Hindawi
Journal of Sensors
Volume 2018, Article ID 6492592, 10 pages
https://doi.org/10.1155/2018/6492592

Research Article

Secure Data Encryption for Cloud-Based Human Care Services

Taehwan Park,[1] **Hwajeong Seo**,[2] **Sokjoon Lee**,[3] **and Howon Kim**[1]

[1]*School of Computer Science and Engineering, Pusan National University, San-30, Jangjeon-dong, Geumjeong-gu, Busan 609-735, Republic of Korea*
[2]*IT Engineering, Hansung University, 116 Samseong-Yoro-16-Gil, Seongbuk-gu, Seoul 136-792, Republic of Korea*
[3]*System Security Research Group, Electronics and Telecommunications Research Institute, Daejeon 34129, Republic of Korea*

Correspondence should be addressed to Howon Kim; howonkim@pusan.ac.kr

Received 22 February 2018; Revised 9 June 2018; Accepted 4 July 2018; Published 6 August 2018

Academic Editor: Mucheol Kim

Sensor network services utilize sensor data from low-end IoT devices of the types widely deployed over long distances. After the collection of sensor data, the data is delivered to the cloud server, which processes it to extract useful information. Given that the data may contain sensitive and private information, it should be encrypted and exchanged through the network to ensure integrity and confidentiality. Under these circumstances, a cloud server should provide high-speed data encryption without a loss of availability. In this paper, we propose efficient parallel implementations of Simeck family block ciphers on modern 64-bit Intel processors. In order to accelerate the performance, an adaptive encryption technique is also exploited for load balancing of the resulting big data. Finally, the proposed implementations achieved 3.5 cycles/byte and 4.6 cycles/byte for Simeck32/64 and Simeck64/128 encryption, respectively.

1. Introduction

At present, numerous human care services are used in hospitals and clinics according to the development of IoT (Internet of Things) technologies. To provide human care services in the United States, a service provider must follow the HIPAA (Health Insurance Portability and Accountability Act). The purpose of the HIPAA is to protect the privacy of medical information. For this reason, the U.S. Department of Health and Human Services (HHS) has stipulated that human care service providers must follow the HIPAA to enhance the protection of patients' private and health information. In the *HIPAA Compliance Guide* [1], the service provider must use data encryption on portable devices and computer networks during secure PHI (protected health information) transmission and storage processes. However, various types of devices, such as 8-bit, 16-bit, and 32-bit IoT devices as well as communication protocols such as Bluetooth and Wi-Fi, are used to provide human care services. These IoT devices for human care services operate in a resource-constrained environment. For this reason, existing block ciphers use a SPN (substitution and permutation network) architecture such as AES [2], which requires a considerable amount of memory to save S-boxes and the round keys and operations. To address this problem, numerous lightweight block ciphers use ARX (addition/AND, rotation, eXclusive-OR) operations such as SIMON, SPECK [3], or Simeck [4]. A Simeck family block cipher [4] is suitable for RFID (radio-frequency identification) sensor environments, though it has not been considered by any standardization institute to be included in a standard. However, the Simeck family block cipher [4] is lightweight according to ARX (AND, rotation, eXclusive-OR) operations and supports various block/key sizes. Moreover, it is suitable for RFID, sensor devices, and IoT end devices, all of which have resource-restricted environments. If Simeck family block ciphers are used in the end devices or sensors used in conjunction with various human care application services, cloud or service platform servers must deal with encrypted big data from various end devices or sensors in their human care application services as rapidly as possible to ensure the availability of their human care application services. At that time, the size of the big data can differ

Table 1: Simeck family block cipher specification.

Block cipher	Block size (bit)	Key size (bit)	Round (T)
Simeck32/64	32	64	32
Simeck48/96	48	96	36
Simeck64/128	64	128	44

according to the data transmission cycle of the end devices. To process big data on a cloud or service platform, the cloud or service platform servers must have high-performance modern 64-bit processors (high-frequency processors which are capable of supporting SIMD (single instruction multiple data)) such as an Intel Xeon processor or an i7 series processor for human care application services. To address these issues, in this paper we propose efficient parallel implementation methods of the Simeck family block cipher [4] using an Intel AVX2 (Advanced Vector Extension 2) SIMD and an efficient adaptive encryption method to enhance human care service availability based on Simeck family block cipher AVX2-optimized implementations which support various data block sizes.

The remainder of this paper is organized as follows. Section 2 discusses previous works related to the Simeck family block cipher, AVX2, and human care service security. We propose the efficient AVX2-optimized Simeck implementation and adaptive encryption method for big data encryption for use within a human care service in Section 3. Section 4 provides experimental and evaluation results from the proposed AVX2-optimized Simeck implementation and adaptive encryption approach. Section 5 provides the conclusion.

2. Related Works

In this section, we describe works related to Simeck family block cipher and Intel AVX2 SIMD and works related to human care service security.

2.1. Simeck Family Block Cipher. The Simeck family block cipher was proposed at CHES 2015 [4], and it succeeded the architecture of SIMON and SPECK [3] and therefore has similar encryption round functions with different numbers of rotation operations and bit rotation operations. The purpose of Simeck is suitability for use with a lightweight block cipher for hardware environments and RFID systems. There are three types of Simeck family block cipher, that is, Simeck32/64, Simeck48/96, and Simeck64/128, which have 32, 36, and 44 rounds, respectively. Table 1 presents the Simeck family block cipher specifications, including the block/key size (bit) and the number of rounds.

The Simeck family block cipher encryption and decryption round functions have ARX (AND (⊙), rotation (rotation left, (<<<)), eXclusive-OR (⊕)) operations. Figure 1 shows the encryption round function of the Simeck block cipher at the ith round. In Figure 1, l_i denotes the left word, r_i denotes the right word during the ith round, and k_i denotes

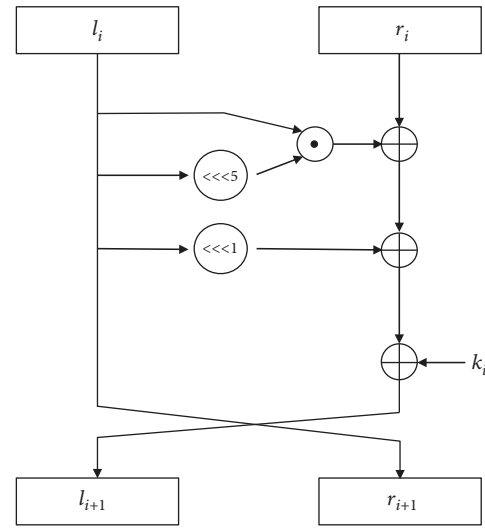

Figure 1: Simeck family block cipher encryption round function.

