Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 14(3): 177-183, Sep. 2016 **Regular paper**

Efficient Implementation of Simeck Family Block Cipher on 8-Bit Processor

Taehwan Park, Hwajeong Seo, Bongjin Bae, and Howon Kim*, *Member, KIICE*

Department of Computer Engineering, Pusan National University, Busan 46241, Korea

Abstract

A lot of Internet of Things devices has resource-restricted environment, so it is difficult to implement the existing block ciphers such as AES, PRESENT. By this reason, there are lightweight block ciphers, such as SIMON, SPECK, and Simeck, support various block/key sizes. These lightweight block ciphers can support the security on the IoT devices. In this paper, we propose efficient implementation methods and performance results for the Simeck family block cipher proposed in CHES 2015 on an 8-bit ATmega128-based STK600 board. The proposed methods can be adapted in the 8-bit microprocessor environment such as Arduino series which are one of famous devices for IoT application. The optimized on-the-fly (OTF) speed is on average 14.42 times faster and the optimized OTF memory is 1.53 times smaller than those obtained in the previous research. The speed-optimized encryption and the memory-optimized encryption are on average 12.98 times faster and 1.3 times smaller than those obtained in the previous studies, respectively.

Index Terms: ATmega128, Lightweight block cipher, Optimized implementation, Simeck

I. INTRODUCTION

Recently, the development of Internet of Things (IoT) technologies has led to the development of a number of IoT devices. The Arduino series is one of the types of devices used for providing the IoT application services. These devices are based on the 8-bit Atmel AVR series microprocessors and a support network such as Wi-Fi or Bluetooth. For providing security, they use the existing block ciphers. However, low power consumption and small code size are required in a resource-restricted environment such as an 8-bit processor. With respect to these requirements, optimized implementation research has been conducted on the existing block cipher. In 2013, the United States National Security Agency (NSA) proposed the SIMON and SPECK lightweight block ciphers [1] to support various block and key sizes for an efficient implementation in a resource-restricted environment. Thereafter, a significant amount of research has been focused on SIMON and SPECK. The Simeck lightweight block cipher was proposed in CHES 2015. Its design is similar to that of SIMON and SPECK in that it is in fact a combination of the SIMON and SPECK designs. Simeck is suitable for RFID systems and is optimized for hardware implementations. In this paper, we propose efficient implementation methods for the Simeck family block cipher in an 8-bit processor environment (Atmel STK 600 board based on ATmega128).

The remainder of this paper is organized as follows: Section II describes the Simeck family block cipher and discusses the previous literature related to implementing the cipher in an 8-bit processor environment. We suggest an efficiently optimized implementation of the Simeck family

Received 09 August 2016, **Revised** 10 August 2016, **Accepted** 11 August 2016
***Corresponding Author** Howon Kim (E-mail:howonkim@pusan.ac.kr, Tel: +82-51-510-1010)
Department of Computer Engineering, Pusan National University, 2, Busandaehak-ro 63beon-gil, Geumjeong-gu, Busan 46241, Korea.

Open Access **http://dx.doi.org/10.6109/jicce.2016.14.3.177** print ISSN: 2234-8255 online ISSN: 2234-8883

block cipher on an 8-bit processor in Section III. Section IV describes the experimental environment, the performance results of the proposed method, and the results of the comparison between the proposed method and the reference code of the Simeck family block cipher. Section V provides some final conclusions.

II. BACKGROUND AND RELATED WORK

A. Simeck Family Block Cipher

In this subsection, we describe the Simeck family block cipher proposed in CHES 2015. The Simeck family block cipher is an optimized hardware environment and is suitable for an RFID system [2].

The Simeck family block cipher's key schedule, encryption, and decryption consist of bitwise eXclusive-OR (XOR) ($\oplus$), bitwise AND ($\odot$), and c-bit rotation left and right (<<< and >>>) operations. The Simeck family block cipher can be divided according to the block/key size as Simeck32/64 (input/output size: 32 and key size: 64), Simeck48/96 (input/output size: 48 and key size: 96), and Simeck64/128 (input/output size: 64 and key size: 128). Table 1 shows that the Simeck family block cipher supports various block sizes (bit) and round numbers.

