 4) \oplus (m << 12) \oplus (m << 14)$

5) Notation of S-box

Based on the nine criteria presented in [1], S-boxes are selected, which are listed in TABLE VI.

TABLE VI
S-BOX IN ENCRYPTION

x	$S_1(x)$	$S_2(x)$	$S_3(x)$	$S_4(x)$
0	8	0	2	0
1	6	7	E	7
2	5	E	F	3
3	F	1	5	4
4	1	5	C	C
5	C	B	1	1
6	A	8	9	A
7	9	2	A	F
8	E	3	B	D
9	B	A	4	E
A	2	D	6	6
B	4	6	8	B
C	7	F	0	2

D	0	C	7	8
E	D	4	3	9
F	3	9	D	5

In case of decryption, S-boxes are selected inverse operation of encryption. For example, first value of $S_1(x)$ in TABLE VI is 8. To conduct inverse operation, 8th value of $S_1(x)$ in TABLE VII is chosen to 0. Like this way, S-boxes can be filled with value.

TABLE VII
S-BOX IN DECRYPTION

x	$S_1(x)$	$S_2(x)$	$S_3(x)$	$S_4(x)$
0	D	0	C	0
1	4	3	5	5
2	A	7	0	C
3	F	8	E	2
4	B	E	9	3
5	2	4	3	F
6	1	B	A	A
7	C	1	D	1
8	0	6	B	D
9	7	F	6	E
A	6	9	7	6
B	9	5	8	B
C	5	D	4	4
D	E	A	F	8
E	8	2	1	9
F	3	C	2	7

B. Target Platform : TELOSB MSP430

1) Architecture of MSP430

The Tmote Sky sensor node is equipped with an MSP430 16-bit processor clocked at 8.192MHz. It contains 48 KB of program flash memory and 10KB of RAM. The MSP430 consists 12 general purpose registers and a instruction set with 27 instructions. Determining the number of cycles taken by each instruction is simple because there is no cache : one cycle to fetch the instruction, one cycle to fetch each offset word, one for each memory read and two for memory write. The sensor node is equipped with a 16-bit hardware based multiplier. There are three multiplication mode : MPYU(unsigned multiplication), MPYS(signed multiplication) and MAC(multiply and accumulate) multiplication.[2]. And four addressing modes are available : direct(from registers), indirect(memory address stored in a register), indexed (address stored in a register, plus an offset) and indirect with post increment(which automatically increments the contents of the register holding the address).

2) Development environment of MSP430

An implementation is programmed Tmote sky node equipped MSP430 processor with nesC programming language on TinyOS operation system, and the timing is measured with the software, IAR embedded workbench.

TinyOS is a free and open source component-based operating system and platform targeting wireless sensor network. TinyOS is an equipped operating system written in the nesC programming language as an extension of C language and a set of cooperating tasks and processes.

The resource constrained hardware with limited I/O and debugging abilities combined with the hardware debugging tools makes low-level debugging on the target hardware difficult. To measure the timing, we used IAR embedded workbench which emulates a MSP430 processor environment and tests the assembly code and c code.

3) Instruction set in MSP430 processor

Following is instruction set defined in MSP430 processor and used to optimize the software implementation in proposed algorithms. The operation set is inserted with inline assembly code into nesC language.

TABLE VIII
DESCRIPTION OF INSTRUCTION SET

Mnemonic	Description
add src,dst	Add source to destination
mov src,dst	Move source to destination
swpb dst	Swap bytes in destination
rlc dst	Roll destination left through from carry
rrc dst	Roll destination right through from carry

III. PROPOSED ALGORITHM

A. Definition of S-box structure.

To reduce the memory access, 4-bit S-boxes are combined to 8-bit S-boxes, which are allocated into registers with half clock cycle compared with 4-bit one. Furthermore S-boxes comprised of $S_3(x)$ and $S_4(x)$ are stored in 16-bit variables, whose lower 8-bit is set to zero value, to optimize the allocation process.

TABLE IX
DEFINITION OF MERGED S-BOX TABLE

Definition of 8bit S-box	
$S_1(x) \| S_2(x) = S_{12}(x)$ $(1 \le x \le 256)$	$S_3(x) \| S_4(x) = S_{34}(x)$ $(1 \le x \le 256)$
0x08, 0x78, 0xe8, 0x18, 0x58, 0xb8, 0x88, 0x28, 0x38, 0xa8, 0xd8 …	0x0200, 0x7200, 0x3200, 0x4200, 0xc200, 0x1200, 0xa200, 0xf200…

B. 16-bit block cipher encryption.

TABLE X presents process of key mixing and substitution layer in encryption. Compared to previous one programmed with C language, assembly language is concise and clear featuring acceleration of operation. In step 1, a message is conducted with exclusive-or

operation. After step 2 to step 10, address of s-boxes are computed with index value and starting point of s-boxes then value of s-boxes in memory are allocated into registers.

TABLE X
COMPARISON OF IMPLEMENTATION IN KEY MIXING AND SUBSTITUTION LAYER

C language	Assembly language
1: message = message ^ subkey[j]; 2: a=message&0X000F; 3: b=(unsigned char) ((message&0X00F0)>>4); 4: c=(unsigned char) ((message&0X0F00)>>8); 5: d=(unsigned char) ((message&0XF000)>>12); 6: message = s_box[3][d]<<4; 7: message |= s_box[2][c]; 8: message = message<<8; 9: message |= s_box[1][b]<<4; 10: message |= s_box[0][a];	1: xor r10,r9 2: mov r9,r14 3: mov r9,r15 4: swpb r15 5: and #0x00FF,r14 6: and #0x00FF,r15 7: add r14,r14 8: add r15,r15 9: add %0, r14 10: add %1, r15 11: mov @15, r9 12: swpb r9 13: add @14, r9
r9 :value of message r10 :value of subkey r14 :address of s-box r15 :address of s-box %0 :index of s-box($S_{12}(x)$) %1 :index of s-box($S_{34}(x)$)	

Linear transform described in TABLE XI are efficient as applying the assembly code. To cut the clock cycle, the message is swapped then stored, which is suitable to apply to the linear transform operation required to rotate the register 4 or more times. From step 1 to step 3, value of message is stored into temporal register. Next step, temporal registers (r14, r15) are shifted to each direction then exclusive or operation is executed in r9.