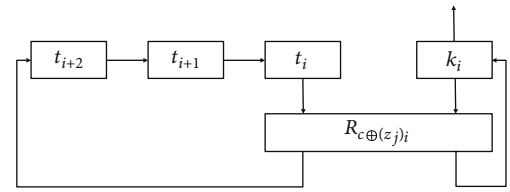

Figure 2: Simeck family block cipher key expansion.

the ith round key. The ith round function of the Simeck block cipher can be represented by the following equation:

$$R_{k_i}(l_i, r_i) = (r_i \oplus f(l_i) \oplus k_i, l_i),$$
$$f(x) = (x \odot \mathrm{ROL1}(x)) \oplus \mathrm{ROL5}(x). \quad (1)$$

In the equation of the Simeck encryption round function, the ROLr () function refers to the r-bit left rotation operation. It is expressed as <<<r in Figure 1. The Simeck encryption round function consists of 1 AND (⊙), 2 rotation left (<<<), and 3 eXclusive-OR (⊕) operations during each round.

In the Simeck family block cipher key schedule, Figure 2 shows the Simeck family block cipher key schedule as a block diagram. The key schedule of the Simeck family block cipher is similar to the key schedule procedure of the SIMON and SPECK family block ciphers [3]. $R_{C\oplus(z_j)_i}$ in Figure 2 is the Simeck round function with $C \oplus (z_j)_i$, which acts as the round key during each round. The round key k_i is generated from the master key K, which is initially segmented into four words and which loads as the initial states (t_2, t_1, t_0, k_0) of the feedback register architecture, as shown in Figure 2. The Simeck family block cipher key schedule uses initial states such as (1, 1, 1, 1, 1) and (1, 1, 1, 1, 1, 1). The first initial state is used with Simeck32/64 and Simeck48/96, and the second initial state is used with Simeck64/128. With regard to k_0, it is the least significant n bits of the master key K. To update the register values and generate a round key during each

round, it uses a round function with a round constant of $C \oplus (z_j)_i$ which is the bitwise eXclusive-OR result between the constant C and the i-bit of sequence Z_j, which can also act as the round key, that is, $R_{C\oplus(z_j)_i}$. The constant value (C) in the key schedule can be expressed as $2^n - 4 = 0 \times \mathrm{FF}, \ldots, \mathrm{FC}$. The $(z_j)_i$ in Figure 2 refers to the i-bit of sequence Z_j. There are two sequences: Z_0 and Z_1. Sequence Z_0 has 31 periods and can be generated using the primitive polynomial $x^5 + x^2 + 1$, while sequence Z_1 has 63 periods and can be generated using the primitive polynomial $x^6 + x + 1$. Sequence Z_0 is used with Simeck32/64 and Simeck48/96, and sequence Z_1 is used with Simeck64/128 for the Simeck family block cipher key schedule. The updating operation can be expressed as follows:

$$\begin{aligned} k_i &= t_i, \\ t_{i+3} &= k_i \oplus f(t_i) \oplus C \oplus (z_j)_i, \\ &\qquad 0 \le i \le T - 1. \end{aligned} \tag{2}$$

2.2. Related Works on Simeck Family Block Cipher. Related works which focus on the Simeck family block cipher can be divided into three types: cryptanalysis, side-channel attacks, and efficient implementations.

The first type involved cryptanalyses of the Simeck family block cipher. Kölbl and Roy [5] presented brief comparison results between SIMON and Simeck. Bagheri [6] proposed a linear cryptanalysis method and results with the reduced-round Simeck block cipher. Qiao et al. [7] proposed a differential analysis of Simeck using dynamic key-guessing techniques. Zhang et al. [8] proposed a zero-correlation linear cryptanalysis of Simeck. Wang [9] proposed a related-key differential analysis of the round-reduced Simeck block cipher based on mixed-integer linear programming. They found that a 14-round related-key differential distinguisher for Simeck32/64 is the best known method, and they used the dependencies of bitwise AND operations. Sadeghi and Bagheri [10] proposed an improved miss-in-the-middle approach to find zero-correlation linear distinguishers and impossible differentials on Simeck48 and Simeck64. They attacked 15-round Simeck48 and 17-round Simeck64 using their proposed zero-correlation linear approximation method which relies on the duality of the zero-correlation and impossible differential, and they attacked 27-round Simeck48 and 31-round Simeck64 based on their proposed zero-correlation linear distinguishers. Moreover, they proposed impossible differential attacks on 22-round Simeck48 and 24-round Simeck64 based on the impossible differential characteristics of Simeck.

The second category is side-channel attacks on the Simeck family block cipher. Qin et al. [11] proposed a linear hull attack with dynamic key-guessing techniques on round-reduced Simeck, similar to an approach by Qiao et al. [7]. Ryabko and Soskov [12] proposed a distinguishing attack on several lightweight block ciphers, including Simeck. Yoshikawa et al. [13] proposed a multiple-round-aware power analysis attack method and attack results on Simeck. Nozaki et al. [14] proposed an electromagnetic analysis of Simeck FPGA implementation using the linear relationship between the Hamming distance and electromagnetic waves, similar to DEMA and CEMA. Nalla et al. [15] proposed a differential fault attack on Simeck. They conducted random bit-flip fault attacks (requiring $n/2$ faults to recover the n-bit last round key) and a random byte fault attack (needing $n/6.5$ faults to recover the n-bit last round key) on Simeck. Nozaki et al. [16] proposed a double-round-driven electromagnetic analysis attack on Simeck FPGA implementation.

The final category contains methods related to the implementation of the Simeck block cipher. There are two efficient software implementations of Simeck on IoT-embedded devices: 8-bit AVR [17] and 16-bit MSP430 [18]. Simeck-based permutations for lightweight sponge cryptographic primitive hardware implementation were proposed at SAC 2017 [19].