Fig. 1 shows the Simeck family block cipher's i-th encryption round function. The Simeck encryption round function consists of 1 bitwise AND operation ($\odot$), 3 bitwise XOR operations ($\oplus$), and a 5-bit rotation left and a 1-bit rotation left operation (<<<5 and <<<1).

The Simeck encryption round function can be written as follows, where $\boldsymbol{l_i}$ and $\boldsymbol{r_i}$ denote the two words for the initial state and $\boldsymbol{k_i}$ represents the i-th round key:

$$\boldsymbol{R_{k_i}(l_i, r_i) = (r_i \oplus f(l_i) \oplus k_i.} \quad (1)$$

$$\mathbf{f(x) = (x \odot (x \lll 5)) \oplus (x \lll 1).} \quad (2)$$

The Simeck key schedule and expansion function can generate round keys from the master key K.

The Simeck key schedule and expansion function can be written as following equations and is illustrated in Fig. 2. This figure shows the Simeck key schedule and expansion function equation as a block diagram.

Table 1. Simeck family block cipher

Cipher	Block size (bit)	Round (T)
Simeck32/64	16	32
Simeck48/96	24	36
Simeck64/128	32	44

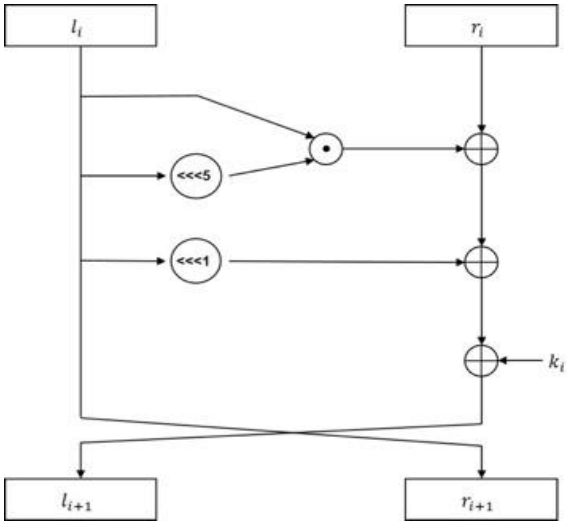


Fig. 1. Simeck encryption round function.

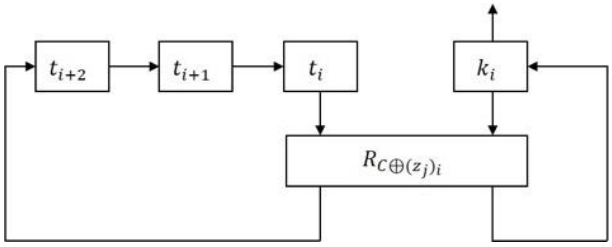


Fig. 2. Simeck key schedule.

The Simeck master key K consists of four words as the initial state $(\boldsymbol{t_2, t_1, t_0, k_0})$. The constant (C) used in the key schedule and expansion can be written as $\boldsymbol{2^n - 4 = 0XFF \ldots FC}$.

$\boldsymbol{(Z_j)_i}$ denotes the i-th bit of sequence $\boldsymbol{Z_j}$. In the case of Simeck32/64 and Simeck48/96, use $\boldsymbol{Z_0}$ whose period is 31; this can be generated by using the primitive polynomial $\boldsymbol{x^5 + x^2 + 1}$ and the initial state (1, 1, 1, 1, 1). Simeck64/128 uses $\boldsymbol{Z_1}$ whose period is 63; this can be generated by using the primitive polynomial $\boldsymbol{x^6 + x + 1}$ and the initial state (1, 1, 1, 1, 1, 1).

B. Literature Reviews

In this subsection, we describe the recent research on the Simeck family block cipher and the cipher implementation in an 8-bit processor environment.