TABLE XI
COMPARISON OF IMPLEMENTATION IN LINEAR TRANSFORM

C language	Assembly language
1: message = message ^ (message<<6) ^ (message>>10) ^ (message<<10) ^ (message>>6);	1: mov r9, r14 2: swpb r14 3: mov r14, r15 4: rlc r14 5: rlc r14 6: rrc r15 7: rrc r15 8: xor r14,r9 9: xor r15,r9
r9 :value of message r14 :temporal register for storing results r15 : temporal register for storing results	

C. Linear Feedback Shift Register.

Linear feedback shift register operation is presented in TABLE XII. In the operation, many shift operations are needed and gaps between purpose bits are negligible so that shift operation is roughly conducted. In step 1, value of l is stored into temporal register r10. From next operation, r10 is shifted to right direction and then exclusive-or operation is conducted with r9 containing value of l.

TABLE XII
COMPARISON OF IMPLEMENTATION IN LINEAR FEEDBACK SHIFT REGISTER

C language	Assembly language
1: bit=((l>>0)^(l>>1)^(l>>4)^(l>>6)^(l>>9)^(l>>13))&1; 2: l = (l>>1) | (bit << 15);	1: mov r9,r10 2: rrc r9 3: xor r9,r10 4: rrc r9 5: rrc r9 6: rrc r9 7: xor r9,r10 8: rrc r9 9: rrc r9 10: xor r9,r10 11: rrc r9 12: rrc r9 13: rrc r9 14: xor r9,r10 15: rrc r9 16: rrc r9 17: rrc r9 18: rrc r9 19: xor r9,r10
r9:value of l r10:temporal register for storing results	

IV. EVALUATION

Our implementation has a superior feature in terms of reduced clock cycle in process including initialization, encryption, and decryption. Table XII shows that comparison between implementations in terms of cycle count reports performance improvements estimated by over 27 percent. This enhancement lies with assembly optimization reducing the clock cycle from previous c language implementation.

TABLE XIII
CYCLE COUNT COMPARISON

	Speed Optimized implementation of Hummingbird[1]	Our implementation
Key Size[bit]	256	256
Block Size[bit]	16	16
Microcontroller	MSP430F1611	MSP430F1611
Init[cycles]	4,824	3,498
Enc[cycles/block]	1,220	871
Dec[cycles/block]	1,461	891

V. CONCLUSION

In this paper, we present speed optimized HUMMINGBIRD cryptography over the sensor node containing limited computational capability, battery and storage capacity. To accelerate the response time

of encryption execution, we apply the assembly code to the algorithm which improves the performance significantly. Furthermore, we also present the HUMMINGBIRD 16-bit decryption cipher which is omitted in previous works. Future works are implementations of space optimized HUMMINGBIRD algorithm and the algorithm on FPGA.

ACKNOWLEDGMENT

"This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MEST) (No.2010-0026621)."

REFERENCES

[1] D.Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: Ultra-Lightweight Cryptography for Resource Constrained Devices", to appear in the Proceedings of The 14th International Conference on Financial Cryptography and Data Security-FC2010, Berlin, Germany: Springer-Verlag, pp. 5-18. 2010.

[2] Texas instruments, "MSP430x11x MIXED SIGNAL MICROCONTROLLERS", 2004.

[3] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C. "PRESENT: An Ultra-Lightweight Block Cipher.," In: Paillier, P., Verbauwhede, I. (eds.) CHES 2007. LNCS, vol. 4727, pp. 450–.466. 2007

[4] Anderson, R., Biham, E., Knudsen, L. "Serpent: A Proposal for the Advanced Encryption Standard," http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf

[5] Engels, D., Fan, X., Gong, G., Hu, H., Smith, E.M."Ultra-Lightweight Cryptography for Low-Cost RFID Tags: Hummingbird Algorithm and Protocol," Centre for Applied Cryptographic Research (CACR) Technical Reports, 2009.

Hwajeong Seo He received the BSEE degree from Pusan National University, Pusan, Republic of Korea in 2010 and he is in MS degree in Computer engineering from Pusan National University. His research interests include sensor network, information security, Elliptic Curve Cryptography and RFID security. He is a member of IEEE.

Howon Kim He received the BSEE degree from Kyungpook National University, Daegu, Republic of Korea, in 1993 and the MS and PhD degrees in electronic and electrical engineering from the Pohang University of Science and Technology (POSTECH), Pohang, Republic of Korea, in 1995 and 1999, respectively. From July 2003 to June 2004, he studied with the COSY group at the Ruhr-University of Bochum, Germany. He was a senior member of technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea. He is currently working as an assistant professor with the Department of Computer Engineering, School of Computer Science and Engineering, Pusan National University, Busan, Republic of Korea. His research interests include RFID technology, sensor networks, information security, and computer architecture. Currently, his main research focus is on mobile RFID technology and sensor networks, public key cryptosystems, and their security issues. He is a member of the IEEE, and the International Association for Cryptologic Research (IACR).