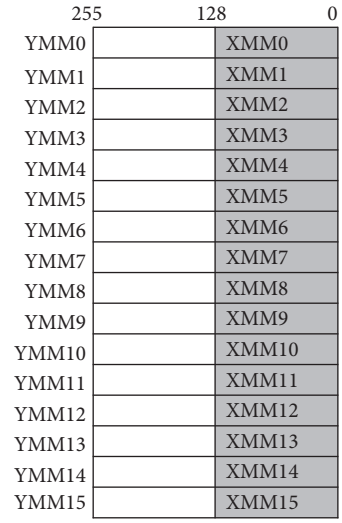

Figure 3: AVX2 registers.

2.3. Intel AVX2 SIMD. Intel AVX2 means Advanced Vector Extension 2 [20]. Intel has been providing AVX2 (Advanced Vector Extension 2) since the Intel Haswell architecture (Q2 2013). For AMD CPUs, the AMD Excavator processor (Q2 2015) and the Zen processor (Q1 2017) support AVX2. The main difference between AVX and AVX2 is that AVX2 supports 16 256-bit registers (YMM0–YMM16), as shown in Figure 3, with three-operand general-purpose bit manipulation and multiplication. Specifically, since the development of AVX2, vector shift operations are supported. If we want to use a rotation operation for an ARX-based block cipher, AVX2 shift operations can easily be used.

AVX2 registers of 256 bits can be written as 8-bit × 32, 16-bit × 16, and 32-bit × 8 for SIMD (single instruction multiple data), as shown in Figure 4. If we use a SIMD operation, we can calculate multiple instances of data simultaneously, implying that this approach can be used for multimedia processing. AVX2 SIMD supports intrinsic functions which correspond to AVX2 assembly instructions, and it is a type

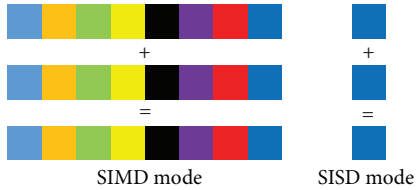

Figure 4: SIMD and SISD.

of API (application program interface). Accordingly, a developer can easily use Intel AVX2.

2.4. Related Works on Cryptographic Algorithm Implementation Based on AVX2. There are numerous research results pertaining to the implementation of cryptographic algorithms using AVX2.

Gueron and Krasnov [21] proposed multiprime RSA implementation methods and results using AVX2. In particular, they parallelized r modular exponentiations on RSA. Faz-Hernández and López [22] utilized efficient arithmetic operations on the prime field using AVX2, with performance benchmarked on the Intel Haswell processor. Faz-Hernández and López [23] proposed an efficient implementation of an elliptic curve (Curve25519) using AVX2. They proposed an accelerated prime field and elliptic curve arithmetic using AVX2. The Martins Paulo method [24] involves optimized fully homomorphic encryption implementation methods and results using AVX2. Cabral and López [25] proposed parallel SHA-3 family implementation methods and results using AVX2. They parallelized four digests from four different messages.

At present, AVX2 is used for implementing postquantum cryptography (PQC). Du et al. [26] proposed an efficient optimized number-theoretic transform (NTT) and high-precision discrete Gaussian sampler implementation methods and performance results for lattice-based public-key encryption using AVX2. Gueron and Schlieker [27] proposed optimized NTRUEncrypt implementation methods and results using AVX2. They also proposed replacing the SHA hash functions by pipelined AES-NI (Advanced Encryption Standard-New Instructions) for the rapid generation of randomness. Hamburg [28] proposed a new cryptosystem based on *integer module learning with errors* (I-MLWE), which uses the integers modulo ring with a generalized Mersenne prime number. In his paper, he proposed efficient software implementation methods and results using AVX2. Steinfeld et al. [29] proposed titanium, postquantum public-key encryption, and KEM. They proposed AVX2-optimized implementation methods and results with titanium. In PQCrypto 2017, fast lattice-based encryption SPRING implementation methods and results using AVX2 were posted. The security of lattice-based encryption SPRING relies on the hardness of the learning-with-rounding (LWR) problem. In these PQC implementation results using AVX2, the Hamburg method [28] and the approach developed by Steinfeld et al. [29] are focused on the NIST (National Institute of Standards and Technology) PQC standard competition. Accordingly, the efficient software implementation of a postquantum cryptographic algorithm using AVX2 is one of the important aspects of the NIST PQC standard competition.

Specifically, eBACs [30] offers SUPERCOP (System for Unified Performance Evaluation Related to Cryptographic Operations and Primitives), which measures the performance of hash functions, secret-key streams, public-key encryption, and other functions on modern processors using SSE, AVX, and AVX2.

2.5. Related Works on Human Care Service Security. There are two areas of related works on human care service security. The first is the security of commercial human care services, and the second area consists of research results on human care service security.

With regard to commercial human care services, Microsoft Azure, IBM Watson, and Amazon Web Services (AWS) are widely used. Microsoft published the "Microsoft Azure HIPAA/HITECH Act Implementation Guidance" publication [31]. MS Guidance follows the HIPPA/HITECH act and stores encryption keys separately. It also supports encryption-at-rest using .NET cryptographic services and SQL server encryption, including transparent data encryption with the Azure SQL database of PHI data. For communication security, SSL and TLS 1.1 are required. IBM Watson published the "Watson Developer Cloud Security Overview" [32]. In that document, they comply with the HIPPA Act and support authentication and authorization based on IBM Bluemix and end-to-end encryption following HTTPS (via TLS 1.2). Amazon Web Services (AWS) published "Architecting for HIPAA Security and Compliance on Amazon Web Services" [33]. They support PHI data encryption and protection in AWS during data transmission and storage using the AWS Key Management Service (KMS). They support SQL or Oracle database encryption for secure data storage and end-to-end encryption based on TLS or IPSec VPNs. AWS uses the AES-256 block cipher for data encryption on the database.