First, the Stefan method [3] describes the features of the Simeck family block cipher architecture and the differential analysis of the Simeck by comparing the Simeck family block cipher and the SIMON block cipher. The Nasour method [4] proposes a linear analysis of the round-reduced Simeck block cipher and describes a possible linear attack

on Simeck32/64, Simeck48/96, and Simeck64/128 at rounds 13, 19, and 22, respectively. Kexin et al. [5] implemented a differential attack on the Simeck block cipher by reducing the key space used in key guessing and described a differential attack on Simeck32/64, Simeck48/96, and Simeck64/128 at rounds 22, 28, and 35, respectively. Zhang et al. [6] conducted a linear analysis of the Simeck by using a zero-correlation linear distinguisher, and implemented a linear attack on Simeck32/64, Simeck48/96, and Simeck64 /128 at rounds 11, 12, and 15, respectively. Yoshikawa et al. [7] implemented a power analysis attack on the FPGA hardware implementation of the Simeck block cipher and described the results of the power analysis attack. The Lingyue method [8] can find a differential that has a high probability and a low Hamming weight of the Simeck block cipher by using Kolb's tool and implemented a linear hullback attack on Simeck32/64, Simeck48/96, and Simeck64/128 at rounds 23, 30, and 37, respectively, by using dynamic key-guessing techniques.

Next, we present the results of an implementation of the cipher in an 8-bit processor environment.

Beaulieu et al. [1] implemented the SIMON and SPECK block ciphers on an 8-bit AVR processor and reported the results of this implementation. Liu et al. [9] implemented the ring-LWE encryption scheme on an 8-bit AVR processor. They reported that the performance of ring-LWE on AVR is better than that of RSA and ECC on AVR.

Buchmann et al. [10] efficiently implemented lattice-based public key encryption on 8-bit ATXmega128 and 32-bit Cortex-M0 and reported the implementation results. Poppelmann et al. [11] carried out a speed-optimized implementation of ring-LWE and BLISS signature on 8-bit Atmel ATxmega128. Dinu et al. [12] compared the performance of the operation mode (CBC and CTR) of various block ciphers such as AES, HIGHT, LED, SIMON, and SPECK in an 8-bit ATmega environment. Seo et al. [13] carried out a compact implementation of the LEA block cipher on Atmel 8-bit AVR.

III. PROPOSED METHODS

In this section, we propose efficient implementation methods of the Simeck block cipher in an 8-bit ATmega128 environment. Table 2 presents some AVR assembly language commands for the XOR and rotation operations.

Atmel ATmega128 supports the *EOR* assembly command for an XOR operation between two 8-bit registers. However, in the case of a rotation operation, Atmel ATmega128 supports rotation in 8-bit register assembly commands (*ROR* and *ROL*), shift operation assembly commands (*LSR* and *LSL*), the clear the register assembly command (*CLR*), and the addition with carry operation assembly command (*ADC*).

Table 2. Efficient XOR and rotation on AVR

XOR	Right rotation	Left rotation
EOR R12, R20	*CLR R20*	*LDI R20,0x00*
EOR R13, R21	*LSR R15*	*LSL R15*
EOR R14, R22	*ROR R14*	*ROL R14*
EOR R15, R23	*EOR R15,R20*	*ADC R15,R20*

For an efficient rotation operation, we divide a data block into several pieces of 8-bit data and use an additional 8-bit register to save the shift bit value. After a shift or rotation command operation, we conduct an XOR operation (for right rotation) or an addition with carry operation (for left rotation).

Algorithm 1. Efficient Shift Offset and Direction in ATmega128 [13]

Require: direction **d**, offset **o**
Ensure: direction **d**, offset **o**
1: $\mathbf{o} = \mathbf{o} \bmod 8$
2: if $\mathbf{o} > 4$ **then**
3: $\mathbf{o} = 8 - \mathbf{o}, \mathbf{d} = !\mathbf{d}$
4: return **d**, **o**

Algorithm 1 [13] describes an efficient shift offset and direction in Atmega128.