In research results on human care service security, Arunkumar and Anbuselvi [34] proposed secure cloud computing using the AES block cipher during PHI data storage in the cloud. Kumar et al. [35] proposed an IoT- and cloud-based patient monitoring system using the block ciphers of AES, DES, and Blowfish for data security. They suggested that the Blowfish block cipher is the most appropriate algorithm for their proposed health care system. Zhang et al. [36] described survey results on searchable encryption schemes and proposed searchable encryption to provide security and privacy on healthcare applications. Mohit et al. [37] proposed a mutual authentication protocol for a cloud-computing-based health care system. Mekala et al. [38] proposed a homomorphic encryption technique for healthcare multi-cloud computing which uses the Dynamo DB and Amazon Web Services (AWS). Mhatre et al. [39] compared attribute-based encryption for health records in cloud storage. They proposed multiauthority attribute-based encryption and provided an adequate, effective, and expressive solution to health record security problems. Zhao et al. [40] proposed attribute-based encryption with nonmonotonic access structures supporting fine-grained attribute

Table 2: AVX2 intrinsic functions for Simeck implementation.

Operations	AVX2 intrinsic functions
Load	*_mm256 loadu si256((m256i∗)x)*
Store	*_mm256 storeu si256((m256i∗)x,y)*
Set	*_mm256 set1 epi16(a)* *_mm256 set1 epi32(a)*
Bitwise AND	*_mm256 and si256(x,y)*
Bitwise OR	*_mm256 or si256(x,y)*
Bitwise XOR	*_mm256 xor si256(x,y)*
Shift left by r-bits	*_mm256 slli epi16(x,r)* *_mm256 slli epi32(x,r)*
Shift right by r-bits	*_mm256 srli epi16(x,r)* *_mm256 srli epi32(x,r)*
Rotation left by r-bits	*_mm256 or si256(_ mm256 slli epi16(x,r),* *_mm256 srli epi16(x,16-r))* *_mm256 or si256(_ mm256 slli epi32(x,r),* *_mm256 srli epi32(x,32-r))*

revocation in m-healthcare. However, they noted that their methods have problems such as large ciphertext sizes and lower efficiency rates.

3. Proposed Method

In this section, we propose efficient parallel implementation methods which use Simeck family block ciphers with AVX2 SIMD and adaptive encryption based on Simeck family block cipher AVX2-optimized implementation for human care services.

For the efficient parallel implementation of the Simeck family block cipher using AVX2, we used AVX2 intrinsic functions and optimized the AVX2 SIMD pipeline to avoid data hazards (aka: stall) and to enhance the performance level.

3.1. Efficient AVX2 Intrinsic Functions for Simeck. There are two methods for implementing the algorithm using AVX2. The first involves the use of AVX2 assembly instructions, and the second relies on AVX2 intrinsic functions. AVX2 intrinsic functions can correspond to AVX2 assembly instructions, as described earlier. We used the AVX2 intrinsic functions presented in Table 2. To implement the Simeck family block cipher, it is necessary to implement the data load from normal data to the AVX2 registers for the AVX2 SIMD operation, the data store from the AVX2 register to normal data, the data set to set the AVX2 register value as the Simeck encryption round key, and the bitwise AND, bitwise OR, bitwise XOR, shift left/right, and rotation operations for the Simeck encryption round function.

For the implementations of Simeck32/64 and Simeck 64/128, we used 256-bit AVX2 m256i data as follows: 16-bit × 16 and 32-bit × 8, respectively. AVX2 does not support vector rotation operations, and it only supports vector shift operations such as shift left/right via the r-bit intrinsic functions in Table 2. For this reason, it was necessary to create a rotation operation using the AVX2 vector shift left/right intrinsic functions and bitwise OR intrinsic functions shown in Table 2. For efficient rotation operations, we used 16-r and 32-r parts for left rotation operation by the r-bit AVX2 intrinsic functions shown in Table 2 as a precalculated constant value. For example, for rotation left by five bits on 16-bit × 16 AVX2 data x, we used the AVX2 intrinsic functions of *_mm256 or si256 (_mm256 slli epi16(x,5)* and *_mm256 srli epi16(x,11))*.

3.2. AVX2 SIMD Pipeline Optimization. For the optimization of the AVX2 SIMD pipeline, if we reused the AVX2 SIMD data which was the result data immediately before the operation as operand data during the next operation, it has a data dependency issue and therefore incurs the read-after-write (RAW) data hazard (aka: stall). This data hazard means that it requires several clock cycles to reload the data which was the result data immediately before the operation.

To avoid this data hazard, we rescheduled the order of operations for Simeck encryption. In the Simeck encryption round function shown in Figure 1, the order of operations is as follows: ROL5 (left) → AND (ROL5 (left), left) → XOR (right, AND (ROL5 (left), left)) → ROL1 (left) → XOR (ROL1 (left), XOR (right, AND (ROL5 (left), left))) → XOR (round key, XOR (right, AND (ROL5 (left), left))) → exchange left and right. If we implement the Simeck encryption round function in the above order, it can cause a data hazard. We rescheduled the order of operations as follows: ROL5 (left) → AND (ROL5 (left), left) → XOR (right, AND (ROL5 (left), left)) → ROL1 (left) → XOR (ROL1 (left), XOR (right, AND (ROL5 (left), left))) → round key Load → backup = left → XOR (round key, XOR (right, AND (ROL5 (left), left))) → right = backup to avoid data hazard.

For high performance during the encryption of big data, we maximize the usage of AVX2 registers by reusing AVX2 registers which are not used during each operation and considering AVX2 registers having plaintext blocks and round key blocks.

3.3. Adaptive Encryption for Human Care Service. Service servers for a human care service process massive amounts of data from various sensors or devices. However, the data sizes can differ in each case because each sensor or device has a different data transmission period. For this reason, it is necessary to encrypt data according to the data size each time when receiving or storing data in the database. To solve this problem, we proposed an adaptive encryption method for a human care service. Adaptive encryption can encrypt big data according to the data size. For adaptive encryption, we implemented AVX2-optimized Simeck32/64 and Simeck64/128 according to the encryption data block size. In the case of Simeck32/64, it can encrypt data from 16 blocks to 64 blocks. When it does this, each block size is 16 bits. Simeck64/128 can encrypt data from 8 blocks to 32 blocks. Each block size is 32 bits. Adaptive encryption uses the proposed AVX2-optimized Simeck32/64, and Simeck64/128 supports various block sizes for efficient data encryption of massive data according to the number of plaintext blocks.

Require: Plaintext Blocks $P \in \{P_1, P_2 \cdots, P_{blkNum}\}$ Round keys *RK*, The Number of Plaintext Blocks *blkNum*, Encryption Type *t*
Ensure: Ciphertext Blocks $C \in \{C_1, C_2, \cdots C_{blkNum}\}$
1: if t == 1 **then**
2: let *n1* =*blkNum*/64; and *r1* =*blkNum* %64;
3: let *n2* =*r1*/48; and *r2* =*r1*%48;
4: let *n3* =*r2*/32; and *r3* =*r2*%32;
5: let *n4* =*r3*/16; and *r4* =*r3*%16;
6: **if** r4 ≥1 **then**
7: *n4* ++;
8: *Pad* ();
9: **for** *i* from 1 to *n1* **do**
10: *Simeck* 32/64 *Enc SIMD* 64*Blks* $(RK, \_P_{partial_i}, C_{partial_i})$;
11: **for** *i* from 1 to *n2* **do**
12: *Simeck* 32/64 *Enc SIMD* 48*Blks* $(RK, \_P_{partial_{n1+i}}, C_{partial_{n1+i}})$;
13: **for** *i* from 1 to *n3* **do**
14: *Simeck* 32/64 *Enc SIMD* 32*Blks* $(RK, \_P_{partial_{n2+i}}, C_{partial_{n2+i}})$;
15: **for** *i* from 1 to *n4* **do**
16: *Simeck* 32/64 *Enc SIMD* 16*Blks* $(RK, \_P_{partial_{n3+i}}, C_{partial_{n3+i}})$;
17: **else**
18: let *n1* =*blkNum*/32; and *r1* =*blkNum* %32;
19: let *n2* =*r1*/24; and *r2* =*r1*%24;
20: let *n3* =*r2*/16; and *r3* =*r2*%16;
21: let *n4* =*r3*/8; and *r4* =*r3*%8;
22: **if** r4 ≥1 **then**
23: *n4* ++;
24: *Pad* ();
25: **for** *i* from 1 to *n1* **do**
26: *Simeck* 64/128 *Enc SIMD* 32*Blks* $(RK, \_P_{partial_i}, C_{partial_i})$;
27: **for** *i* from 1 to *n2* **do**
28: *Simeck* 64/128 *Enc SIMD* 24*Blks* $(RK, \_P_{partial_{n1+i}}, C_{partial_{n1+i}})$;
29: **for** *i* from 1 to *n3* **do**
30: *Simeck* 64/128 *Enc SIMD* 16*Blks* $(RK, \_P_{partial_{n2+i}}, C_{partial_{n2+i}})$;
31: **for** *i* from 1 to *n4* **do**
32: *Simeck* 64/128 *Enc SIMD* 8*Blks* $(RK, \_P_{partial_{n3+i}}, C_{partial_{n3+i}})$;
33 *Return C*

Algorithm 1: Adaptive encryption based on Simeck.

Algorithm 1 describes the proposed adaptive encryption algorithm. Input data for the adaptive encryption algorithm are plaintext blocks $P \in \{P_1, P_2, \ldots, P_{\text{blkNum}}\}$, the Simeck encryption round key RK, the number of plaintext blocks blkNum, and the encryption type *t*. The encryption type *t* indicates which Simeck family block cipher will be used for adaptive encryption. If it has a value of 1, this indicates that Simeck32/64 encryption will be used for adaptive encryption. The number of plaintext blocks blkNum refers to the total number of plaintext blocks, and plaintext blocks consist of blkNum blocks. Adaptive encryption using Simeck32/64 or Simeck64/128 has a similar algorithm routine. If the user wants to use Simeck32/64, it supports four types of blocks: 64, 48, 32, and 16 bits. Therefore, during the adaptive encryption process, this method determines how many encryption operations to run according to the block number and the number of plaintext blocks. From line 2 to line 5, it calculates *n*1, *n*2, *n*3, and *n*4 which refer to the number of encryptions for each supported block number. In the case of *n*4, if the remainder value *r*4 is 1 or more, it means that there are a number of plaintext blocks for encryption (less than 16 blocks), and we increase the value of *n*4 and use data padding to adjust the number of blocks used in the encryption process. In the case of data padding, there are many padding standards, such as PKSC7 and X923, among others. We assume that the padding method follows the relevant standard. After calculating the number of encryptions at each supported block number, we encrypt plaintext blocks according to the calculated number using each AVX2-optimized Simeck encryption supported fixed block size.

4. Experiment and Evaluation

In this section, we describe the experimental environment, procedures, and analysis method used to assess the performance of the proposed AVX2-optimized Simeck family block cipher implementation and adaptive encryption techniques.

Table 3: Proposed Simeck family block cipher performance.

Cipher	Number of blocks	Cycles/byte
Simeck32/64	16	5.3125
	32	3.9063
	48	3.5417
	64	3.5859
Simeck64/128	8	6.8750
	16	4.6875
	24	4.6146
	32	4.6875

4.1. Experiment. We developed the proposed AVX2-optimized Simeck family block cipher using AVX2 intrinsicfunctions and C language. We used GCC compiler version 5.4.0 with the compile options of *-O3-fomit-frame-pointer-mavx2-march = nativ-std = c99-mtune = native-fwrapv-funroll-loops* to optimize the compiling process. We conducted the experiment on a computer with the following specifications: Ubuntu 16.04.3 LTS 64 bit and Intel (R) Core (TM) i7-6700 CPU (@3.40 GHz, 32 GB RAM). We measured the average performance of 10,000,000 times encryption to ensure an accurate performance measurement.

4.2. Evaluation of AVX2-Optimized Simeck Implementation Performance. Table 3 describes the performance of the proposed AVX2-optimized Simeck family block cipher implementation. The performance is calculated in units of cycles/byte.