For an efficient shift or rotation operation, we check whether the result of offset **o** modulo 8 is greater than 4. If it is greater than 4, then we set offset **o** as 8 - **o** and change the direction **d**. It can save the rotation operation cost by reducing the shift bit value and changing the direction.

From the Simeck block cipher encryption round function structural aspects, the Simeck block cipher encryption round function can be written as Eqs. (1) and (2). It requires the 5-bit Rotation Left, and the 1-bit Rotation Left of the l_i value follows a certain order. However, in this study, we can reduce the 5-bit Rotation Left 1 time and the 1-bit Rotation Left 1 time as just the 5-bit Rotation Left 1 time by the 1-bit Rotation Left value during the 5-bit Rotation Left operation of the l_i value. It can optimize the speed and the code size of the Simeck implementation in the 8-bit ATmega128 environment.

In Fig. 3, the red box denotes the 5-bit Rotation Left and the 1-bit Rotation Left of the l_i value parts. The gray box represents the pre-computed parts. The pre-computed parts consist of the XOR operation, and the operation order is as shown in Fig. 3. However, XOR operations (pre-computed available part in Fig. 3) can be pre-computed because the required operation is an XOR operation (it does not need to be executed in the specified order).

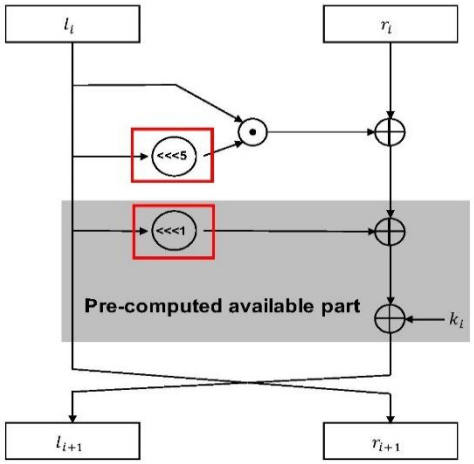


Fig. 3. Pre-computed available part.

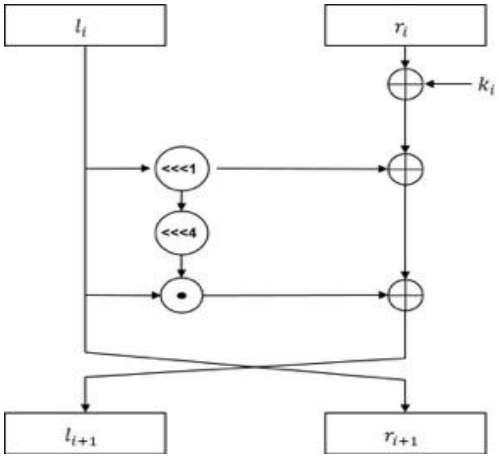


Fig. 4. Re-organized Simeck block cipher encryption round function.

We optimized the speed and the code size of the Simeck implementation by pre-computing the gray part shown in Fig. 3 and using the 1-bit Rotation Left operation result of $\boldsymbol{l_i}$ in the process of the 5-bit Rotation Left operation of $\boldsymbol{l_i}$. Therefore, we re-organized the Simeck family block cipher encryption round function as shown in Fig. 4.

In the case of code size optimization, we used for loop to reuse the common encryption round function and used loop unrolling for speed optimization (this can save the additional cost in the for loop operation).

IV. EXPERIMENT AND EVALUATION

In this section, we describe the experimental environment, procedures, and analysis of the performance of the proposed method. For the evaluation, we implemented the on-the-fly (OTF) method of Simeck32/64 and Simeck48/96. We cannot implement the OTF version of Simeck64/128 because ATmega128 supports 32 8-bit general-purpose registers but Simeck64/128 needs more than 32 registers for the encryption round function.

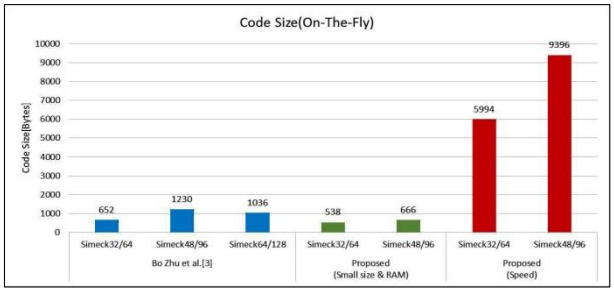
A. Experimental Setup

We used the ATmel STK-600 device, which is based on ATmega128, for conducting the experiment.