As shown in Table 3, Simeck32/64 encryption for 48 blocks and Simeck64/128 encryption for 34 blocks have the best performance, with 3.5417 cycles/byte and 4.6146 cycles/byte, respectively. Simeck32/64 encryption for 16 blocks and Simeck64/128 encryption for 8 blocks show lower performance outcomes because these encryptions encrypt only one 256-bit AVX2 register value. Hence, the performance in these cases is slightly lower. In the cases of Simeck32/64 encryption for 64 blocks and Simeck64/128 encryption for 32 blocks, these encryptions result in lower performance than Simeck32/64 encryption for 48 blocks and Simeck64/128 encryption for 32 blocks. Although Simeck32/64 encryption for 64 blocks and Simeck64/128 encryption for 32 blocks use the maximum number of AVX2 registers for efficiency, during the compile procedure, the GCC compiler does not use the AVX2 register designated by the C source code (AVX2 intrinsic functions).

For an objective evaluation of the Simeck AVX2-optimized implementation with the proposed methods, we implemented the SIMON family block cipher [3] with the proposed methods and measured the performance of the SIMON family block cipher with the proposed methods on the same environment. Table 4 describes the performances of the SIMON family block cipher with the proposed methods. SIMON32/64 encryption on 64 blocks has 4.1797 cycles/byte but the proposed Simeck32/64 encryption method on 64 blocks runs at 3.5859 cycles/byte (the best performance of Simeck32/64 is 3.5417 cycles/byte on 48 blocks of encryption) while SIMON64/128 encryption on 32 blocks operates at 5.7031 cycles/byte but proposed Simeck64/128 encryption on the same block size has 4.6875 cycles/byte (the best performance of Simeck64/128 encryption on 24 blocks runs at 4.6146 cycles/byte). The cause of the performance difference is the difference between the number of rotations left by the r-bit operations for Simeck and SIMON encryption round functions. The SIMON encryption round function is $R_k(x, y) = (y \oplus f(x) \oplus k, x)$, $f(x) = (Sx \& S^8 x) \oplus S^2 x$, and it requires three rotation left operations by r-bit ($S^r x$, r can be 1 bit, 8 bits, or 2 bits), three bitwise eXclusive-OR operations ($\oplus$), and one bitwise AND operation (&). However, the Simeck encryption round function is $R_k(x, y) = (y \oplus f(x) \oplus k, x)$, $f(x) = (x \& S^5 x) \oplus S^1 x$, and it requires two rotation left operations by r-bit (r can be 1 bit or 5 bits), three bitwise eXclusive-OR operations, and one bitwise AND operation. For this reason, the proposed Simeck32/64 AVX2-optimized implementations show performance improvements of 23.53%, 25.20%, 26.47%, and 16.56% over SIMON32/64 with the proposed methods for each respective number of blocks. With the proposed Simeck64/128 AVX2-optimized implementations, the performance improvements are 28.64%, 43.33%, 30.02%, and 21.67% over SIMON64/128 with the proposed methods for each corresponding number of blocks. Specifically, if comparing the best performance between the proposed Simeck and SIMON with the proposed methods, the best performance of the proposed Simeck32/64 and Simeck64/128 shows performance improvements of 18.01% and 23.59%, respectively, over the best performance of SIMON32/64 and SIMON64/128 with the proposed methods.

Table 4: SIMON family block cipher with proposed methods performance.

Cipher	Number of blocks	Cycles/byte
SIMON32/64	16	6.5625
	32	4.8906
	48	4.4792
	64	4.1797
SIMON64/128	8	8.8438
	16	6.7188
	24	6.0000
	32	5.7031

Table 5: Performance comparison result.

Cipher	Cycles/byte
SIMON64/128 for 64 bytes [30]	7.72
Proposed Simeck64/128 for 16 blocks (64 bytes)	*4.6875*

We also compared the performance capabilities between the proposed AVX2-optimized Simeck64/128 implementation for 16 blocks (64 bytes) and the SIMON64/128 for 64 bytes on eBACS SUPERCOP [30] (https://bench.cr.yp.to/results-stream.html). The SIMON64/128 for 64 bytes on

Table 6: Predicted performance of adaptive encryption.

Type of adaptive Enc.	Cycles/byte
Adaptive Enc.(Simeck32/64)	$(3.5859 \times n1) + (3.5417 \times n2) + (3.9063 \times n3) + (5.3125 \times n4)$
Adaptive Enc.(Simeck64/128)	$(4.6875 \times n1) + (4.6146 \times n2) + (4.6875 \times n3) + (6.8750 \times n4)$

Table 7: Comparison evaluation between previous works and the proposed method.

Method	Encryption	Optimized implementation	Optimized for big data
Arunkumar et al. [34]	AES	X	X
Kumar et al. [35]	AES, DES, Blowfish	X	X
Zhang et al. [36]	Searchable encryption	X	X
Zhao et al. [40]	Attribute-based encryption	X	X
Proposed method	*Simeck*	*O*	*O*

eBACS SUPERCOP [30] is implemented using AVX2. The performance of the SIMON64/128 for 64 bytes on eBACS SUPERCOP [30] is measured on systems using the AMD64, Kaby Lake (906e9), and 2017 Intel Xeon E3-1220 v6 processors with 4× in a 3000 MHz environment. Table 5 describes the performance comparison results between the proposed AVX2-optimized Simeck64/128 implementation for 16 blocks (64 bytes) and SIMON64/128 for 64 bytes on eBACS SUPERCOP [30]. The proposed AVX2-optimized Simeck64/128 implementation for 16 blocks (64bytes) is approximately 60.72% faster than the SIMON64/128 for 64 bytes on eBACS SUPERCOP [30] with the same data size (64 bytes).

4.3. Evaluation of Adaptive Encryption. Table 6 describes the predicted performance of the proposed adaptive encryption approach based on the performance of the AVX2-optimized Simeck family block cipher implementation. If the total number of plaintext blocks is N, it can be written as $N = n1 + n2 + n3 + n4$. Here, $n1$ refers to the number of Simeck32/64 encryptions for 64 blocks or Simeck64/128 for 32 blocks, $n2$ is the number of Simeck32/64 encryptions for 48 blocks or Simeck64/128 encryption for 24 blocks, $n3$ is the number of Simeck32/64 encryptions for 32 blocks or Simeck64/128 encryptions for 16 blocks, and $n4$ denotes the number of Simeck32/64 encryptions for 16 blocks or Simeck64/128 encryptions for eight blocks according to Algorithm 1. We can then predict the performance of the proposed adaptive encryption by aggregating the product of each number of Simeck encryptions for specific blocks and each proposed AVX2-optimized Simeck encryption performance outcome, as shown in Table 6.

The advantage of the proposed adaptive encryption becomes apparent when the number of plaintext blocks is increased. In such a case, the performance is also increased. Therefore, the proposed adaptive encryption method with the proposed AVX2-optimized Simeck family block cipher implementation is an adequate and efficient data encryption method which can be used by human care services to comply with the HIPPA security requirement (data encryption).

4.4. Comparison Evaluation. We compared previous works on human care service security and the proposed method from optimized implementation and optimized for big data perspectives. Table 7 describes the comparison evaluation results. From the perspective of optimized implementation, previous works focused on applying an encryption method to a human care service for security. Accordingly, they did not optimize the encryption implementation step. However, with the proposed methods, these are optimized using AVX2 SIMD considering the SIMD implementation condition, and the Simeck family block cipher outperforms AES [4]. From the perspective on optimization for big data, previous works such as those by Arunkumar et al. [34] and Kumar et al. [35] applied encryption to a cloud computing environment without considering the processing of big data. However, the proposed adaptive encryption method considers efficient big data processing on a cloud environment or on the server side to provide security (data encryption) with good availability following the HIPAA security requirements for a human care service.

5. Conclusion

In a human care service, service providers must comply with the HIPAA security and privacy requirement in the United States for security. For this reason, human care service providers such as MS Azure, IBM Watson, and Amazon Web Service provide security following HIPAA, and there are many research results on encryption for a human care service. However, previous research results focused generally on applying encryption methods to a human care service without considering efficiency and availability. If lightweight block ciphers are used at the end devices or sensors for various human care application services, cloud or service platform servers must deal with encrypted big data from the various end devices or sensors of their human care application service as rapidly as possible to ensure the availability of their human care application service considering different received data sizes according to the data transmission cycles of the end devices. In this paper, to solve these problems, we proposed AVX2-optimized Simeck family block cipher implementations supporting various numbers of blocks with good performance to provide efficiency and availability of data encryption at the cloud or on the server side. The

proposed AVX2-optimized Simeck32/64 encryption for 48 blocks has 3.5417 cycles/byte while Simeck64/128 encryption for 24 blocks has 4.6146 cycles/byte. For an objective evaluation of the proposed methods, we compared Simeck family block cipher AVX2-optimized encryption and SIMON family block cipher AVX2-optimized encryption based on the proposed method. The best performance outcomes with the proposed Simeck32/64 and Simeck64/128 were correspondingly 18.01% and 23.59% performance improvements as compared to the best performance of SIMON32/64 and SIMON64/128 with the proposed methods. Specifically, the proposed AVX2-optimized Simeck64/128 implementation method for 16 blocks (64 bytes) is approximately 60.72% faster than the SIMON64/128 for 64 bytes on eBACS SUPERCOP [30] with the same data size (64 bytes). We also proposed adaptive encryption based on AVX2-optimized Simeck encryption for efficient big data encryption on the cloud or server side to ensure high performance. The strong points of the proposed methods are AVX2-optimized Simeck block cipher implementation and an efficient adaptive encryption method for efficient big data encryption to enhance the availability of a human care application service. However, a limitation of the proposed methods is that it has not been applied to an actual human care service with optimized Simeck implementation using AVX2. To address this issue, we will apply the proposed methods on IBM Bluemix and IBM Watson for a human care service and conduct research on the efficient AVX-512-optimized implementation of the Simeck family block cipher in the future.

Data Availability

The proposed Simeck family block cipher implementation source codes are uploaded to the GitHub repository (https://github.com/pth5804/Simeck_AVX2). SIMON family block cipher implementation source codes based on the proposed methods are also uploaded to the GitHub repository (https://github.com/pth5804/SIMON_AVX2).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work of Taehwan Park and Howon Kim was supported by the Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. 2017-0-01791, Development of Security Technology for Energy Platform and Device). This work of Hwajeong Seo was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (no. NRF-2017R1C1B5075742). This work of Sokjoon Lee was supported by Institute for Information & Communications Technology Promotion (IITP) grant funded by the Korea government (MSIT) (no. B0717-16-0097, Development of V2X Service Integrated Security Technology for Autonomous Driving Vehicle). This paper was proofread by the KAIST Language Center.

References

[1] K. Andrew, A. Steve, and G. Shane, *HIPAA Compliance Guide*, Tech. Rep. HIPAA Compliancy Group, 2017.

[2] J. Daemen and V. Rijmen, *The Design of Rijndael: AES-The Advanced Encryption Standard*, Springer Science & Business Media, 2013.

[3] R. Beaulieu, S. Treatman-Clark, D. Shors, B. Weeks, J. Smith, and L. Wingers, "The SIMON and SPECK lightweight block ciphers," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, San Francisco, CA, USA, June 2015.

[4] 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*, T. Güneysu and H. Handschuh, Eds., Springer, Berlin, Heidelberg, 2015.

[5] S. Kölbl and A. Roy, "A brief comparison of Simon and Simeck," in *International Workshop on Lightweight Cryptography for Security and Privacy*, Springer, Cham, 2016.

[6] N. Bagheri, "Linear cryptanalysis of reduced-round Simeck variants," in *International Conference in Cryptology in India*, Springer, Cham, 2015.

[7] K. Qiao, L. Hu, and S. Sun, "Differential analysis on Simeck and Simon with dynamic key-guessing techniques," in *International Conference on Information Systems Security and Privacy*, Springer, Cham, 2016.

[8] K. Zhang, J. Guan, B. Hu, and D. Lin, "Security evaluation on Simeck against zero-correlation linear cryptanalysis," *IET Information Security*, vol. 12, no. 1, pp. 87–93, 2018.

[9] S. Wang, "Related-key differential analysis of round-reduced Simeck," in *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, Palladam, India, February 2017.

[10] S. Sadeghi and N. Bagheri, "Improved zero-correlation and impossible differential cryptanalysis of reduced-round Simeck block cipher," *IET Information Security*, vol. 12, no. 4, pp. 314–325, 2018.