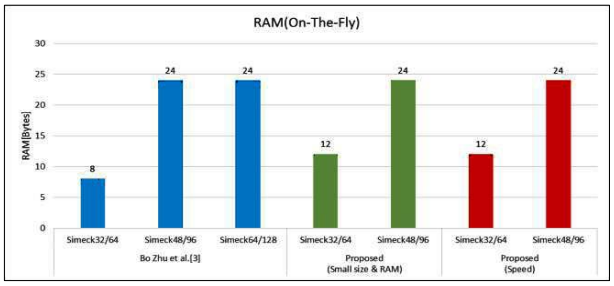
Table 3 lists the specifications of the ATmel STK-600 board. The ATmel STK-600 board has 128-kB app/boot memory, 4,096-byte data memory, and 4,096-byte EEPROM.

Table 3. Specifications of ATmel STK-600

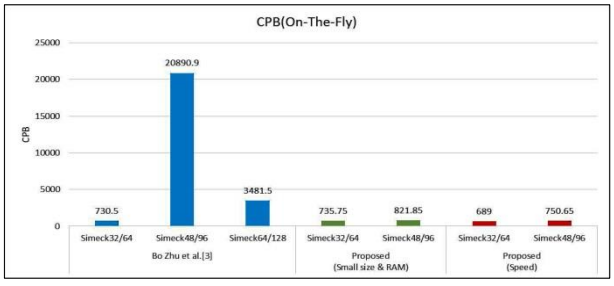
CPU	Atmel ATmega128
App/Boot memory	128 kB
Data memory	4,096 bytes
EEPROM	4,096 bytes



(a)



(b)



(c)

Fig. 5. Simeck block cipher OTF implementation comparison performance results. (a) Code size, (b) RAM, (c) CPB.

Table 4. Performance comparison results

	Method	Cipher	Code size (bytes)	RAM (bytes)	CPB
OTF & small size & RAM	Zhu [3]	Simeck32/64	652	8	730.5
		Simeck48/96	1230	24	20890.9
		Simeck64/128	1036	24	3481.5
	Proposed	Simeck32/64	538	12	735.75
		Simeck48/96	666	24	821.85
		Simeck64/128	-	-	-
OTF & speed	Proposed	Simeck32/64	5994	12	689
		Simeck48/96	9396	24	750.65
		Simeck64/128	-	-	-
Encryption & small size & RAM	Encryption-based Zhu [3]	Simeck32/64	694	10	458
		Simeck48/96	1404	8	10424.85
		Simeck64/128	1020	8	1553.9
	Proposed	Simeck32/64	432	12	281.5
		Simeck48/96	944	8	401.85
		Simeck64/128	954	24	406.9
Encryption & speed	Proposed	Simeck32/64	2482	12	249
		Simeck48/96	9812	184	318
		Simeck64/128	5914	24	361

CPB: cycles per byte.

B. Evaluation

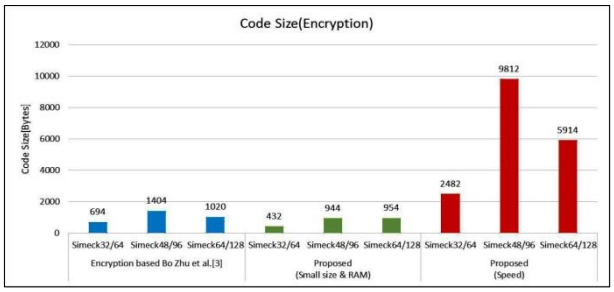
We implemented the proposed methods on Atmel Studio 7 (version 7.0.943) and used the AVR assembly language for the optimization. For the performance comparison, we compared the proposed methods and that proposed by Zhu [14], which is the Simeck block cipher OTF reference code on GitHub. In the case of the encryption part, we compared the AVR assembly version of the proposed method and the encryption part from the method proposed by Zhu [14].