[11] L. Qin, H. Chen, and X. Wang, "Linear hull attack on round-reduced Simeck with dynamic key-guessing techniques," in *Australasian Conference on Information Security and Privacy*, Springer, Cham, 2016.

[12] B. Ryabko and A. Soskov, *The Distinguishing Attack on Speck, Simon, Simeck, HIGHT and LEA*, Cryptology ePrint, 2018.

[13] M. Yoshikawa, Y. Nozaki, and K. Asahi, "Multiple rounds aware power analysis attack for a lightweight cipher Simeck," in *2016 IEEE Second International Conference on Big Data Computing Service and Applications (Big Data Service)*, Oxford, UK, 2016.

[14] Y. Nozaki, Y. Ikezaki, and M. Yoshikawa, "Tamper resistance of IoT devices against electromagnetic analysis," in *2016 IEEE International Meeting for Future of Electron Devices, Kansai (IMFEDK)*, Kyoto, Japan, June 2016.

[15] V. Nalla, R. A. Sahu, and V. Saraswat, "Differential fault attack on Simeck," in *Proceedings of the Third Workshop on Cryptography and Security in Computing Systems-CS2'16*, Prague, Czech Republic, January 2016.

[16] Y. Nozaki, Y. Ikezaki, and M. Yoshikawa, "Double-rounds-driven electromagnetic analysis attack for a lightweight block cipher Simeck and its evaluation," *Electronics and Communications in Japan*, vol. 100, no. 12, pp. 29–38, 2017.

[17] T. Park, H. Seo, B. Bae, and H. Kim, "Efficient implementation of Simeck family block cipher on 8-bit processor," *Journal of Information and Communication Convergence Engineering*, vol. 14, no. 3, pp. 177–183, 2016.

[18] T. Park, H. Seo, G. Lee, and H. Kim, "Efficient implementation of Simeck family block cipher on 16-bit MSP430," in *2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN)*, Milan, Italy, July 2017.

[19] R. AlTawy, R. Rohit, M. He, K. Mandal, G. Yang, and G. Gong, "sLiSCP: Simeck-based permutations for lightweight sponge cryptographic primitives," in *International Conference on Selected Areas in Cryptography*, pp. 129–150, Springer, Cham, 2017.

[20] G. Lento, *Optimizing Performance with Intel Advanced Vector Extensions*, 2014, online.

[21] S. Gueron and V. Krasnov, "Speed records for multi-prime RSA using AVX2 architectures," in *Information Technology: New Generations*, pp. 237–245, Springer, Cham, 2016.

[22] A. Faz-Hernández and J. López, "On software implementation of arithmetic operations on prime fields using AVX2," *XIV Simpósio Brasileiro em Segurança da Informaçao e de Sistemas Computacionais*, pp. 338–341, 2014.

[23] A. Faz-Hernández and J. López, "Fast implementation of Curve 25519 using AVX2," in *International Conference on Cryptology and Information Security in Latin America*, Springer, Cham, 2015.

[24] P. Martins and L. Sousa, "Enhancing data parallelism of fully homomorphic encryption," in *International Conference on Information Security and Cryptology*, Springer, Cham, 2016.

[25] R. Cabral and J. López, "Software implementation of SHA-3 family using AVX2," *Simpósio Brasileiro em Segurança da Informaçao e de Sistemas Computacionais*, pp. 330–333, 2014.

[26] C. Du, G. Bai, and H. Chen, "Towards efficient implementation of lattice-based public-key encryption on modern CPUs," in *2015 IEEE Trustcom/BigDataSE/ISPA*, Helsinki, Finland, August 2015.

[27] S. Gueron and F. Schlieker, "Software optimizations of NTRUEncrypt for modern processor architectures," in *Information Technology: New Generations*, pp. 189–199, Springer, Cham, 2016.

[28] M. Hamburg, *Integer Module LWE Key Exchange and Encryption: the Three Bears*, Rambus, Inc., 2017.

[29] R. Steinfeld, A. Sakzad, and R. K. Zhao, *Titanium: Proposal for a NIST Post-Quantum Public-Key Encryption and KEM Standard*, Monash University, 2017.

[30] D. J. Bernstein and T. Lange, *eBACS: ECRYPT Benchmarking of Cryptographic Systems*February 2018, https://bench.cr.yp.to.

[31] Microsoft, "Microsoft Azure HIPAA/HITECH Act implementation guidance," Tech. Rep., Microsoft, 2015.

[32] IBM, "Watson Developer Cloud security overview," Tech. Rep., IBM, 2016.

[33] A. W. Services, "Architecting for HIPAA security and compliance on Amazon Web Services," Tech. Rep., Amazon Web Services, 2018.

[34] R. J. Arunkumar and R. Anbuselvi, "Enhancement of cloud computing security in the health care sector," *International Journal of Computer Science and Mobile Computing*, vol. 6, no. 8, pp. 23–31, 2017.

[35] B. V. Kumar, M. Ramaswami, and P. Swathika, "Data security on patient monitoring for future healthcare application," *International Journal of Computer Applications*, vol. 163, no. 6, pp. 20–23, 2017.

[36] R. Zhang, R. Xue, and L. Liu, "Searchable encryption for healthcare clouds: a survey," *IEEE Transactions on Services Computing*, p. 1, 2017.

[37] P. Mohit, R. Amin, A. Karati, G. P. Biswas, and M. K. Khan, "A standard mutual authentication protocol for cloud computing based health care system," *Journal of Medical Systems*, vol. 41, no. 4, p. 50, 2017.

[38] M. Bhaskar, M. Ashok, and M. D. A. Hasan, "Homomorphic encryption algorithm used in multi-format data in collaborated healthcare multi cloud computing," *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*, vol. 2, no. 6, pp. 80–84, 2017.

[39] S. Mhatre, A. V. Nimkar, and S. N. Dhage, "Comparative study on attribute-based encryption for health records in cloud storage," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, Bangalore, India, May 2017.

[40] Y. Zhao, P. Fan, H. Cai, Z. Qin, and H. Xiong, "Attribute-based encryption with non-monotonic access structures supporting fine-grained attribute revocation in M-healthcare," *International Journal of Network Security*, vol. 19, no. 6, pp. 1044–1052, 2017.