Table 4 and Figs. 5, 6 describe the results of a performance comparison between the method proposed by Zhu [14] and that proposed in this paper, in terms of the code size (bytes), RAM (bytes) and cycles/byte.

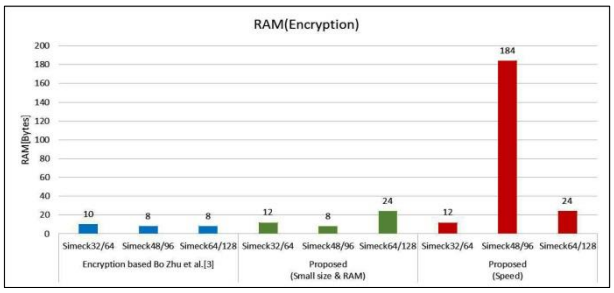
The Simeck64/128 OTF implementation needs 8-bit registers more than 8-bit registers supported by Atmel Atmega128 for key expansion and encryption, so we cannot implement the Simeck64/128 OTF-optimized version.

The proposed OTF speed-optimized version is on average 14.42 times faster than Zhu [14].

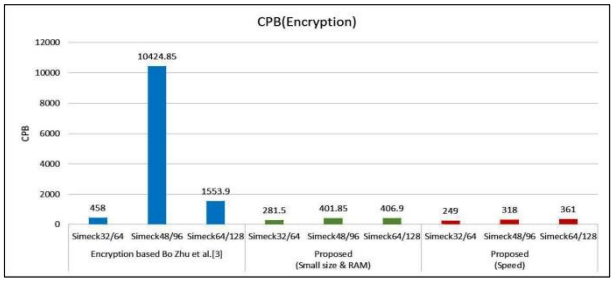
The required memory at the OTF memory-optimized version based on the proposed method is 1.53 times smaller than that required by the method proposed by Zhu [14]. The speed-optimized encryption and memory-optimized encryption based on the proposed method are on average 12.98 times faster and 1.3 times smaller than the encryption part of the method proposed by Zhu [14].



(a)



(b)



(c)

Fig. 6. Simeck block cipher encryption implementation comparison performance results. (a) Code size, (b) RAM, (c) CPB.

V. CONCLUSION

In this paper, we proposed an efficient implementation method of the Simeck block cipher proposed in CHES 2015 in terms of the code size and speed optimization on an 8-bit ATmega128 device.

For the code size/speed-optimized implementation, we proposed an efficient rotation operation by using the AVR assembly command and an efficient shift offset and direction algorithm. For the code size optimization, we used a for loop operation in the encryption round function, but we used loop unrolling for speed optimization (to reduce the cost of the for loop operation).

The OTF speed-optimized implementation is on average 14.42 times faster and the OTF memory-optimized implementation is 1.53 times smaller than the method proposed by Zhu [14]. The speed-optimized and the memory-optimized encryption implementations are on average 12.98 times faster and 1.3 times smaller than the encryption part of the method proposed by Zhu [14].

These results will be useful for applying the Simeck block cipher to various IoT application services.

In the future, we intend to study the Simeck family block cipher code size/speed-optimized implementation and secure implementation against side-channel attacks on various hardware platforms such as MSP-430 and ARM.

ACKNOWLEDGMENTS

This work was supported by a 2-year Research Grant of Pusan National University.

REFERENCES

[1] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, “The SIMON and SPECK lightweight block ciphers,” in *Proceedings of the 52nd Annual Design Automation Conference*, San Francisco, CA, 2015.

[2] G. Yang, B. Zhu, V. Suder, M. D. Aagaard, and G. Gong, “The Simeck family of lightweight block ciphers,” in *International Workshop on Cryptographic Hardware and Embedded Systems*. Heidelberg: Springer, pp. 307-329, 2015.

[3] S. Kolbl and A. Roy, “A brief comparison of SIMON and SIMECK,” 2015 [Internet]. Available: http://eprint.iacr.org/2015/706.pdf.

[4] N. Bagheri, “Linear cryptanalysis of reduced-round SIMECK variants” in *Proceedings of 16th International Conference in Cryptology in India*, Bangalore, India, pp. 140-152, 2015.

[5] K. Qiao, L. Hu, and S. Sun, “Differential analysis on simeck and simon with dynamic key-guessing techniques,” 2015 [Internet]. Available: https://eprint.iacr.org/2015/902.pdf.

[6] K. Zhang, J. Guan, B. Hu, and D. Lin, “Security evaluation on simeck against zero correlation linear cryptanalysis,” 2015 [Internet]. Available: https://eprint.iacr.org/2015/911.pdf.

[7] M. Yoshikawa, Y. Nozaki, and K. Asahi, “Multiple rounds aware power analysis attack for a lightweight cipher SIMECK,” in *Proceedings of 2016 IEEE 2nd International Conference on Big Data Computing Service and Applications (BigDataService)*, Oxford, UK, pp. 252-256, 2016.

[8] L. Qin, H. Chen, and X. Wang, “Linear hull attack on round-reduced Simeck with dynamic key-guessing techniques,” 2016 [Internet]. Available: https://eprint.iacr.org/2016/066.pdf.

[9] Z. Liu, H. Seo, S. S. Roy, J. Großschädl, and H. Kim, and I. Verbauwhede, “Efficient ring-LWE encryption on 8-bit AVR processors,” in *Proceedings of 17th International Workshop on Cryptographic Hardware and Embedded Systems*, Saint-Malo, France, pp. 663-682, 2015.

[10] J. Buchmann, F. Gopfert, T. Guneysu, T. Oder, and T. Poppelmann, “High-performance and lightweight lattice-based public-key encryption,” in *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security*, Xian, China, pp. 2-9, 2016.

[11] T. Poppelmann, T. Oder, and T. Guneysu, “Speed records for ideal lattice-based cryptography on AVR,” 2015 [Internet]. Available: http://eprint.iacr.org/2015/382/20150428:235531.

[12] D. Dinu, Y. Le Corre, D. Khovratovich, L. Perrin, J. Großschadl, and A. Biryukov, “Triathlon of lightweight block ciphers for the Internet of Things,” 2015 [Internet]. Available: https://eprint.iacr.org/2015/209.pdf.

[13] H. Seo, Z. Liu, J. Choi, T. Park, and H. Kim, “Compact Implementations of LEA block cipher for low-end microprocessors,” in *Proceedings of 16th International Workshop on Information Security Applications*, Jeju, Korea, pp. 28-40, 2015.

[14] B. Zhu, “Reference code for Simeck family of block ciphers,” 2015 [Internet]. Available: https://github.com/bozhu/Simeck.

Taehwan Park
received his B.S.E.E. from Pusan National University, Pusan, Republic of Korea, in 2013. He is currently pursuing the combined M.S. and Ph.D. course in Computer Engineering at Pusan National University. His research interests include IoT device security, information security, elliptic curve cryptography, and post quantum cryptography.

Hwajeong Seo
received his B.S.E.E. from Pusan National University, Pusan, Republic of Korea, in 2010. He also received his M.S. and Ph.D. in Computer Engineering from the same university. His research interests include sensor networks, information security, elliptic curve cryptography, and RFID security.

Bongjin Bae
received his B.S.E.E. from Pusan National University, Pusan, Republic of Korea, in 2015. Currently, he is pursuing his master's degree at Pusan National University. His research interests include IoT, information security, and post quantum cryptography.

Howon Kim
received his B.S.E.E. from Kyungpook National University, Daegu, Republic of Korea, in 1993, and his M.S. and Ph.D. in Electronic and Electrical Engineering from 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 the technical staff at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, Republic of Korea. He is currently working as an associate 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 the Internet of Things (IoT) technology and the related security issues, mobile RFID technology, and sensor networks, public key cryptosystems, post quantum cryptography, and their security issues. He is a member of the IEEE.